MATHEMATICS OF COMPUTATION, VOLUME 33, NUMBER 147
JULY 1979, PAGES 881–890

A Marching Technique for Nonseparable Equations*

By Louis W. Ehrlich

Abstract. A multiple-shooting marching technique is described which is applicable to arbitrary block tridiagonal matrices derived from nonseparable difference equations which are solved many times. Comparison with other methods on a particular problem shows the method to be competitive with respect to time and storage.

Introduction. Our interest here is in solving the difference equation which results from approximating a nonseparable partial differential equation. There are a variety of techniques available; and if our problem is only to be solved a few times, most any method will do. However, if the system is to be repeatedly solved many times (such as the stream function equation in a time dependent Navier-Stokes problem), our interest then centers on fast, efficient methods consistent with our storage availability. We present here a multiple shooting marching method which appears to be competitive in time, consistent with limited storage, with other available methods. The method is applied to a particular problem and compared with other techniques.

1. The Marching Technique. Consider the linear system

$$Ax = b, \tag{1.1}$$

where

$$A = \begin{bmatrix} D_1 & B_1 & & & \\ C_2 & D_2 & B_2 & & 0 \\ & C_3 & D_3 & B_3 & \\ & 0 & & & B_{M-1} \\ & & & C_M & D_M \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ x_M \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \\ \\ b_M \end{bmatrix}, \tag{1.2}$$

and where the D_i are $N \times N$ tridiagonal matrices, the B_i and C_i are appropriate diagonal or tridiagonal matrices, and x and b are vectors partitioned accordingly. (For simplicity we assume C_i and B_i are $N \times N$.) Assume the matrix is nonsymmetric and the corresponding difference equation is nonseparable. We propose a marching technique to solve the above.

The marching technique can be described briefly as follows. Consider the nth equation block; i.e.,

$$C_n x_{n-1} + D_n x_n + B_n x_{n+1} = b_n. \tag{1.3}$$

Received April 17, 1978.
AMS (MOS) subject classifications (1970). Primary 65N20, 65F05, 15A06.
*This work was supported by the Department of the Navy, Naval Sea Systems Command, under Contract No. N00024-78-C-5384.

0025-5718/79/0000-0101/$03.50

Assuming we can solve for x_{n+1}, we have

$$x_{n+1} = B_n^{-1}(b_n - C_n x_{n-1} - D_n x_n). \tag{1.4}$$

Thus, if we guess x_1, then we can march along solving for the other x_i. The final equation

$$C_M x_{M-1} + D_M x_M = b_M \tag{1.5}$$

will in general not be satisfied. We, thus, must modify our guess for x_1 so that the last equation will be satisfied on a second march through the region. This modification is done by inverting an "influence matrix", in the terminology of Roache [11], [12]. Bank and Rose [2], [3] consider separable equations and look at the marching procedure as an LU decomposition of the given matrix. Further, the separability allows the influence matrix to be inverted via polynomial decomposition. Roache [11], [12], on the other hand, actually produces the influence matrix and uses a direct inversion scheme. Since we do not have separability, we use the latter. The influence matrix is actually produced by marching over the region with x_1 equal to all zeros except for a single unknown which is equal to 1. This is repeated for each of the N unknowns. The errors in x_M, determined from (1.5), for each unknown then become the appropriate columns of the $N \times N$ influence matrix.

Because marching techniques are inherently unstable [2], [11], we propose a multiple shooting method as used by Bank and Rose [2], [3], i.e., the marching is limited to blocks of q lines. Thus, we first permute the matrix so that we have

$$\bar{A} = \left[\begin{array}{ccccc:ccc} G_1 & & & & & v_1 & & \\ & G_2 & & 0 & & w_2 & v_2 & \\ & & G_3 & & & & w_3 & v_3 \\ 0 & & & & & & & v_{k-1} \\ & & & G_k & & & & w_k \\ \hdashline r^T_{m_1} & s^T_{m_1} & 0 & & & D_{m_1} & & 0 \\ & r^T_{m_2} & s^T_{m_2} & & & & D_{m_2} & \\ & 0 & & r^T_{m_{k-1}} & s^T_{m_{k-1}} & 0 & & D_{m_{k-1}} \end{array}\right], \tag{1.6}$$

