An Efficient Direct Solver for Separable and Non-Separable Elliptic Equations

RANGARAO V. MADALA

Naval Research Laboratory, Washington, DC 20375

(Manuscript received 13 February 1978, in final form 2 August 1978)

ABSTRACT

A new method for the numerical solution of elliptic difference equations called stabilized error vector propagation (SEVP) is derived. This method has most of the advantages of the error vector propagation algorithm and, in addition, is stable for all grid sizes. By solving Poisson's equation with Dirichlet boundary conditions, SEVP is found to be 3 to 10 times faster than competitive direct methods on a vector computer and requires an order of magnitude smaller computer memory. SEVP is at least 10 times faster than successive overrelaxation. Applications of this method to algorithms using both five- and nine-point stencils as well as stretched grids are discussed.

1. Introduction

The numerical solution of elliptic partial differential equations is frequently required for atmospheric problems. These equations may be separable or nonseparable and their solutions are subject to a variety of boundary conditions. If we write the finite-difference form of the differential equation as $\mathbf{Ax} = \mathbf{F}$, where $\mathbf{A}$ is the coefficient matrix, $\mathbf{F}$ the forcing function and $\mathbf{x}$ the required solution, then $\mathbf{A}$ can be diagonally dominant, weakly dominant or even nondiagonally dominant. Even though a variety of iterative and direct elliptic equation solvers have been developed, no one method appears to be computationally efficient for all types of equations. The iterative methods such as SOR (successive overrelaxation), ADI (alternate direction implicit) and hybrid methods such as optimized block implicit relaxation (Dietrich *et al.*, 1975) are very fast for diagonally dominant equations; however, they converge slowly or may even diverge for the weakly or nondiagonally dominant equations.

The direct solvers developed by Hockney (1965), Buneman (1969), Rosmond and Faulkner (1976), Bank and Rose (1977) and Bank (1977) have no convergence problems but are restricted to separable equations. For nonseparable equations the direct method popularized by Lindzen and Kuo (1969) (hereafter referred to as LK) and the method described by Crout (1941) are available but require large amounts of computer memory and thus are limited to small arrays. Erlich's (1978) extension of multiple-shooting methods to nonseparable equations requires a factor of 4 less auxiliary memory than the LK and Crout methods, but the memory requirements are still too great to solve difference equations with a large number of grid points [O(100) on standard computers].

The error vector propagation method (EVP) (Roche, 1971, 1978; Hirota *et al.*, 1970) is applicable to any type of elliptic equation and requires an order of magnitude less memory than LK. However, as shown by McAvaney and Leslie (1972), EVP is unstable for more than 40 grid points in the sweep direction.

In this paper we develop a procedure which stabilizes EVP called the stabilized error vector propagation method (SEVP). In Section 2 a review of EVP (on which SEVP is based) to solve an elliptic difference equation using five grid stencil is given. The SEVP algorithm is described in Section 3. A comparison of the computer time and memory requirements for several representative direct and iterative solvers is given in Section 4. The extension of the EVP and SEVP methods to equations using nine-point stencils is given in Section 5. A summary is given in Section 7.

2. Error vector propagation method (EVP)

A general two-dimensional elliptic equation in differential form can be written

$$A(X,Y)\frac{\partial^2 x}{\partial X^2} + B(X,Y)\frac{\partial^2 x}{\partial X \partial Y} + C(X,Y)\frac{\partial^2 x}{\partial Y^2} + D(X,Y)\frac{\partial x}{\partial X} + E(X,Y)\frac{\partial x}{\partial Y} + G(X,Y)x = F(X,Y), \quad \text{(1a)}$$

where $A(X,Y)C(X,Y) - B^2(X,Y) > 0$.

For the case $B(X,Y) = 0$, Eq. (1a) can be written

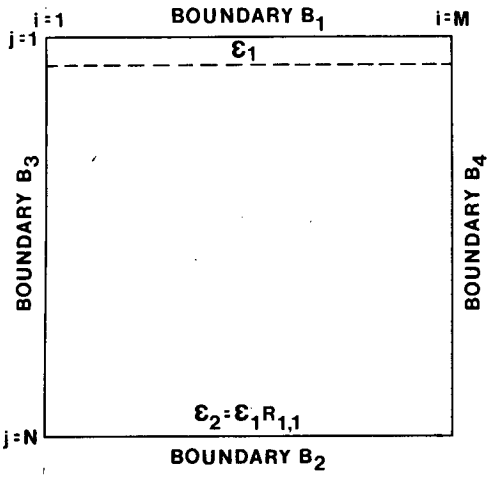

FIG. 1. The region over which the elliptic equation is solved by EVP method. B_1, B_2, B_3 and B_4 are the boundaries.

