NRL Memorandum Report 6450

A User's Guide to the SEVP Solver: An Efficient Direct Solver for Elliptic Partial Differential Equations

KEITH D. SASH[illegible]

Science Applications International Corporation
P.O. Box 1303, McLean, VA 22102

RANGARAO V. MADALA

Atmospheric Physics Branch
Space Science Division

April 13, 1989

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 6450	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b OFFICE SYMBOL (If applicable) Code 4110	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Arlington, VA 22217	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 61153N PROJECT NO: RR033-03-4A TASK NO: WORK UNIT ACCESSION NO: DN156-046

11 TITLE (Include Security Classification)
A User's Guide to the SEVP Solver - An Efficient Direct Solver for Elliptic Partial Differential Equations

12 PERSONAL AUTHOR(S)
Sashegyi,* K.D. and Madala, R.V.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Interim	FROM _____ TO _____	1989 April 13	42

16 SUPPLEMENTARY NOTATION
*Science Applications International Corporation, McLean, VA 22102

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Numerical solution, (KE) Elliptic partial differential equations Finite difference method Fortran subroutines

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

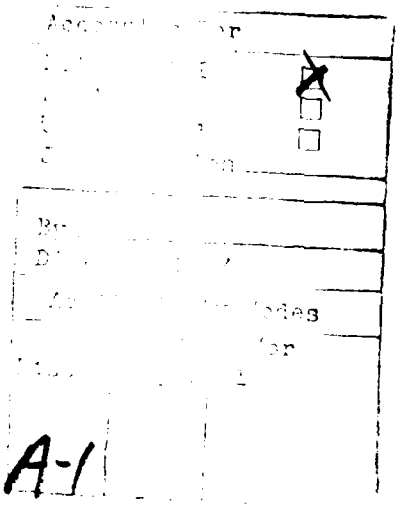
This technical report describes how to use the Stabilized Error Vector Propagation (SEVP) fortran subroutines, which have been developed at the Naval Research Laboratory, to solve an elliptic partial differential equation using finite difference methods. The SEVP method is an efficient direct method which can be used for separable and non-separable elliptic equations. The derivation of the finite difference equations and the use of Dirichlet, Neumann and periodic boundary conditions are demonstrated. The method and a test example are also described.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL R. Madala	22b TELEPHONE (Include Area Code) (202) 767-2226 22c OFFICE SYMBOL Code 4110

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

CONTENTS



A USER'S GUIDE TO THE SEVP SOLVER: AN EFFICIENT DIRECT SOLVER FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

1. INTRODUCTION

This report describes the Stabilized Error Vector Propagation (SEVP) fortran subroutines, which have been developed at the Naval Research Laboratory (NRL) for the solution of elliptic partial differential equations using finite difference methods. The SEVP subroutines have been used extensively with the NRL numerical weather prediction models. Since the code was first developed (Madala, 1978), various enhancements have been made. The code is now also being more widely used at NRL and at other research institutions and a users guide has therefore become necessary.

The SEVP method is an efficient direct method which is used to solve the finite difference approximation to an elliptic partial differential equation of the form

$$a(x,y)\frac{\partial^2\phi}{\partial x^2} + b(x,y)\frac{\partial^2\phi}{\partial y^2} + c(x,y)\frac{\partial\phi}{\partial x} + d(x,y)\frac{\partial\phi}{\partial y} + e(x,y)\,\phi = f(x,y)$$

in which $a(x,y)\,b(x,y) > 0$ on a rectangular domain in some arbitrary x and y coordinate system. Grid points may be of variable spacing. The boundary conditions for $\phi(x,y)$ can be Dirichlet, Neumann, periodic in one direction, or a mixed combination of these. The SEVP solver can be used for separable and non-separable elliptic equations.

Common examples of these elliptic partial differential equations are Poisson's equation $\nabla^2\phi = f$ in two dimensions and the special case when $f \equiv 0$, Laplace's equation in two dimensions. For the boundary conditions in each case, the value of ϕ (Dirichlet boundary condition) or the normal derivative of ϕ (Neumann boundary condition) are specified on each boundary. Poisson's equation, for example, arises in boundary value problems in which a stream function is used to solve for the motion of an incompressible fluid when the vorticity is given. A Poisson's equation is also encountered whenever a steady state potential distribution is produced in response to sources or

Manuscript approved February 27, 1989.

sinks of a quantity, where the rate of flow of the quantity is proportional to the gradient of the potential distribution.

As shown in Madala (1978), the SEVP method, which is based on the error vector propagation (EVP) method (Hirota, et al., 1970; for example), is much faster than successive over-relaxation (by 10 to 20 times) and uses an order of magnitude less storage space than the method of Lindzen and Kuo (1969). Unlike the EVP method, which was shown by McAvaney and Leslie (1972) to be unstable for more than 40 grid points in the marching direction, the SEVP method is stable for any number of grid points. The stabilization is accomplished by dividing the integration region into EVP stable overlapping blocks and imposing artificial boundaries between the blocks.

In this report, the form of the difference equation, boundary conditions and the subroutine arguments for the SEVP solver are first described. We then describe the derivation of the difference equation and boundary conditions from a two-dimensional, elliptic partial differential equation, defined on a rectangular grid. The following sections describe the stabilized error vector propagation method and each of the SEVP subroutines. In the initial set up in subroutine SEVP, the difference equations are modified for the boundary conditions. The sections on the preprocessor subroutine BSM1 and the solution subroutines BSM2, BSM3 may be skipped by the reader who is only interested in using the routines. As an example, Poisson's equation is solved in spherical coordinates on a domain on the surface of the earth bounded by constant latitude and longitude circles. Two test examples demonstrating the use of Dirichlet and Neumann boundary conditions in the y (marching) direction with periodic boundary conditions in x are described. A listing of the test programs is provided in the appendix.

2. SUBROUTINE SEVP

Subroutine SEVP solves a system of linear equations of the form

$$AX(i,j)*X(i-1,j) + AY(i,j)*X(i,j-1) + BB(i,j)*X(i,j) + CX(i,j)*X(i+1,j) + CY(i,j)*X(i,j+1) = F(i,j) \qquad (2.1)$$

for $i = 2,\ldots,M-1$ and $j = 2,\ldots,N-1$.

with boundary conditions for $j = 1$, $j = N$, $i = 1$ and $i = M$ of the form

$$\begin{aligned} X(i,1) &= (1-A11)*X(i,1) + A11*(X(i,2) - F11(i)) \\ X(i,N) &= (1-A1N)*X(i,N) + A1N*(X(i,N-1) + F1N(i)) \quad \text{for } i = 1,\ldots,M \\ &\text{and} \\ X(1,j) &= (1-A21)*X(1,j) + A21*(X(2,j) - F21(j)) \\ X(M,j) &= (1-A2M)*X(M,j) + A2M*(X(M-1,j) + F2M(j)) \quad \text{for } j = 1,\ldots,N \end{aligned} \qquad (2.2)$$

or $X(1,j) = X(M-1,j)$

$X(M,j) = X(2,j)$ for $j = 1,\ldots,N$.