where the blocks of q lines are to be separated at lines $m_0 = 0, m_1, m_2, \ldots, m_{k-1}, m_k = M + 1$ and where

$$G_i = \begin{bmatrix} D_{m_{i-1}+1} & B_{m_{i-1}+1} & & \\ C_{m_{i-1}+2} & D_{m_{i-1}+2} & B_{m_{i-1}+2} & \\ & & & \\ & & & D_{m_i-1} \end{bmatrix}, \tag{1.7}$$

$$v_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ B_{m_i-1} \end{bmatrix}, \qquad w_i = \begin{bmatrix} C_{m_{i-1}+1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{1.8}$$

$$r_{m_i}^T = (0 \quad 0 \cdots 0 \quad C_{m_i}),$$
$$s_{m_i}^T = (B_{m_i} \quad 0 \cdots 0), \qquad i = 1, 2, \ldots, k. \tag{1.9}$$

Following Bank and Rose [2], we consider an LU decomposition of the above matrix, i.e., $\bar{A} = LU$, where

$$L = \left[\begin{array}{ccccc:ccc} I_1 & & & 0 & & & & \\ & I_2 & & & & & 0 & \\ & & & & I_k & & & \\ \hdashline r_{m_1}^T G_1^{-1} & s_{m_1}^T G_2^{-1} & & & & I_{m_1} & & \\ & r_{m_2}^T G_2^{-1} & s_{m_2}^T G_3^{-1} & & & & 0 & \\ & & & & & & I_{m_2} & \\ & & r_{m_{k-1}}^T G_k^{-1} & s_{m_{k-1}} G_k^{-1} & & & & I_{m_{k-1}} \end{array}\right], \tag{1.10}$$

$$U = \begin{bmatrix} G_1 & & & v_1 & & 0 & & \\ & G_2 & & w_2 & v_2 & & & \\ 0 & & & & & & v_{k-1} & \\ & & G_k & 0 & & & w_k & \\ & 0 & & \Delta_{11} & \Delta_{12} & 0 & & \\ & & & \Delta_{21} & \Delta_{22} & \Delta_{23} & & \\ & & & & \Delta_{32} & \Delta_{33} & \Delta_{34} & \\ & & & 0 & & & & \Delta_{k-2,k-1} \\ & & & & & \Delta_{k-1,k-2} & & \Delta_{k-1,k-1} \end{bmatrix}, \tag{1.11}$$

and where

$$
(1.12)\qquad \left\{\begin{array}{l}
\Delta_{11} = D_{m_1} - r_{m_1}^T G_1^{-1} v_1 - s_{m_1}^T G_2^{-1} w_2 \\
\Delta_{12} = -s_{m_1}^T G_2^{-1} v_2 \\
\Delta_{21} = -r_{m_2}^T G_2^{-1} w_2 \\
\left.\begin{array}{l}
\Delta_{ii} = D_{m_i} - r_{m_i}^T G_i^{-1} v_i - s_{m_i}^T G_{i+1}^T w_{i+1} \\
\Delta_{i,i+1} = -s_{m_i}^T G_{i+1}^{-1} v_{i+1},\ i \neq k-1 \\
\Delta_{i,i-1} = -r_{m_i}^T G_i^{-1} w_i
\end{array}\right\},\quad i = 2, 3, \ldots, k-1.
\end{array}\right.
$$

Let

$$
(1.13)\qquad \Delta = \begin{bmatrix}
\Delta_{11} & \Delta_{12} & & 0 & \\
\Delta_{21} & \Delta_{22} & \Delta_{23} & & \\
 & \Delta_{32} & \Delta_{33} & & \\
 & & 0 & & \Delta_{k-2,k-1} \\
 & & & \Delta_{k-1,k-2} & \Delta_{k-1,k-2}
\end{bmatrix}.
$$