in finite-difference form (using a five-point stencil); the result is a general two-dimensional elliptic equation of the form

$$AX(i,j)x(i-1,j) + AY(i,j)x(i,j-1) + BB(i,j)x(i,j) + CX(i,j)x(i+1,j) + CY(i,j)x(i,j+1) = F(i,j), \quad (1b)$$

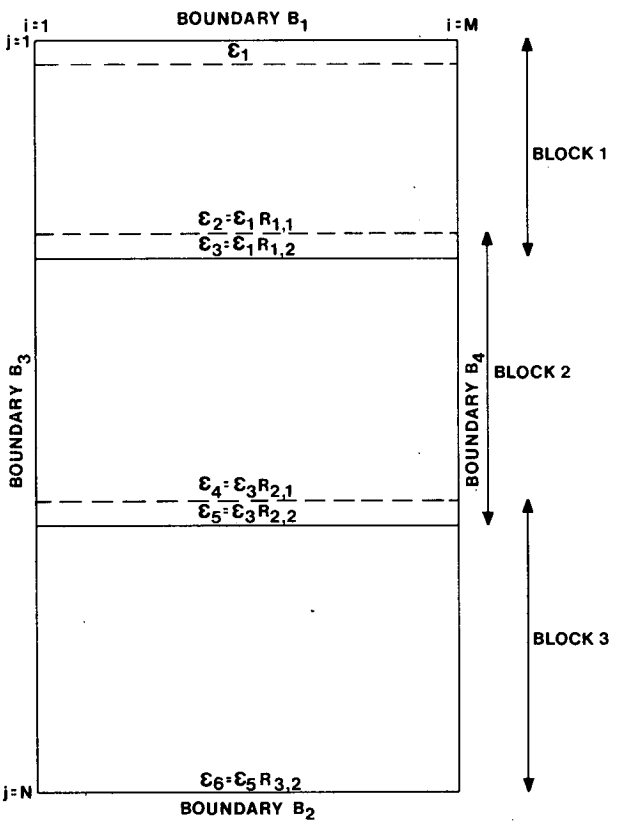

FIG. 2. A three-block division of the region for the SEVP method.

where the coefficients AX, AY, BB, CX and CY may be functions of both i and j, and F is the forcing function. We now review the EVP procedure described by Roche (1971, 1977) for solving Eq. (1b) over a rectangular region shown in Fig. 1 using Diriclet boundary conditions. By rearranging the terms, (1) can be written as

$$x(i,j+1) = [F(i,j) - AX(i,j)x(i-1,j) - AY(i,j)x(i,j-1) - BB(i,j)x(i,j) - CX(i,j)x(i+1,j)][CY(i,j)]^{-1}. \quad (2)$$

It is clear from Eq. (2) that if we know the solution on any two consecutive rows, then we can march in j to obtain the required solution. Since the solution is known only on the boundaries B_1, B_2, B_3 and B_4, the EVP method requires an initial guess of the solution on the row interior to B_1 to march forward in j. Based on this initial guess, a particular solution, $x^P(i,j)$ is obtained which satisfies Eq. (2) and all the boundary conditions except along B_2. If we assume that the particular solution deviates from the real solution $x(i,j)$ by $x^H(i,j)$, then

$$x(i,j) = x^P(i,j) + x^H(i,j). \quad (3)$$

Substituting (3) into (2) and noting that the particular solution satisfies (2) and boundary conditions at B_1, B_3, B_4, we obtain the homogeneous equation

$$x^H(i,j+1) = -[AX(i,j)x^H(i-1,j) + AY(i,j)x^H(i,j-1) + BB(i,j)x^H(i,j) + CX(i,j)x^H(i+1,j)][CY(i,j)]^{-1}, \quad (4)$$

with zero boundary conditions along B_1, B_3, B_4. Along B_2 we find from Eq. (3)

$$\mathbf{x}^H(N) = \mathbf{x}(N) - \mathbf{x}^P(N), \quad (5)$$

where the vector $\mathbf{x}^H(N)$ whose elements are $x^H(i,N)$ for $i = 2, 3, \ldots, M-1$ can be called the error vector.