These difference equations result from the discretization of elliptic partial differential equations of the form

$$a(x,y)\frac{\partial^2\phi}{\partial x^2} + b(x,y)\frac{\partial^2\phi}{\partial y^2} + c(x,y)\frac{\partial\phi}{\partial x} + d(x,y)\frac{\partial\phi}{\partial y} + e(x,y)\,\phi = f(x,y) \qquad (2.3)$$

defined in a rectangular domain $x_a < x < x_b$, $y_a < y < y_b$ in which $a(x,y)\,b(x,y) > 0$, and the x and y represent some coordinate system. Grid points can be of variable spacing. Boundary conditions for $\phi(x,y)$ may be Dirichlet or Neumann at the x and y boundaries $x = x_a$, $x = x_b$, $y = y_a$, $y = y_b$; or periodic in x; or a mixed combination of these.

2.1 Description of Arguments

The SEVP subroutines are called by the statement

```
CALL SEVP(AX,AY,CX,CY,BB,RINV,RINV1,RCOR,NBSIZ2,IS,IE,INX,SUMF,
DUMMY1,RTILDA,X,F,ERR,F11,F1N,F21,F2M,M,N,M1,N1,M2,N2,NBLK,
NBLK1,B1,B2,A11,A1N,A21,A2M,ICY,TOL,IFLG)
```

where the arguments are as follows:

ON INPUT

M
The number of points in the i direction, including the boundary.

M1 = M-1, M2 = M-2.

N
The number of points in the j (marching) direction, including the boundary. N must be larger than 4.

N1 = N-1, N2 = N-2.

NBLK
Represents the number of blocks in j (marching) direction. For N larger than 14, NBLK = (N+3)/9 gives blocks of approximately 8 to 13 rows each. For N less than 11 but larger than 4, use NBLK = 1; while for N between 11 and 14, one or two blocks may be used.
Note: For the case of Neumann boundary conditions at boundary j = N (*i.e.* A1N = 1) and with NBLK larger than one, last block is fixed with 7 rows by SEVP solver, to allow for a more efficient convergence of the solution. In this case, use formula to calculate the number of blocks required for the first N-5 rows and then add one to get total number of blocks.

NBLK1
= NBLK-1, for NBLK greater than one.
= 1 for NBLK equal to one.

AX, AY, CX, CY, BB
Two-dimensional arrays of dimension M x N containing the coefficients of the difference equation.

X
Two-dimensional array that on input contains an initial guess for solution on interior points i = 2,...M-1, j = 2,...,N-1; and for Dirichlet boundary conditions, contains the boundary values at the appropriate boundaries. Initial guess might be an average of the boundary values (for Dirichlet BC's) or a zero value (for Neumann BC's), for example.

F
Two-dimensional array of dimension M x N that specifies the forcing on the right hand side of the difference equation.

F11, F1N
One-dimensional arrays of length M that contain coefficients for the Neumann boundary conditions at the j boundaries.

F21, F2M
One-dimensional arrays of length N that contain coefficients for the Neumann boundary conditions at the i boundaries.

A11, A1N, A21, A2M
Real variables which take the value 0.0 for Dirichlet boundary conditions, and 1.0 for Neumann boundary conditions at their respective boundaries. A11 corresponds to j = 1, A1N to j = N, A21 to i = 1, and A2M to i = M. For periodic boundary conditions in i (with ICY = 1), A21 and A2M are zero.

ICY
= 1 for periodic boundary conditions in i.
= 0, otherwise.

TOL
Error tolerance for SEVP solver.

IFLG
= 0 for first call of SEVP solver. Preprocessor, subroutine BSM1 called.
= 1 for subsequent calls if coefficients AX, AY, CX, CY, BB of difference equation and the boundary condition type A11, A1N, A21, A2M, ICY do not change. Preprocessor is then skipped.

ON OUTPUT

X
Two-dimensional array of dimension M x N containing the solution.

ERR
Two-dimensional array of dimension M x N containing the residual error for the difference equation.

OUTPUT ARRAYS COMPUTED BY PREPROCESSOR

These arrays must not be destroyed if SEVP will be called again with IFLG = 1.

RINV
Three-dimensional array of dimension M2 x M2 x NBLK, containing two-dimensional M2 x M2 influence or residual matrices for the blocks. The first NBLK-1 influence matrices relate the solution error on the last interior row of each block to that on the second row of that block. The last two-dimensional matrix is a residual matrix relating the residual error computed on the last interior row to the solution error on the second row of the last block.

RINV1
Three-dimensional array of dimension M2 x M2 x NBLK1, containing NBLK-1 two-dimensional M2 x M2 influence matrices, which relate the solution error on the last two rows of each block, except the last.

WORKING INTEGER ARRAYS

NBSIZ2
One-dimensional integer array of size NBLK, containing the number of interior rows in the j (marching) direction between the end boundaries of the blocks. The first block contains a total of NBSIZ2(1) + 2 rows, while the remaining blocks contain NBSIZ2() + 3 rows each.

IS, IE
One-dimensional integer arrays of size NBLK, defining the beginning and end row of each block, respectively.

INX
One-dimensional integer array of size M containing i indices.

WORKING REAL ARRAYS

SUMF
One-dimensional array of size NBLK containing the sum of the absolute values of the forcing on the last interior row of each block.

RCOR
Two-dimensional array of dimension M x 3 containing set of three row vectors used for marching in j direction.

RTILDA
One-dimensional error vector of size M-2.

B1, B2
One-dimensional work arrays of size M-2.

DUMMY1
Two-dimensional work arrays of size M-2 x M-2.

INTERNAL PARAMETERS IN SUBROUTINE SEVP

IOUT
To print residual errors after each iteration set iout=1, otherwise iout=0.

EPS
The machine precision of computer, used as a lower limit on the tolerance TOL. For the Cray with 64 bit words, single precision floating point words (with 48 bit mantissa) are precise to 14 decimal places or for a machine precision of about 0.5×10^{-14}.

MAXSIZ
Maximum number of points allowed in each block. For the precision of the Cray computer setting MAXSIZ = 15 ensures the solver converges rapidly to a solution.