To solve $\Delta y = c$ we use the method described in [5], [7], i.e., we let

$$
(1.14)\qquad \Delta = \bar{L}\bar{U} = \begin{bmatrix}
L_1 & & \\
H_2 & L_2 & 0 \\
0 & & \\
 & H_{k-1} & L_{k-1}
\end{bmatrix}\begin{bmatrix}
I_1 & Q_1 & & 0 \\
 & I_2 & Q_2 & \\
0 & & & Q_{k-2} \\
 & & & I_{k-1}
\end{bmatrix},
$$

where

$$
(1.15)\qquad \left\{\begin{array}{l}
L_1 = \Delta_{11} \\
Q_1 = L_1^{-1}\Delta_{12} \\
\left.\begin{array}{l}
H_i = \Delta_{i,i-1} \\
L_i = \Delta_{ii} - \Delta_{i,i-1} Q_{i-1} \\
Q_i = L_i^{-1}\Delta_{i,i+1},\ i \neq k-1
\end{array}\right\},\quad i = 2, \ldots, k-1.
\end{array}\right.
$$

Then

$$
(1.16)\qquad \begin{array}{l}
\Delta y = \bar{L}\bar{U}y = c, \\
\bar{L}t = c, \\
\bar{U}y = t,
\end{array}
$$

$$\begin{cases} t_1 = L_1^{-1} c_1, \\ t_i = L_i^{-1}(c_i - \Delta_{i,i-1} t_{i-1}), \quad i = 2, \ldots, k-1, \end{cases} \tag{1.17}$$

$$\begin{cases} y_{k-1} = t_{k-1}, \\ y_i = t_i - Q_i y_{i+1}, \quad i = k-2, k-3, \ldots, 1. \end{cases} \tag{1.18}$$

To solve (1.1) we write

$$Ax = LUx = b \tag{1.19}$$

and solve

$$Lz = b, \quad Ux = z. \tag{1.20}$$

Thus, from (1.10) and (1.11)

$$\begin{cases} z_j = b_j, \quad j = 1, \ldots, k, \\ z_{m_i} = b_{m_i} - r_{m_i}^T G_i^{-1} z_i - s_{m_i}^T G_{i+1}^{-1} z_{i+1}, \quad i = 1, \ldots, k-2, \\ z_{m_{k-1}} = b_{m_{k-1}} - r_{m_{k-1}}^T G_k^{-1} z_k, \end{cases} \tag{1.21}$$

$$\begin{bmatrix} x_{m_1} \\ x_{m_2} \\ \\ x_{m_{k-1}} \end{bmatrix} = \Delta^{-1} \begin{bmatrix} z_{m_1} \\ z_{m_2} \\ \\ z_{m_{k-1}} \end{bmatrix}, \tag{1.22}$$

$$x_j = G_j^{-1}(z_j - w_j z_{m_{j-1}} - v_j z_{m_j}), \quad j = 1, \ldots, k, \tag{1.23}$$

where $z_0 = 0$.

Assume our region is rectangular and $N \times M$ and assume further we march in the M direction. The procedure is as follows. The preliminary computation involves the calculation, LU decomposition and storage of the influence matrix of each of the G_i, $i = 1, \ldots, k$. This will require $k \times N^2$ storage spaces. Also, the matrix Δ must be computed, decomposed and stored, as noted in (1.15). Δ_{ii} is replaced by the LU decomposition of L_i. $\Delta_{i,i+1}$ is replaced by Q_i, and $\Delta_{i,i-1}$ is left intact. All this requires $(3k-5)N^2$ spaces, or a total of $(4k-5)N^2$ spaces.

Each solution of (1.1) requires the following computation. Equation (1.21) is solved for the z_{m_i} by a marching solution for $G_i^{-1} z_i$ and $G_{i+1}^{-1} z_{i+1}$. Since $r_{m_i}^T$ and $s_{m_i}^T$ are matrices, each row of which contain at most 3 nonzero components, the inner products require at most three multiplications. Next, Eq. (1.22) is solved by using (1.17) and (1.18). Finally, (1.23) is solved by marching again.