We can relate the homogeneous solution on any two rows by using $(M-2)$ independent vectors, where M represents total number of grid points in i including boundary points. Since the solution along B_2 [given by Eq. (5)] is known, we will relate it to the solution on the row just interior to the boundary B_1. In other words, if the vectors $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ (Fig. 1) represent the solution on the second and last rows of the region, respectively, then

$$\boldsymbol{\epsilon}_2 = \boldsymbol{\epsilon}_1 \mathbf{R}_{1,1}, \quad (6)$$

where $\mathbf{R}_{1,1}$ is the influence matrix. In order to obtain $\mathbf{R}_{1,1}$, we start with the kth unit vector (which has a value unity for the kth element and zero for all the other elements) along the second row and march Eq. (4) in j, with zero boundary conditions along B_1, B_3 and B_4. The kth row elements of the $\mathbf{R}_{1,1}$ matrix are the elements of Eq. (4) along B_2 boundary.

By varying the values of k from 2 to $(M-1)$ we complete $\mathbf{R}_{1,1}$.

From Eq. (6) and the error vector given by Eq. (5) we compute the homogeneous solution on the second row. Using these values on the second row and zero boundary conditions on B_1, B_3 and B_4, we march Eq. (4) in j to obtain the homogeneous solution throughout the region. Superposition of the particular and homogeneous solutions gives the complete solution. Due to round-off errors in computing the inverse of the matrix $\mathbf{R}_{1,1}$, this solution does not satisfy the boundary condition along B_2 exactly. The accuracy of the solution is improved by recomputing the homogeneous part of the solution a few times. Normally about six iterations are required for 20 grid points in marching direction to obtain an error of 10^{-14} on a computer with 56-bit word precision. The number of iterations required increases as the number of grid points in the marching direction is increased up to about 25; beyond 25 the method becomes unstable.

The influence matrix $\mathbf{R}_{1,1}$ depends only on the coefficients of the elliptic equation and therefore can be computed without knowledge of the forcing function and the boundary conditions. This step is called the preprocessor. If we want to solve Eq. (1) repeatedly with the same coefficients but with different forcing functions and boundary conditions, the influence matrix needs to be computed only once and stored in the memory for further use.

Although the EVP method is very efficient and requires a very small amount of computer memory, as previously noted it is unstable for large number of grid points in the marching direction (McAvaney and Leslie, 1972). For computers with 24-bit precision, the limit is about 12 grid points, which may be increased by going to higher precision or by using the two-directional marching method described by Hirota *et al.* (1970). Even with 56-bit precision and two-way marching the limit is extended to only about 40 grid points.

3. Stabilized error vector propagation method (SEVP)

In the SEVP method, the integration region is divided into blocks each of which is stable for the EVP method. Further, any two consecutive blocks has two rows in common. The division of the integration region into three blocks is shown in Fig. 2. As with the EVP method, SEVP is divided into two steps: the preprocessor step and the solution step. In the preprocessor step, $2 \times \text{NBLK} - 1$ influence matrices are computed where NBLK is the total number of blocks. NBLK of the influence matrices relate the values of the homogeneous solution on the second and last rows of each block, while the remaining (NBLK − 1) matrices relate the homogeneous solution on the second rows of consecutive blocks.

a. Preprocessor

The first $(M-2)$ unit vectors on the second row of the first block are used to march in j direction, using (4) and zero boundary conditions on B_1, B_3 and B_4 to obtain influence matrices for the last two rows of the block. If ϵ_1, ϵ_2 and ϵ_3 (shown in Fig. 2) represent the homogeneous solution on the second and last two rows of the first block, then

$$\epsilon_2 = \epsilon_1 \mathbf{R}_{1,1}, \tag{7}$$

$$\epsilon_3 = \epsilon \mathbf{R}_{1,2}, \tag{8}$$

where $\mathbf{R}_{1,1}$ and $\mathbf{R}_{1,2}$ are the influence matrices for the last two rows of the block. Combining (7) and (8) to eliminate ϵ_1 we obtain

$$\epsilon_3 = \epsilon_2 \mathbf{R}_{1,1}^{-1} \mathbf{R}_{1,2} = \epsilon_2 \mathbf{S}_1 \tag{9}$$

The matrices $\mathbf{R}_1$, and $\mathbf{S}_1$ relate homogeneous solutions on the second and last rows of the block and on the last two rows of the block, respectively.