IX, JX, JF
For 4-sided Neumann, or 2-sided Neumann with cyclic boundary conditions, a boundary value must be prescribed at point (IX, JX) for a solution to be obtained. JF is the nearest interior row to the prescribed boundary point (IX, JX). For boundary point IX=2, JX=1, the nearest interior row is JF=2.

PROGRAM LANGUAGE Fortran-77

PRECISION Single precision on 64 bit computer. For 32 bit computer, subroutines should be modified for double precision.

ORIGINATOR Rangarao V. Madala
Naval Research Laboratory, Washington D.C. 20375

3. DIFFERENCE APPROXIMATION

In this section we show how the difference equation is derived from a two-dimensional elliptic partial differential equation. A two-dimensional elliptic equation of the form

$$a(x,y)\frac{\partial^2\phi}{\partial x^2} + b(x,y)\frac{\partial^2\phi}{\partial y^2} + c(x,y)\frac{\partial\phi}{\partial x} + d(x,y)\frac{\partial\phi}{\partial y} + e(x,y)\,\phi = f(x,y) \qquad (3.1)$$

is assumed to be defined in a rectangular domain $x_a < x < x_b$, and $y_a < y < y_b$, where $a(x,y)\,c(x,y) > 0$, with boundary conditions specified at boundaries $x=x_a$, $x=x_b$, $y=y_a$, and $y=y_b$. Boundary conditions may be Dirichlet, Neumann in x and y or periodic in x.

As an illustration, we define a grid mesh of M by N points (x_i, y_j) with a grid spacing of constant Δx and Δy. However, variable grid spacing can be used, resulting in modification to the coefficients of the difference equation. A second order finite difference approximation of Eq. (3.1), using a five point stencil, is then

$$a_{i,j}\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + b_{i,j}\frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} + c_{i,j}\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\,\Delta x} + d_{i,j}\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\,\Delta y} + e_{i,j}\,\phi_{i,j} = f_{i,j} \qquad (3.2)$$

where $\phi_{i,j} = \phi(x_i, y_j)$,

$f_{i,j} = f(x_i, y_j)$

and $a_{i,j} = a(x_i, y_j)$, etc.

By defining $X(i,j) = \phi_{i,j}$ and

$$AX(i,j) = \frac{a_{i,j}}{\Delta x^2} - \frac{c_{i,j}}{2\,\Delta x}$$

$$AY(i,j) = \frac{b_{i,j}}{\Delta y^2} - \frac{d_{i,j}}{2\,\Delta y}$$

$$CX(i,j) = \frac{a_{i,j}}{\Delta x^2} + \frac{c_{i,j}}{2\,\Delta x}$$

$$CY(i,j) = \frac{b_{i,j}}{\Delta y^2} + \frac{d_{i,j}}{2\,\Delta y} \qquad (3.3)$$

$$BB(i,j) = -\frac{2\,a_{i,j}}{\Delta x^2} - \frac{2\,b_{i,j}}{\Delta y^2} + e_{i,j}$$

$$F(i,j) = f_{i,j}$$

Our difference equation (3.2) can then be written in the form

$$AX(i,j)*X(i-1,j) + AY(i,j)*X(i,j-1) + BB(i,j)*X(i,j) + CX(i,j)*X(i+1,j) + CY(i,j)*X(i,j+1) = F(i,j) \qquad (3.4)$$

for $i = 2,\ldots,M-1$ and $j = 2,\ldots,N-1$.

We will now demonstrate the use of Neumann, Dirichlet and periodic boundary conditions in x.

3.1 Dirichlet Boundary Conditions

For Dirichlet boundary conditions in x, boundary values are given at $x=x_a$ and $x=x_b$

$$\phi(x_a,y) = p(y)$$

and $$\phi(x_b,y) = q(y) \quad \text{for} \quad y_a \leq y \leq y_b$$

For our grid mesh, defining M points in x with

$$x_i = (i-1)\,\Delta x + x_a \quad \text{for } i = 1,\ldots,M$$

where $\Delta x = (x_b-x_a)/(M-1)$, we have

$$\phi(1,j) = p_j$$

$$\phi(M,j) = q_j \text{ for } j = 1,\ldots,N$$

By defining A21 = A2M = 0.0 and

$$X(1,j) = p_j \quad \text{and} \quad X(M,j) = q_j \quad \text{for } j=1,\ldots,N$$

our boundary conditions are in the appropriate form of Eq. (2.2).

3.2 Neumann Boundary Conditions

For Neumann boundary conditions in x, derivatives are specified at the boundaries $x=x_a$ and $x=x_b$

$$\frac{\partial \phi}{\partial x}(x_a,y) = u(y) \quad \text{and} \quad \frac{\partial \phi}{\partial x}(x_b,y) = v(y) \quad \text{for} \quad y_a < y < y_b$$

For our grid mesh, we include two columns of fictitious points at $x_a-\Delta x/2$ and $x_b+\Delta x/2$, and define our x grid points with a shift of $\Delta x/2$, so that

$x_i = (i-1)\,\Delta x + x_a - \Delta x/2 \quad$ for $i = 1,\ldots,M$

with $\Delta x = (x_b-x_a)/(M-2)$. Then using centered differences we have

$$\frac{\phi_{2,j} - \phi_{1,j}}{\Delta x} = u_j \quad \text{and} \quad \frac{\phi_{M,j} - \phi_{M-1,j}}{\Delta x} = v_j \quad \text{for } j = 2,\ldots,N-1$$

By defining A21 = A2M = 1.0,

F21(j) = $\Delta x\, u_j$, and F2M(j) = $\Delta x\, v_j$ for j = 2,...N-1

we have our boundary conditions in the appropriate form of Eq. (2.2).

3.3 Periodic Boundary Conditions

Suppose that ϕ is a periodic function in x with period $L = x_b-x_a$ so that

$$\phi(x+x_b,y) = \phi(x+x_a,y)$$

With a column of points included at $x=x_b+\Delta x$, we define our grid mesh over a x domain of $x_a \le x \le x_b+\Delta x$ so that

$x_i = (i-1)\,\Delta x + x_a$ for $i = 1,\ldots,M$

with $\Delta x = (x_b-x_a)/(M-2)$. Then we have

$$\phi_{1,j} = \phi_{M-1,j} \quad \text{and} \quad \phi_{M,j} = \phi_{2,j} \quad \text{for } j = 1,\ldots,N.$$

Setting ICY = 1 with A21 = A2M = 0.0 will give the boundary conditions

X(1,j) = X(M-1,j)

X(M,j) = X(2,j), for j = 1,...,N

as required in Eq. (2.2).