Solving (1.21) requires roughly $12M \cdot N$ operations to march, including solving the tridiagonal system for each line, kN^2 operations to invert the influence matrices,

and another $12M \cdot N$ operations for the second march. Equation (1.17) requires $(3k - 5)N^2$ operations for the inversions. Finally, Eq. (1.23) requires again as many operations as the marching of (1.21). Thus, the total is roughly $48NM + (5k - 1)N^2$ operations per solution. The choice of which direction to march depends roughly on the term $5kN^2$. Generally, one marches along the direction M, where $M > N$, for both time and storage considerations.

2. A Numerical Example. Recently, a fair amount of work has been done investigating the numerical transformation of arbitrary regions onto rectangular regions, particularly in the study of fluid flow problems (see e.g., [14], [15]). The partial differential equations are then solved numerically on a rectangular grid. Although this tends to make the region simple, the equations are transformed into a mess, involving, among other things, cross derivatives which in turn lead to 9 point difference equations. Thus, the stream function equation in a Navier-Stokes problem leads to a difference equation which is not separable, and a matrix which is not symmetric. In the literature [15], these matrices have been solved by Successive Overrelaxation (SØR). Both a pointwise varying relaxation factor (ω) and a constant ω have been tried [15]. (The cross derivatives were assumed negligible in estimating ω.) Experience indicates an "optimum" constant ω is more effective [15]. For this report, the constant optimum ω was determined empirically. Line or block SØR was not tried since overall computer time usually favors point SØR [10a].

The marching technique described in this memo was applied to such a problem. We describe the results, and compare these with other methods for solving the same problem.

The problem has been described in detail elsewhere [6]. An arbitrary region in the (r, z) plane is numerically mapped into a rectangular region in the (u, v) plane by the mapping

$$\begin{cases} \alpha r_{uu} - 2\beta r_{uv} + \gamma r_{vv} = 0, \\ \alpha z_{uu} - 2\beta z_{uv} + \gamma z_{vv} = 0, \end{cases} \tag{2.1}$$

where

$$\begin{cases} \alpha = z_v^2 + r_v^2, \\ \beta = z_u z_v + r_u r_v, \\ \gamma = z_u^2 + r_u^2, \\ \\ J = z_u r_v - z_v r_u, \\ ZU = (\delta_u z)J/r, \\ ZV = (\delta_v z)J/r, \end{cases} \tag{2.2}$$

where $(\delta_u z)$ is the first central difference in the u direction, etc. (Equation (2.1) was

actually solved by SØR since it only had to be solved once.) The partial differential equation is transformed into an equation whose finite difference stencil has the form

$$\begin{bmatrix} \beta & \gamma - ZU & -\beta \\ \alpha - ZV & -2(\alpha+\gamma) & \alpha + ZV \\ -\beta & \gamma + ZU & \beta \end{bmatrix}_{ij}, \tag{2.3}$$

where the coefficients are determined from the numerical solution of (2.1) and (2.2), and vary from point to point. See [6] for details.

In solving the above problem by the multiple shooting marching method, one must determine how many steps to march to keep the error within a reasonable bound. The number of steps to march is actually a function of the matrix problem, the precision of the arithmetic (i.e., the computer being used) and the accuracy desired. Roache [11], [12] discusses this point, noting that the number of steps can vary from 6 to 60. Tables I and II note the errors encountered in our problem on the IBM 360 in double precision. We actually used 9 marching steps, making the error consistent with the SØR convergence criteria we used.