To obtain the influence matrices for the second block, we start with the unit vectors on the second row of the second block, and compute the corresponding values on the first row of the block from Eq. (9) to get the equivalent of a boundary condition for this block. The homogeneous solution on the first row corresponding to the kth unit vector on the second is the kth row of the matrix $\mathbf{S}_1$. Using $(m-2)$ unit vectors on the second row with corresponding vectors on the first row and zero boundary values on B_3 and B_4, we march Eq. (4) in j to obtain influence matrices for the last two rows of the second block. If the vectors ϵ_3, ϵ_4 and ϵ_5 represents a homogeneous solution on the second and last two rows of the block, respectively, then

$$\epsilon_4 = \epsilon_3 \mathbf{R}_{2,1}, \tag{10}$$

$$\epsilon_5 = \epsilon_3 \mathbf{R}_{2,2}, \tag{11}$$

where $\mathbf{R}_{2,1}$ and $\mathbf{R}_{2,2}$ are the influence matrices for the last two rows of the block. Eliminating ϵ_3 from (10) and (11) we obtain

$$\epsilon_4 = \epsilon_5 \mathbf{S}_2 \mathbf{S}_2 = \mathbf{R}_{2,2}^{-1} \mathbf{R}_{2,1}. \tag{12}$$

Repeating the procedure described above for the third block (last block in Fig. 2) we obtain a matrix $\mathbf{R}_{3,2}$ relating the homogeneous solutions on the second and last rows of the block. If ϵ_5 and ϵ_6 represents the homogeneous solutions on these rows, then

$$\epsilon_6 = \epsilon_5 \mathbf{R}_{3,2}. \tag{13}$$

As in the EVP method, all the influence matrices depend only on the coefficients of Eq. (1).

b. Solution

We obtain the required solution by one forward and one backward sweep of the blocks. During the

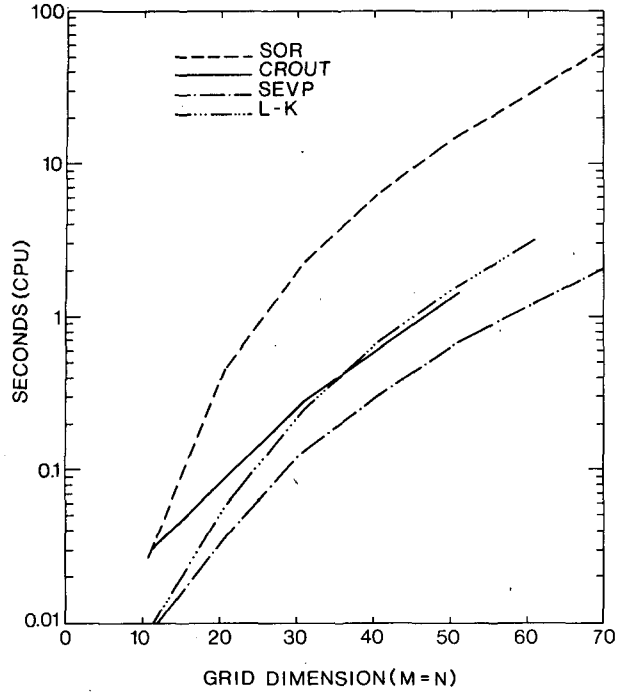

FIG. 3. A comparison of the computing time requirements for the SEVP, LK, Crout and SOR methods.

forward sweep, an approximate solution is obtained which satisfies the equation and the boundary conditions everywhere except on boundary B_2. In the backward sweep, this solution is corrected to obtain the exact solution.

The forward sweep is started with the second row on the first block with an initial guess for the elements of that row and the given boundary values along the first row. This procedure is exactly the same as EVP. When the solution has been marched to the end of the first block, arbitrary values are assigned along the block boundary to obtain the correction vector ϵ_3. The second sweep forward generates a homogeneous solution for the first block. Iterations are applied, if necessary, to reduce round-off error.