4. SOLUTION METHOD

The method is based on the error propagation method (EVP) or sweep out method (Hirota et al., 1970). In the EVP method, a forward sweep provides a particular solution, which is followed by a further forward sweep of the solution error and the correction of the solution. We demonstrate the method for the case of Dirichlet boundary conditions.

At the interior points, we start with an initial guess solution, which may be a constant value equal to the average of the boundary values, for example. Rearranging the difference equation (2.1) we can write

$$X(i,j+1) = [F(i,j) - AX(i,j)*X(i-1,j) - AY(i,j)*X(i,j-1) - BB(i,j)*X(i,j) - CX(i,j)*X(i+1,j)] / CY(i,j)$$

for $i = 2,\ldots,M-1$ and $j = 2,\ldots,N-1$. (4.1)

A particular solution $X_p(i,j)$ can be found by starting with the initial guess on the second row $j = 2$ and marching Eq. (4.1) forward row by row to the end boundary $j = N$. The particular solution $X_p(i,j)$ then satisfies the boundary conditions on three sides $j = 1$, $i = 1$, $i = M$ but differs from the solution at the end boundary $j = N$ by an error row vector defined by

$$\epsilon_N(i) = X(i,N) - X_p(i,N)$$
$$= X(i,N) - [F(i,N-1) - AX(i,N-1)*X_p(i-1,N-1) - AY(i,N-1)*X_p(i,N-2) - BB(i,N-1)*X_p(i,N-1) - CX(i,N-1)*X_p(i+1,N-1)] / CY(i,N-1)$$

for $i = 2,\ldots,M-1$. (4.2)

A solution error $X_h(i,j) = X(i,j) - X_p(i,j)$ can be defined which satisfies the homogeneous difference equation (Eq. (2.1) with $F = 0$) with the homogeneous boundary conditions on the three sides

$$X_h(i,1) = 0, \text{ for } i = 1,\ldots,M$$
$$X_h(1,j) = X_h(M,j) = 0 \text{ for } j = 1,\ldots,N \quad (4.3)$$

and a non-zero boundary value given by the solution error

$$X_h(i,N) = \epsilon_N(i) \text{ for } i = 2,\ldots,M-1 \quad (4.4)$$

at the end boundary $j = N$. We can compute an influence matrix which relates the solution error on the last interior row to that on the second row. The solution error can then be obtained for the whole domain by marching forward from the second row and the particular solution corrected to give the solution.

To obtain the influence matrix we start with a unit row vector e_k defined on the second row. The unit row vector e_k has the k^{th} element unity and the

remaining elements zero. Marching the unit row vector forward, using the homogeneous difference equation, we obtain influence row vectors t_k on the last row j = N, defining the k^{th} row of our influence matrix **T**. Repeating the marching sequence for all the M-1 unit row vectors defines a set of M-1 influence row vectors, completing the influence matrix **T**. Using a linear combination of influence vectors and the principle of linear superposition, we can relate our error row vector $\epsilon_N(i)$ (for the last row j = N) to the error vector $\epsilon_2(i)$ (= $X_h(i,2)$) on the first interior row j = 2. That is, in matrix form using the influence matrix, we have

$$\epsilon_N = \epsilon_2 \, \mathbf{T} \tag{4.5}$$

Therefore given the computed particular solution $X_p(i,j)$ and the error vector ϵ_N on the last row, the error vector can be computed on the first interior row,

$$\epsilon_2 = \epsilon_N \, \mathbf{T}^{-1}. \tag{4.6}$$

Then marching the solution error forward provides the solution error throughout the whole domain, and, when combined with the particular solution, completes the solution.

The marching of the solution error can alternatively be conceptualized as follows. The error vector ϵ_N at the end boundary j = N can be related to a residual vector ρ defined on the last interior row j=N-1. Rearranging Eq. (4.2), we have

$$\begin{aligned} &AX(i,N-1)*X_p(i-1,N-1) + AY(i,N-1)*X_p(i,N-2) + BB(i,N-1)*X_p(i,N-1) \\ &+ CX(i,N-1)*X_p(i+1,N-1) + CY(i,N-1)*X(i,N) \\ &\qquad = F(i,N-1) + CY(i,N-1)*\epsilon_N(i) \end{aligned} \tag{4.7}$$

where we define the residual vector ρ on the last interior row j = N-1 by

$$\rho(i) = CY(i,N-1)*\epsilon_N(i), \quad \text{for } i = 2,\ldots,M-1. \tag{4.8}$$

The solution error can now be thought of satisfying the homogeneous boundary conditions on all sides and the homogeneous difference equation at all rows except at the last interior row. At the last row j=N-1, the solution satisfies the difference equation with a forcing function given by the residual ρ. By similarly marching the unit vectors forward, a set of residuals can be computed on the last interior row, defining a residual matrix R (Hirota, et al, 1970). The residual matrix then relates the solution error on the second row to the residual ρ defined on the last interior row.

$$\rho = \epsilon_2 \, \mathbf{R} \tag{4.9}$$

The two equivalent views of the solution error are shown schematically below in Fig. 1.

Due to round-off errors in marching the particular solution forward and due to inaccuracies in the computed inverse T^{-1}, the error vector ϵ_2 computed on the second row using Eq. (4.6) is an approximation to the true error vector. Marching the error vector forward then and correcting the particular solution provides an approximate solution. Provided that the error vector is not too inaccurate, an improved solution can be found by recomputing a new error vector on the last row and repeating the above procedure. A number of iterations of the procedure are usually required to converge to an accurate solution.