TABLE I

Error vs. Number of Marching Steps
$N = 15, M = 75$

Steps/March	8	9	10
k	10	9	8
Residual	$.8 \times 10^{-8}$	$.14 \times 10^{-6}$	$.16 \times 10^{-5}$

TABLE II

Error vs. Number of Marching Steps
$N = 15, M = 15$

Steps/March	4	6	8	9	10	12
k	4	3	2	2	2	2
Residual	$.5 \times 10^{-12}$	$.5 \times 10^{-8}$	$.1 \times 10^{-6}$	$.5 \times 10^{-5}$	$.8 \times 10^{-3}$	7

3. Other Methods. Besides SØR, several other methods were tried and compared to the marching procedure. The methods chosen were ones which appeared to have a chance for success with nonsymmetric matrices of the type considered here. There may well be other methods which are competitive.

(i) *Conjugate Gradient* (CG). Basically the CG method is designed for symmetric positive definite matrices. However, it does work for some nonsymmetric matrices if one considers a "splitting" [4], "preconditioning" [1] or "acceleration" [16] of the given matrix. A promising approach for most nonsymmetric matrices appears to be the incomplete LU decomposition of A [8], an extension of the results of [10].

TABLE III

Comparison of Methods
$N = 15, M = 90$

Method	*Storage*	*Time*
SØR	0	43 iteration, 1 sec
CG–Direction 1	$12 \cdot N \cdot M = 16{,}200$	37 iter, 2.3 sec
CG–Direction 2	$12 \cdot N \cdot M$	16 iter, 1.1 sec
MM–Direction 1	$12 \cdot N \cdot M$	19 iter, 1.39 sec
MM–Direction 2	$12 \cdot N \cdot M$	18 iter, 1.32 sec
Marching	$(4k - 5)N^2 + NM = 11025$ $(k = 12)$	.2 sec
Equation (1.15)	$(2M - 1)N^2 = 39825$*	.41 sec*
Band	$(2N + 3)NM = 44055$*†	.27 sec*
Sparse	112,400 locations + 16,760 integers* (including fill in)	.18 sec*

* data here for 15 × 89 grid
† see text, Section 3(iii)

One considers L and U such that the nonzero elements of A equal the corresponding elements of LU. Then LU is an approximate inverse of A. Thus, we solve the system

$$U^{-1}L^{-1}AX = U^{-1}L^{-1}b, \tag{3.1}$$

where $U^{-1}L^{-1}A$ is an approximation to the identity. The ordering of the equations also plays a role. Thus, if $N = 15, M = 90$, then ordering first in the N direction (Direction 1) leads to a matrix with semibandwidth of 15. Ordering first in the M direction (Direction 2) leads to a semibandwidth of 90. Table III contains some results of the CG method applied to (3.1).

(ii) *Manteuffel's Method* (MM). Manteuffel [9] suggested a Chebeshev scheme for matrices, whose eigenvalues have real parts which are positive. It is not clear that A satisfies this property, but $U^{-1}L^{-1}A$ from Eq. (3.1) appears to. The appropriate constants needed to determine the iteration factors were computed using the programs listed in [9]. With these optimum values, the method was applied. Table III contains some results.

(iii) *Direct Methods.* There were three direct methods which we considered. The first was the method of Eq. (1.15) applied directly to (1.1) [5], [7]. Assuming the coefficients of the given matrix are available, the amount of storage space needed is about 4 times that needed for the marching method. Although faster than SØR, it is about twice as slow as the marching method. However, as with all direct methods considered here, accuracy is within computational round-off.

The second direct method considered was band elimination. The routine LEQT1B was used from the IMSL library. Although this routine uses partial pivoting, indications were that no pivoting was actually performed. Thus, one could recode the routine to avoid pivoting and save some storage. The storage entry in Table III reflects this fact although the actual amount of storage used was 65415.

The third direct method, was the use of the Yale sparse matrix package described in [13]. The amount of storage needed was about 10 times that of the marching procedure (see Table III). Not counting set-up time, it was slightly less than 10% faster, which is marginal considering the enormous amount of space required. For a more stable problem requiring a smaller k, even this timing might be matched. Other versions of the sparse matrix package reduce storage but increase timing.