For the second block we take the arbitrary solution assigned along the first block boundary (second row second block) and the solution along the first row of the second block and sweep forward again. An arbitrary solution is again imposed along the last row of the second block to obtain the correction vector ϵ_5. On the second (or homogeneous) sweep of the second block we use the correction vectors ϵ_2 and ϵ_3 to form the homogeneous solution. ϵ_3 is computed from ϵ_5 by Eq. (11) and ϵ_2 is obtained from ϵ_3 using Eq. (9). The procedure continues until all blocks are swept out. For the last block, we use the boundary B_2 instead of a guess value to find ϵ_6.

The particular solution we have obtained on the forward sweep satisfies (1) and the given boundary conditions everywhere except on the block boundaries. More importantly, the last block contains the exact solution since the computation of the correction vector ϵ_6 is based on the known boundary conditions along B_2.

The backward sweep now corrects the errors introduced in the particular solution by guessing the solutions along the block boundaries. Using Eqs. (12) and (13) we calculate ϵ_4 and ϵ_5 for the homogeneous sweep of the last block and obtain the total solution for the last block which has two rows in common with the second block. To obtain the homogeneous solution for the second block we need to find the difference ϵ_5' between the particular solution on the last row of the block and the extract solution. One method of obtaining ϵ_5' is to store the particular solution from the forward sweep and substract it from the exact solution. However, it is much easier to recompute the particular solution by backing up three rows into the second block and repeating a small segment of the forward sweep. After ϵ_5' is obtained we sweep the second block to obtain the homogeneous solution which is then added to the particular solution. This procedure is repeated until we finish the blocks. Error reduction iterations are also applied at this step.

The stabilization of EVP is achieved by the introduction of the artificial boundaries. Propagation of error is very severe in EVP, and when the error exceeds the accuracy of the computer it cannot be corrected by the addition of a homogeneous solu-

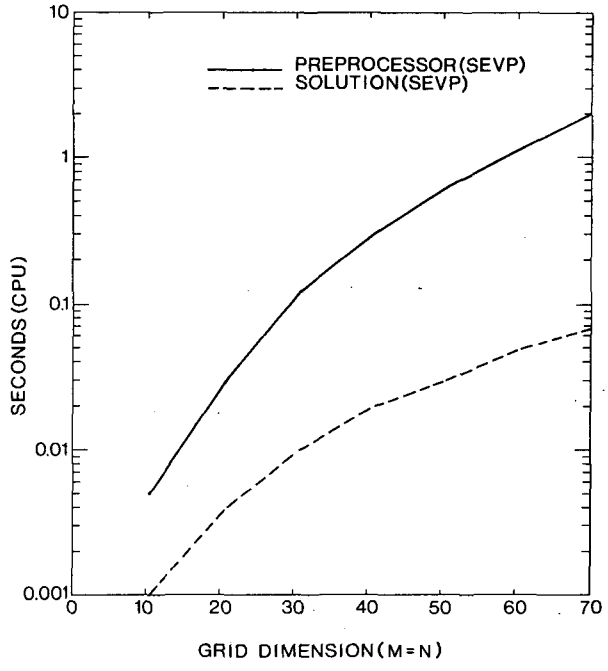

FIG. 4. Computer time taken by the preprocessor and solution steps of the SEVP method.

tion. The introduction of artificial boundaries before the error becomes too large limits the error in each block. Since the influence matrices for small blocks have small error levels, the artificial solution generated by the introduction of these boundaries can be easily corrected to obtain an extremely accurate final solution.

4. Computer time and memory requirements

As a test problem for SEVP, Poisson's equation is solved over a square region with zero boundary conditions. The test problem is also solved with the LK, Crout (1943) and SOR methods. The relaxation by SOR is stopped when the normalized error (normalized with the forcing function) reaches 10^{-4}. A comparison of the computing time taken by these methods and SEVP is given in Fig. 3. When the grid dimension exceeds 60 for LK and 50 for Crout methods the total central memory of the computer available to the author was exceeded.

It is clear from the figure that the SEVP method is more efficient than the other three methods. For $M = N = 40$, the SEVP method is about two times faster than LK and Crout methods and 20 times faster than SOR. The computation times given in Fig. 3 were obtained with double precision (56-bit precision) operations on a vector computer.

Fig. 4 shows the computer time required by the preprocessor step and the solution step of SEVP on a vector computer. The ratio of computing time required by the preprocessor step to that required by the solution step is about 3 for $M = N = 10$ and increases with increasing grid dimension. The LK method can also be separated into a preprocessor step (computing α matrices) and a solution step. For the test problem single precision on 32-bit word computer is adequate for the LK method. However, for the grid size exceeding 40, the solution is found to decrease in accuracy with increase in grid dimension and is unstable when $M = N \geq 65$.