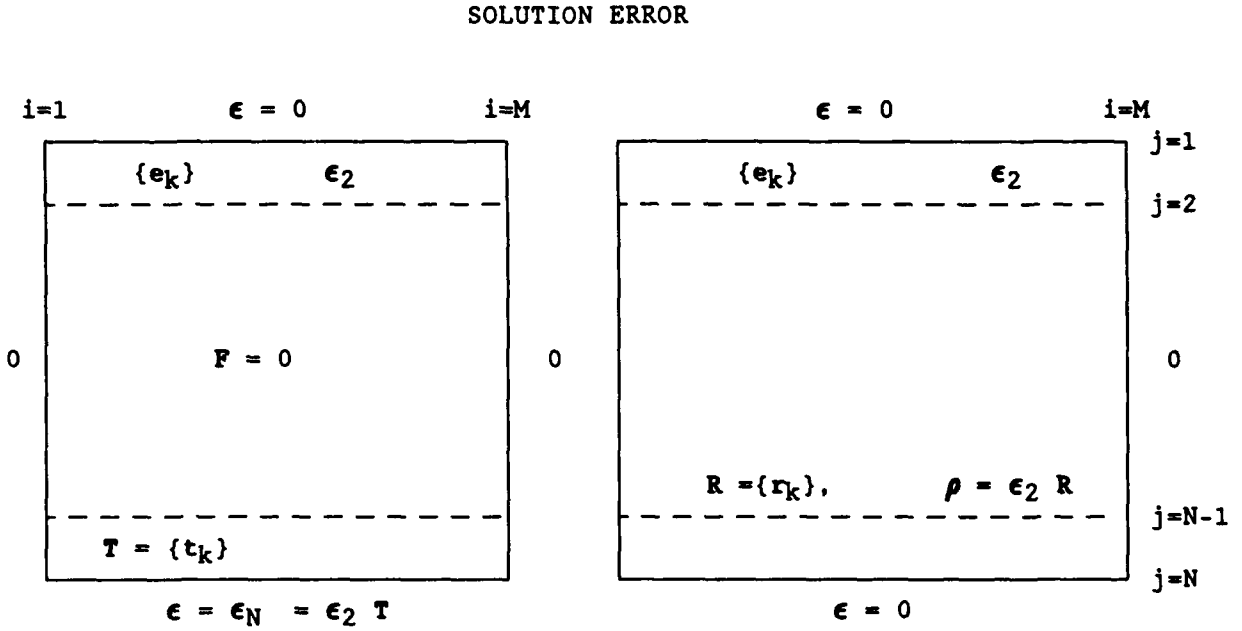


Fig. 1. The two equivalent views of the boundary conditions and forcing for the solution error. (a) Solution error satisfies homogeneous difference equation with a non zero boundary condition at j=N given by the error vector ϵ_N. Influence matrix T found by marching set of unit vectors $\{e_k\}$ to last row j = N. (b) Solution error satisfies homogeneous boundary conditions with a non-zero line forcing function given by the residual ρ at j=N-1. Residual matrix R found by marching unit vectors $\{e_k\}$ to last interior row j = N-1 and computing residuals r_k on that row.

As pointed out by McAvaney and Leslie (1972), the residual matrix (or equivalently the influence matrix) is ill-conditioned, with the degree of ill-conditioning increasing rapidly with increasing number of points N in the marching direction. For large N, the inverse of the influence matrix may not be computed very accurately and for moderate round-off error any accuracy in the computed error vector may be lost. The iterative procedure then will not converge to a solution. The accuracy of the inverse therefore limits the number of points that can be effectively used in the marching and correction process. On the Cray computer with 64 bit words, this limit is about 20 points. The degree of ill-conditioning of the influence matrix, which leads to inaccuracies in the inverse, can be measured by a condition number defined by

$$\text{cond}(\mathbf{T}) = ||\mathbf{T}||\ ||\mathbf{T}^{-1}|| \tag{4.10}$$

where $||\mathbf{T}||$ and $||\mathbf{T}^{-1}||$ are matrix norms of the influence matrix and it's inverse, respectively (see Dahlquist and Bjorck, 1974; for example). Then given an error $||\delta\epsilon_N||$ in the error vector on the last row, an upper bound on the relative error of the computed error vector on the second row can shown to be

$$\frac{||\delta\epsilon_2||}{||\epsilon_2||} \leq \text{cond}(\mathbf{T})\ \frac{||\delta\epsilon_N||}{||\epsilon_N||} \tag{4.11}$$

where the matrix and vector norms are defined in a consistent manner (see Dahlquist and Bjorck). For example, taking the case of a domain with 14 points in the marching j direction and 51 points in i direction the condition number is computed to be of order 10^8. Then given an extreme case of a high relative round-off error of 10^{-12} in the computed error vector ϵ_N on the last row, when using single precision floating point numbers on the 64 bit Cray computer, we expect the computed error vector on the second row to have a precision of at least four significant figures (for a relative error of 10^{-4}). The iterative procedure will then converge to a solution.

To stabilize the EVP method for a large number of grid points, Madala (1978) divided the domain into a series of blocks in the marching direction. Adjacent blocks overlap in the marching direction so that the first and last two rows of adjacent blocks are common. To provide for a rapid convergence