4. Conclusions. It is clear from Table III that SØR is preferred to any other iterative method tried, because its simplicity requires less space and less time. On the other hand, the direct methods are faster and more accurate, but require more storage to implement. In between we have the multiple shooting marching method. For all of the problems tried (varying M), this method was about 4 to 5 times faster than SØR, with comparable accuracy. However, it should be added that one can reduce the final error by allowing more SØR iterations, but one must restart the marching problem to attain higher accuracy.

To check relative times for SØR and marching on a 5 point difference equation, the above problem was run with $\beta_{ij} = 0$. For $N = 15, M = 90$ the SØR method took 43 iterations ($\omega_0 = 1.70$) for a total time of .666 seconds while the marching method, with $k = 12$, took .177 seconds. This particular difference equation was slightly more stable than the test problem which could lead to a smaller k with possible further reduction in time.

Applied Physics Laboratory
The Johns Hopkins University
Laurel, Maryland 20810

1. O. AXELSSON, *On Preconditioning and Convergence Acceleration in Sparse Matrix Problems,* CERN 74-10, Geneva.

2. R. E. BANK & D. J. ROSE, "Marching algorithms for elliptic boundary value problems. I: The constant coefficient case," *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 792–829.

3. R. E. BANK, "Marching algorithms for elliptic boundary value problems. II: The variable coefficient case," *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 950–970.

4. P. CONCUS, G. H. GOLUB & D. P. O'LEARY, "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, Editors), Academic Press, New York, 1976.

5. L. W. EHRLICH, *Iterative vs. a Direct Method for Solving Fourth Order Elliptic Difference Equations,* Proc.–A. C. M. National Meeting, Los Angeles, Calif., 1966, pp. 29–35.

6. L. W. EHRLICH, "The numerical solution of a Navier-Stokes problem in a stenosed tube: A danger in boundary approximations of implicit marching schemes," *Computers and Fluids.* (To appear.)

7. A. C. HINDMARSH, *Solution of Block-Tridiagonal Systems of Linear Algebraic Equations,* Lawrence Livermore Laboratory, UCID-30150, 1977.

8. D. S. KERSHAW, *The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations,* Lawrence Livermore Laboratory, UCRL-78333, 1977.

9. J. A. MANTEUFFEL, *An Iterative Method for Solving Nonsymmetric Linear Systems with Dynamic Estimation of Parameters,* Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, UIUCDCS-R-75-758, 1975.

10. J. A. MEIJERINK & H. A. VAN DER VORST, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric *M*-matrix," *Math. Comp.,* v. 31, 1977, pp. 148–162.

10a. I. J. PEARSON & B. KAPLAN, *Computer Time Comparison of Point and Block Successive Overrelaxation,* Report AFIT-TR-70-6, Air Force Institute of Technology, School of Engineering, Wright-Patterson AFB, Ohio, 1970.

11. P. J. ROACHE, *A Direct Method for the Discretized Poisson Equation,* Sandia Report SC-RR-70-579, 1971.

12. P. J. ROACHE, "Marching methods for elliptic problems: Part I", *Numerical Heat Transfer,* v. 1, 1978, pp. 1–25.

13. S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ & A. H. SHERMAN, *Yale Sparse Matrix Package* II. *The Nonsymmetric Codes,* Res. Report #114, Dept. of Comput. Sci., Yale Univ., 1977.

14. J. F. THOMPSON, F. C. THAMES & C. W. MASTIN, "Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies," *J. Computational Phys.,* v. 15, 1974, pp. 299–319.

15. J. F. THOMPSON, F. C. THAMES & C. W. MASTIN, *Boundary-Fitted Curvilinear Coordinate Systems for Solution of Partial Differential Equations on Fields Containing any Number of Arbitrary Two-Dimensional Bodies,* NASA CR-2729, 1977.

16. D. M. YOUNG, JR. & L. J. HAYES, *Notes on the Conjugate Gradient Method,* CNA Report, Univ. of Texas, Austin.

17. D. M. YOUNG, JR., *Iterative Solution of Large Linear Systems,* Academic Press, New York, 1971.