TABLE 1. A comparison of the auxiliary memory requirements (in thousands of words) for SEVP, LK and Crout methods on 32-bit word computer. Double precision variables used except where noted.

Grid size	Crout	SEVP	LK
10	4.4	0.2	1.4***
20	33.2	2.4	8.8***
30	111.6	5.4	28.8***
40	262.4	16.0	67.2***
50	510.0**	25.0	260.0
60	878.4*	50.4	446.4
70	1391.6*	88.2	705.6*
80	2073.6*	115.1	1049.6*
90	2978.4*	178.2	1490.4*
100	4040.0*	220.0	2040.0*

* More than the total central memory available on the computer at the Naval Research Laboratory.
** Normalized error is more than 10^{-4}.
*** Single precision.

TABLE 2. Operation count for the preprocessor and the solution steps of the SEVP and LK methods on a $M \times N$ grid.

	Preprocessor	Solution
SEVP	$6MN^2 + NBLK^{**}(2N^3 + 2N^2)$	$6MN + 2N^2(2 + 2d^*)$
LK	$2MN^3$	$4NM^2$

* d is the number of round-off iterations, $d < 6$.
** $NBLK$ is the number of blocks used in SEVP.

The total operation count (without distinguishing various types arithmetic operations) for the preprocessor and the solution steps of the SEVP and EVP methods is given Table 2. In obtaining this operation count we assumed $M \leq N$ and inversion of $N \times N$ matrix require $2N^3$ operations. The variable d is the number of round-off iterations used to reduce error. Usually d does not exceed 6. Since NBLK is an order of magnitude smaller than N, it can be seen from the table that both preprocessor and solution steps of the SEVP method require less number of operations than those required by the corresponding steps of LK method.

Fig. 5 gives a comparison of the computer time taken by the single precision LK and double precision SEVP methods to solve the test problem. The top two curves are the total computer time (preprocessor and solution steps) required while the lower two curves are for the solution steps only. It is clear from Fig. 5 and Table 2 that SEVP method is faster than LK method for both the preprocessor and solution steps.

Table 1 gives the auxiliary memory requirements for the three direct methods for a 32-bit word computer using double precision. It is apparent from the table that the SEVP method's auxiliary memory requirement is an order of magnitude smaller than that required by the other two direct methods.

5. Selection of the marching direction and the block size

If $NB(k)$ represents the maximum number of points in the marching direction of the kth block, then $NB(k) \leq PA$, where P is a constant which depends only on the computer precision and A represents the minimum value of $CY(i,j)CX^{-1}(i,j)$ if we march in j and of $CX(i,j)CY^{-1}(i,j)$ if we march in i. We can reduce the auxiliary memory required by the SEVP method by choosing the marching direction in such a way that $A \geq 1$. It can also be shown that the method requires less computing time to solve the elliptic equation in the case when $A \geq 1$ than in cases $A < 1$. Therefore, the computing speed of the method increases with $|A|$. On the other hand, the LK and Crout methods deteriorate with the increasing A and became unstable for $A \geq 100$.

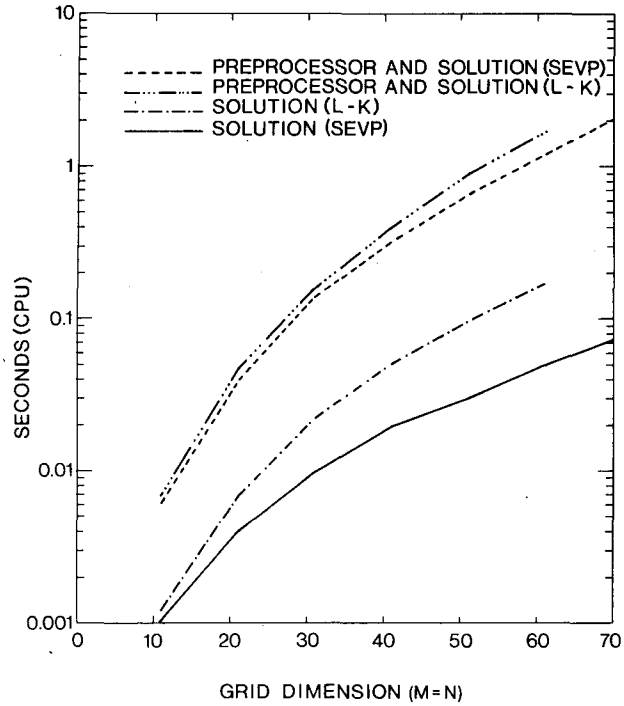

FIG. 5. A comparison of the computer time requirements of single precision LK and double precision SEVP methods using a vector computer with 32-bit word length. The top curves give the total time taken by both steps: the preprocessor and the solution steps. The lower two curves give the time taken by the solution steps.