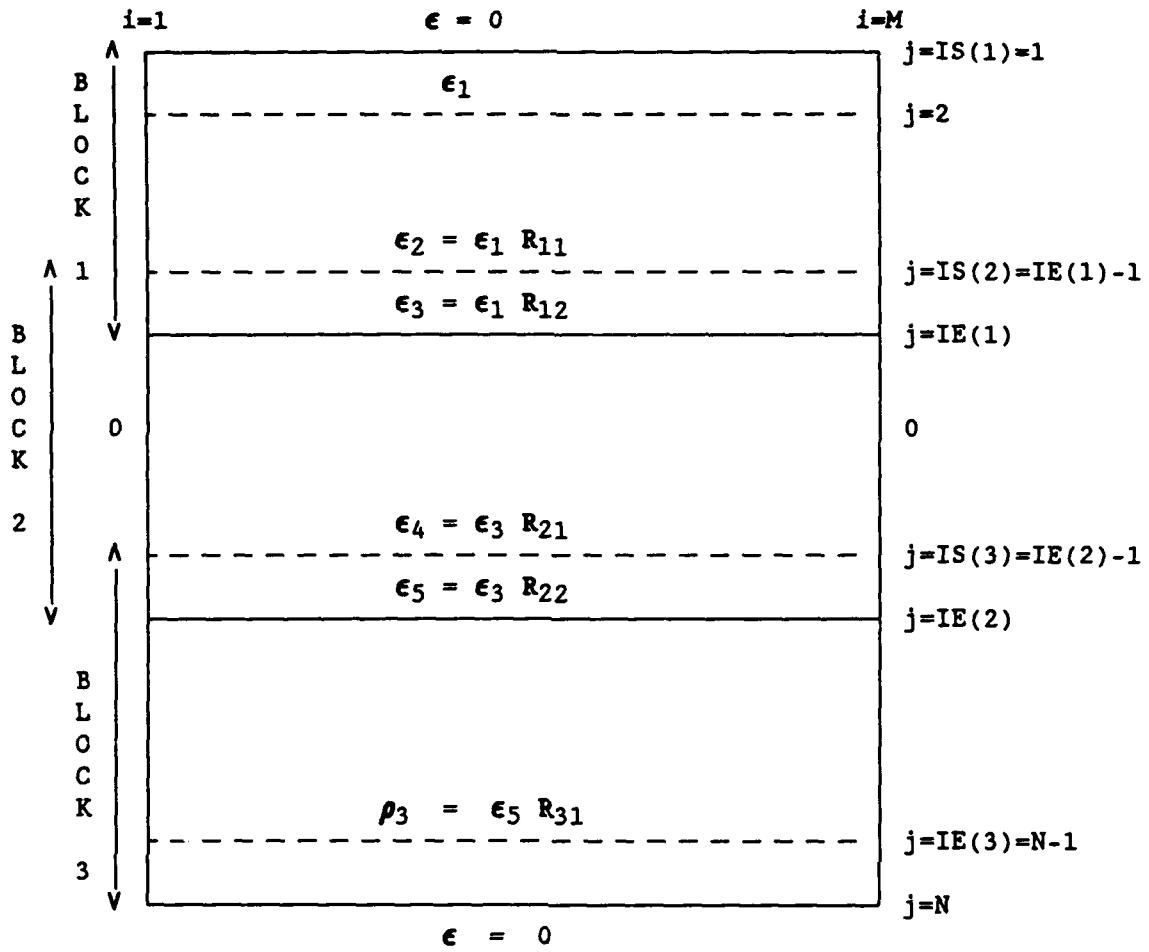


Fig. 2. The relationship between the error vectors ϵ on various rows at the block boundaries and the residual error ρ_3 on the next to the last row j=N-1, for the case of three blocks. R_{11}, R_{12}, R_{21}, R_{22} are influence matrices and R_{31} is a residual matrix, defined on the rows shown.

of the iterative procedure, the number of points in each block is limited to less than 15 points.

In the preprocessor step, NBLK x 2 - 1 matrices are computed which relate the solution error on various boundary rows of each block. In Fig. 2, showing three blocks for example, the influence matrices R_{12}, R_{11} for the first block relate the error vector ϵ_1 on the second row to the error vector ϵ_3 on the last row and to the error vector ϵ_2 on the next to the last row, respectively. Similar influence matrices R_{22}, R_{21} are computed for the second block. For the

last block a residual matrix R_{31} relates the residual solution error ρ_3 on the last interior row j=N-1 to the error vector ϵ_5 on the second row of the last block.

In the forward sweep a particular solution is found by the marching and correction process above, satisfying the boundary conditions and the initial guess field on the interior block boundaries. The particular solution satisfies the forcing function at all points except on the last interior row, where the difference equation satisfies the forcing plus a residual ρ_3. In the backward sweep a solution error can be computed on the block boundaries and the first interior row from the residual ρ_3 on the last interior row, using the residual and influence matrices defined for the blocks. By marching the solution error forward in each block the particular solution is then corrected to give the solution. The details of the stabilized error vector propagation method are described in the next chapter.

5. DESCRIPTION OF SEVP SUBROUTINES

5.1 Initial Set-up, SEVP

In the main subroutine SEVP, the block boundaries are first computed for the prescribed number of blocks and the difference equations are modified to accommodate boundary conditions which are not Dirichlet.

Fig. 2 shows the relationship of the block boundaries for the case of three overlapping blocks. The beginning and end rows of the blocks are defined by the one-dimensional arrays IS and IE, respectively. The integer array NBSIZ2, defining the number of interior rows between the end boundaries of the blocks, is computed for the prescribed number of blocks in subroutine BLKSIZ. The first block contains a total of NBSIZ2(1) + 2 rows, while the remaining blocks contain NBSIZ2() + 3 rows each. For the special case of Neumann boundary conditions at boundary j = N (i.e. A1N = 1), the last block (third block in Fig. 2) is set with NBSIZ2() = 4 by BLKSIZ (for a total of 7 rows for the block), to provide for a more efficient convergence of the solution.

For the case of cyclic boundary conditions we define an i index INX(i) by

$$\begin{aligned} &INX(i) = i \text{ for } i=2,\ldots,M-1 \\ &INX(1) = M-1 \\ &INX(M) = 2 \end{aligned} \tag{5.1}$$

The difference equation (2.1) is then modified

$$\begin{aligned} &AX(i,j)*X(INX(i-1),j) + AY(i,j)*X(i,j-1) + BB(i,j)*X(i,j) \\ &+ CX(i,j)*X(INX(i+1),j) + CY(i,j)*X(i,j+1) = F(i,j) \\ &\text{for } i = 2,\ldots,M-1 \text{ and } j = 2,\ldots,N-1. \end{aligned} \tag{5.2}$$

to provide for the cyclic boundary conditions at the boundaries i = 1 and i = M,

$$\begin{aligned} &X(1,j) = X(M-1,j) \\ &X(M,j) = X(2,j), \text{ for } j = 2,\ldots,N-1. \end{aligned} \tag{5.3}$$

In the case of Neumann boundary conditions at the x boundaries i = 1 and i = M, our boundary conditions are

$$\begin{aligned} &X(1,j) = X(2,j) - F21(j) \text{ and} \\ &X(M,j) = X(M-1,j) + F2M(j) \text{ for } j = 2,\ldots,N-1. \end{aligned} \tag{5.4}$$

Now the dif' rence equation at i = 2 is

$$AX(2,j)*X(1,j) + AY(2,j)*X(2,j-1) + BB(2,j)*X(2,j) + CX(2,j)*X(3,j) + CY(2,j)*X(2,j+1) = F(2,j) \quad (5.5)$$

for j = 2,...,N-1.

Substituting our boundary condition Eq (5.4) for X(1,j) in Eq. (5.5) we have

$$AY(2,j)*X(2,j-1) + [BB(2,j) + AX(2,j)]*X(2,j) + CX(2,j)*X(3,j) + CY(2,j)*X(2,j+1) = [F(2,j) + AX(2,j)*F21(j)] \quad (5.6)$$

for j = 2,...,N-1.

By redefining our variables such that

$$\begin{aligned} X(1,j) &\rightarrow 0.0 \\ BB(2,j) &\rightarrow BB(2,j) + AX(2,j) \\ F(2,j) &\rightarrow F(2,j) + AX(2,j)*F21(j) \text{ for } j = 2,...,N-1 \end{aligned} \quad (5.7)$$

we can leave the form of the difference equation the same as for the case of Dirichlet boundary conditions. Similarly, by redefining

$$\begin{aligned} X(M,j) &\rightarrow 0.0 \\ BB(M-1,j) &\rightarrow BB(M-1,j) + CX(M-1,j) \\ F(M-1,j) &\rightarrow F(M-1,j) - CX(M-1,j)*F2M(j) \text{ for } j = 2,...,N-1 \end{aligned} \quad (5.8)$$

the form of the difference equation is maintained for i=M-1. We have essentially folded our boundary conditions into the difference equation, ostensibly replacing Neumann boundary conditions with Dirichlet boundary conditions with zero boundary values X(1,j) = X(M,j) = 0.0 at the boundaries i = 1 and i = M, respectively. Similar modifications are made for Neumann boundary conditions at the y boundaries, j = 1 and j = N.