6. Extension of EVP and SEVP methods for nine-point stencils

Two-dimensional elliptic difference equations with nine-point stencils (over three or five consecutive rows and columns) can be solved by using the EVP and SEVP methods described in Sections 2 and 3 with a few minor modifications. For elliptic difference equation with a nine-point stencil over three consecutive rows and columns, a general equation can be written

$$\begin{aligned} &AY3(i,j)x(i-1,j+1) + BY3(i,j)x(i,j+1) \\ &\qquad + CY3(i,j)x(i+y,j+1) \\ &= F(i,j) - AY1(i,j)x(i-1,j-1) \\ &\quad - BY1(i,j)x(i,j-1) - CY1(i,j)x(i+1,j-1) \\ &\quad - AY2(i,j)x(i-1,j) - BY2(i,j)x(i,j) \\ &\qquad - CY2(i,j)x(i+1,j). \end{aligned} \tag{14}$$

To obtain the homogeneous and particular solutions of Eq. (2) we use the marching procedure of the jth direction by computing the solution explicitly at every point in the $(j+1)$th row in terms of the jth and $(j-1)$th rows. Since (14) couples the solution at three consecutive points on the $(j+1)$th row, we instead compute the solution at all points on the $(j+1)$th row simultaneously with a tridiagonal solver as suggested by Roache (1978). With this minor modification, the EVP and SEVP methods described in Sections 2 and 3 can be used to solve (14). The use of the tridiagonal solver in solving (14) increases the number of operations by $8(M-2)^2(N-1)$ for the preprocessor step and by $8(M-2)^2(2+\alpha)$ for the solution step. A general elliptic difference equation using nine-point stencil over five consecutive rows and columns can be written

$$\begin{aligned} &CY2(i,j)x(i,j+2) \\ &\quad = F(i,j) - CY1(i,j)x(i,j+1) \\ &\qquad - AX2(i,j)x(i-2,j) \\ &\qquad - AX1(i,j)x(i-1,j) - BB(i,j)x(i,j) \\ &\qquad - CX1(i,j)x(i+1,j) - CX2(i,j)x(i+2,j) \\ &\qquad - AY1(i,j)x(i,j-1) \\ &\qquad\qquad - AY2(i,j)x(i,j-2). \end{aligned} \tag{15}$$

The solution of this fourth-order finite-difference equation requires four boundary conditions in each direction which may be specified as the solution on the two outermost rows and columns of the region. As in the previous sections, we obtain the required solution by superposing a particular and a homogeneous solutions. The particular solution is obtained by marching in the jth direction using the given solution (boundary condition) on the first two rows and a guess solution on the next two rows. The result satisfies (15) with the given forcing function and the boundary condition everywhere except on the boundary representing the last two rows. Since the homogeneous solution is the deviation of the particular solution from the required solution, it will satisfy (15) with zero forcing function everywhere and homogeneous boundary conditions on all the boundaries except on the last two rows. On these two rows the boundary conditions are obtained by subtracting the particular solution from the given boundary conditions. Following the procedure described in Section 2, using the homogeneous equation we compute an influence matrix relating a vector representing the homogeneous solution vectors on the third and fourth rows with that to the last two rows. Since each of these vectors has $2(M-4)$ grid points, we use $2(M-4)$ unit vectors to obtaining the influence matrix. Using the influence matrix and the boundary conditions for the homogeneous solution on the last two rows, we can obtain the error solution on the third and fourth rows. With these values for the error solution on the third and fourth rows and zero boundary conditions on the first two rows, first two columns and last two columns, we can obtain the solution for the homogene-

ous equation everywhere in the interior by marching in the jth direction.

The modifications described above to solve Eq. (15) by the EVP method can be easily incorporated in the SEVP procedure. As in Section 3, the region is divided into a number of blocks in such a way each block is stable for the EVP method. The first and last two rows and columns of each block are the boundaries. Thus any two consecutive blocks now have four rows in common. The preprocessor step consists of computing influence matrices for each block that relate the homogeneous solution on the third and fourth rows to the last two rows. We also compute an influence matrix for every two consecutive blocks to relate the homogeneous solution on the first two common rows with the homogeneous solution on the next two common rows. During the forward sweep of the solution step we obtain an approximate solution given guess values at the blocks boundaries where the solution is unknown. During the backward sweep these boundary conditions are connected to obtain the required solution. The procedure is similar to the one described in Section 3.

The size of each influence matrix described above is approximately four times the size of the corresponding influence matrix discussed in Sections 2 and 3. Thus the EVP and SEVP methods require four times more auxiliary memory to solve a fourth-order finite-difference equation than a second-order equation. It can also be shown that the preprocessor step takes approximately eight times more computing time and the solution step takes approximately four times more computing time.

7. Summary

A new method called stabilized error vector propagation (SEVP) is derived to solve elliptic equations. This method is shown to have most of the advantages of the EVP method while being stable for any number of grid points. The stabilization of EVP is accomplished by dividing the integration region into EVP stable overlapping blocks and imposing artificial boundaries between the blocks. In comparison with the LK, Crout and SOR methods this method is faster and requires about an order-of-magnitude smaller computer memory than the LK and Crout methods. While the SEVP algorithm is only described here for the solution of elliptic equations with Dirichlet boundary conditions, the method can easily be extended to other types of boundary conditions such as Neumann, periodic or mixed. The extension of the SEVP to operators using the nine-point stencils is also described.

Acknowledgments. The author would like to thank Dr. Mark Schoeberl for reading the manuscript and making many useful suggestions. The author would like to thank Dr. David Dietrich for stimulating interest in the stabilization of the EVP method. The author also would like to acknowledge the receipt of the computer programs for the LK and Crout methods from Dr. Mark Schoeberl and Dr. Niels Winsor, respectively.

The research was supported by the Office of the Naval Research.

REFERENCES

Bank, R. E., 1977: Marching algorithms for elliptic boundary value problems. II. The variable coefficient case. *SIAM J. Num. Anal.*, **14,** 950–970.

——, and D. J. Rose, 1977: Marching algorithms for elliptic boundary value problems. I. Constant coefficient case. *SIAM J. Num. Anal.*, **14,** 792–828.

Buneman, O., 1969: A compact non-iterative Poisson solver. SVIPR Rep. No. 294, Stanford University, 9 pp.

Crout, P. D., 1941: A short method of evaluating determinents and solving systems of linear equations. *Trans. AIEE*, **60,** 1235.

Dietrich, D., B. E. McDonald, and A. Warn-Varnas, 1975: Optimized block-implicit relaxation. *J. Comput. Phys.*, **18,** 421–439.

Ehrlich, L. W., 1978: A marching technique for non-separable equation. Tech. Rep. RN-14, Applied Physics Laboratory, Johns Hopkins University, 19 pp.

Hirota, I., T. Tokioka, and M. Nishiguchi, 1970: A direct solution of Poisson's equation by generalized sweep-out method. *J. Meteor. Soc. Japan*, **48,** 161–167.

Hockney, R. W., 1965: A fast direct solution of Poisson's equation using Fourier analysis. *J. Assoc. Comput. Mach.*, **12,** 95–113.

Lindzen, R. S., and H.-L. Kuo, 1969: A reliable method for the numerical integration of a large class of ordinary and partial differential equations. *Mon. Wea. Rev.*, **97,** 732–734.

McAvaney, B. J., and L. M. Leslie, 1972: Comments on "A direct solution of Poisson's equation by generalized sweep-out method". *J. Meteor. Soc. Japan*, **50,** 136–137.

Roache, P. J., 1971: A new direct method for the discretized Poisson equation. *Proc. Second Int. Conference on Numerical Methods in Fluid Dynamics*, University of California, Berkeley, Springer-Verlag, 462 pp.

——, 1978: Marching methods for elliptic problems. Part I. Submitted to *Num. Heat Trans.*

Rosmond, T. E., and F. D. Faulkner, 1976: Direct solution of elliptic equations by block cyclic reduction and factorization. *Mon. Wea. Rev.*, **104,** 641–649.