In the case of 4-sided Neumann boundary conditions or 2-sided Neumann with cyclic boundary conditions in x, the solution is indeterminate. For if ϕ is a solution, then so is $\phi + \kappa$, where κ is an arbitrary constant. For the solver to obtain a solution, a boundary value must be prescribed at some chosen boundary point (IX, JX). For the most efficient convergence of the solution, it is recommended that a zero value be used in this case for the initial guess and for the constant value at the prescribed boundary point. For the nearest interior row j = JF to the prescribed boundary point, the boundary point X(IX,JX) and the coefficients BB(IX,JF) and the forcing F(IX,JF) in the difference equation are not modified by the SEVP subroutine. The boundary point initially chosen in SEVP is IX = 2, JX = 1, with the nearest interior row being JF = 2.

For convenience, we further modify the difference equation for j=N-1 to include the Dirichlet boundary condition at j=N in the forcing function. In this case the value X(i,N) of the solution variable is prescribed on the boundary j=N. The difference equation for j=N-1 is

```
  AX(i,N-1)*X(i-1,N-1) + AY(i,N-1)*X(i,N-2) + BB(i,N-1)*X(i,N-1)
 + CX(i,N-1)*X(i+1,N-1) + CY(i,N-1)*X(i,N)  =  F(i,N-1)                    (5.9)
    for   i = 2,...,M-1,
```

which we rearrange as

```
  AX(i,N-1)*X(i-1,N-1) + AY(i,N-1)*X(i,N-2) + BB(i,N-1)*X(i,N-1)
 + CX(i,N-1)*X(i+1,N-1)  =  F(i,N-1)  -  CY(i,N-1)*X(i,N)                  (5.10)
    for   i = 2,...,M-1.
```

By redefining our forcing function at j=N-1 such that

```
 F(i,N-1)  ⟶  F(i,N-1) - CY(i,N-1)*X(i,N) for i = 2,...,M-1                (5.11)
```

we can assume for convenience that X(i,N) = 0.0 at the end boundary j = N.

5.2 The Preprocessor, BSM1

In the preprocessor, subroutine BSM1, NBLK x 2 - 1 matrices are computed which relate the solution error on various boundary rows of each block and the residual error on the last interior row j = N-1. The solution error satisfies homogeneous boundary conditions on all sides, and the homogeneous difference equation at all interior points except the last interior row j = N-1. On the last interior row the solution error is forced by a residual.

To obtain the influence matrices for the first block, the unit vector e_k (with the k^{th} element one) defined on the second row j = 2, is marched forward with homogeneous boundary conditions to compute row vectors on the last two rows of the block, j = IE(1)-1, j = IE(1). These define the k^{th} rows for two influence matrices (shown as R_{11} and R_{12} in Fig. 2) for the last two rows. Repeating the marching procedure for the other unit vectors completes the influence matrices R_{11} and R_{12}, which then relate the solution error vector ϵ_1 on the second row to the error vectors ϵ_2, ϵ_3 on the last two rows, respectively, so that

$$\epsilon_2 = \epsilon_1 R_{11} \tag{5.12}$$

and

$$\epsilon_3 = \epsilon_1 R_{12} \tag{5.13}$$

By matrix multiplication of the inverse of R_{11} times R_{12} a further influence matrix S_1 can be obtained, which relates the solution error vector on the next to the last and the last rows of the first block. For by the elimination of ϵ_1 in Eqs. (5.12) and (5.13), we have

$$\epsilon_3 = \epsilon_2 R_{11}^{-1} R_{12} = \epsilon_2 S_1. \tag{5.14}$$

The inverse of the influence matrix S_1 is computed using $S_1^{-1} = R_{12}^{-1} R_{11}$.

For the second block (and any other subsequent interior blocks), we again start with a unit vector defined on the second row of the block, but the vector on the first row is now defined by the k^{th} column of the inverse of the influence matrix S_1 defined for the previous block. For if e_k is the unit vector, then the vector on the first row is given by $e_k S_1^{-1}$. Marching to the last row for each unit row vector, then defines the influence matrices R_{21} and R_{22} for the last two rows of the block. If error vectors ϵ_3, ϵ_4 and ϵ_5 represent the solution error on the second and last two rows of the block, respectively, then

$$\epsilon_4 = \epsilon_3 R_{21} \tag{5.15}$$

and

$$\epsilon_5 = \epsilon_3 R_{22} \tag{5.16}$$

An influence matrix S_2 relating the solution error on the last two rows of the block is again defined so that

$$\epsilon_5 = \epsilon_4 S_2 \tag{5.17}$$

where $S_2 = R_{21}^{-1} R_{22}$, and its inverse also computed using $S_2^{-1} = R_{22}^{-1} R_{21}$.

For the last block (third block in Fig. 2), the unit vectors are marched to the next to last row and residuals computed, defining the rows of a residual matrix, R_{31}. The residual matrix then relates the residual solution error ρ on the last interior row j = N-1 to the solution error ϵ_5 on the second row j = IE(2) of the last block, so that

$$\rho_3 = \epsilon_5 R_{31} \tag{5.18}$$

5.3 The Solution, BSM2, BSM3

In the forward sweep an approximate solution is found for each block, satisfying the boundary conditions and an initial guess field on the last row of each block. This particular solution satisfies the forcing function at all points except on the last interior row, where the difference equation satisfies the forcing plus a residual.

For the first block, we want to obtain the particular solution which satisfies an assigned initial guess value on the interior block boundary j = IE(1). The boundary values and the initial guess on the second row are used to march forward to the block boundary and the error ϵ^{P}_{3} from the assigned guess value on the block boundary is computed. The required correction on the second row is then given by $\epsilon^{P}_{1} = \epsilon^{P}_{3} R_{12}^{-1}$, applying Eq. (5.13). By marching the error forward, the corrected particular solution is obtained. The process is repeated for several iterations to reduce the round off error.

For subsequent interior blocks, the particular solution on the first two rows of the block, which has been computed in the previous block, is similarly marched forward and corrected for the assigned initial guess value on the last row of this block. For the last block the particular solution from the previous block is marched to the last interior row j = N-1 and a residual error ρ_3 computed.

In the backward sweep, error vectors are computed on the block boundaries and the first interior row given the residual error on the last interior row, by using the residual and influence matrices defined for the blocks. The solution error can then be marched forward in each block and the particular solution corrected to give the solution.

For the case of three blocks, as shown in Fig. 3, we start the backward sweep with the residual error ρ_3 given on the last interior row j = N-1. The error vectors ϵ_4, ϵ_5 on the first two rows of the last block can then be computed using the residual matrix R_{31} and the influence matrix S_2, using

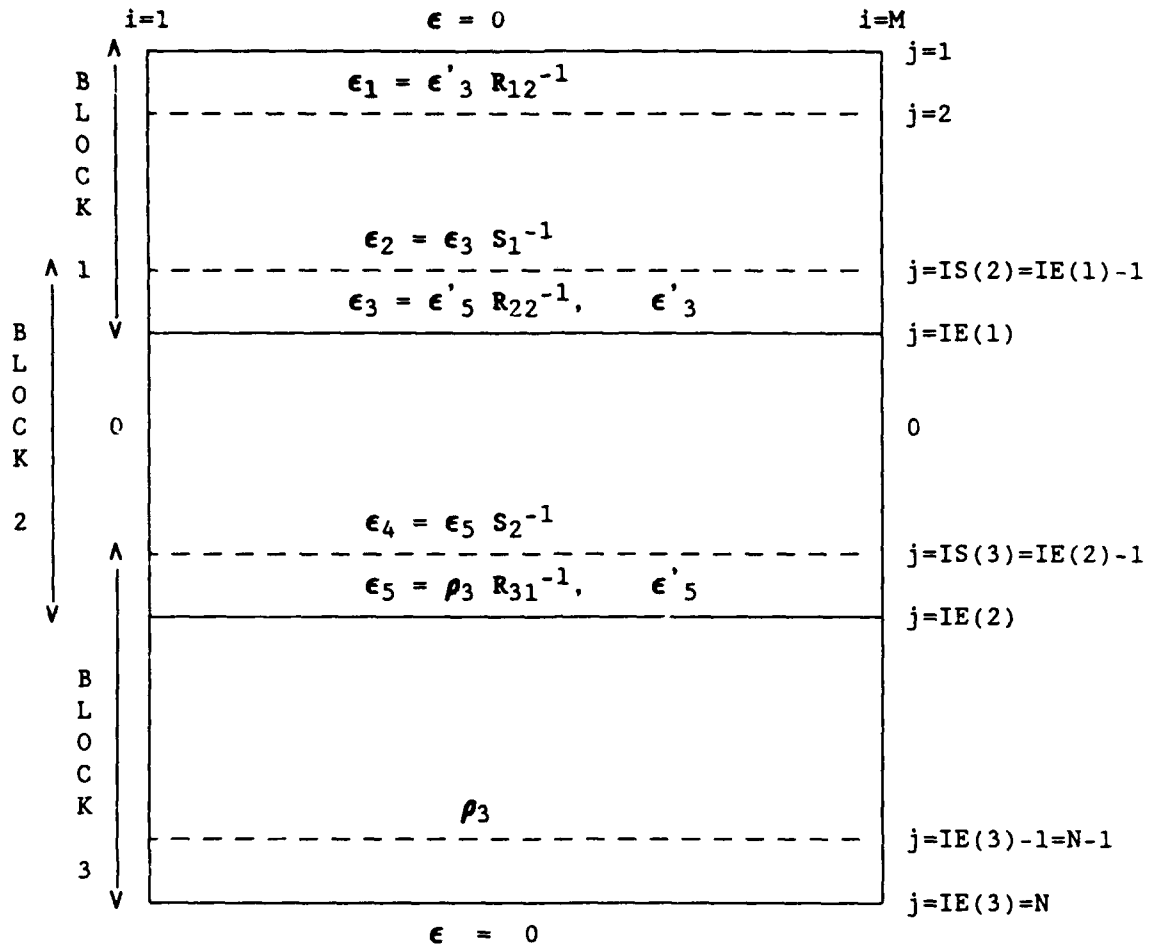


Fig. 3. The solution errors computed in the backward sweep of the blocks. ρ_3 is the residual error on the row j=N-1 and ϵ represents the error vectors on the rows shown.

$$\epsilon_5 = \rho_3 \, R_{31}^{-1} \tag{5.19}$$

$$\epsilon_4 = \epsilon_5 \, S_2^{-1} \tag{5.20}$$

A forward marching of the solution error then corrects the solution in the last block. As before, several iterations are used to reduce round off error.

In the backward sweep of the remaining blocks, the particular solution is recomputed on the last row of each block and the solution error recomputed on the block boundary. For the second block in Fig. 3 for example, the error vector ϵ'_5 is recomputed by backing up three rows into the second block and

repeating a small segment of the forward sweep. The solution error is then obtained on the first two rows of the block using Eqs. (5.16) and (5.14),

$$\epsilon_3 = \epsilon'_5 \, R_{22}^{-1} \tag{5.21}$$

$$\epsilon_2 = \epsilon_3 \, S_1^{-1} \tag{5.22}$$

and the solution corrected throughout the block as above. For the first block the recomputed error vector ϵ'_3 on the last row provides the error vector ϵ_1 on the second row of the block by using Eq (5.13)

$$\epsilon_1 = \epsilon'_3 \, R_{12}^{-1} \tag{5.23}$$

and the solution corrected throughout the block.

With the correction of the solution in the first block the solution is complete. After the forward and backward sweep, the remaining residual error accumulates on the interior boundaries of the blocks. If necessary, the forward and backward sweep of the blocks is repeated for several iterations to reduce any remaining error.

6. DESCRIPTION OF TEST EXAMPLE

As an example we solve Poisson's equation $\nabla^2 \phi = f$ in spherical coordinates in a rectangular domain on the surface of a spherical earth. We define transformed coordinates x and y by

$$x = r\,\lambda \quad \text{and} \quad y = r\,\theta$$

where r = 6371 km is the average radius of the earth, θ and λ are the latitude and longitude, respectively in radians. In our coordinates, Poisson's equation is written

$$\frac{1}{h_x h_y}\left\{\frac{\partial}{\partial x}\left[\frac{h_y}{h_x}\frac{\partial \phi}{\partial x}\right] + \frac{\partial}{\partial y}\left[\frac{h_x}{h_y}\frac{\partial \phi}{\partial y}\right]\right\} = f(x,y), \qquad \begin{matrix} x_a < x < x_b \\ y_a < y < y_b \end{matrix} \tag{6.1}$$

where for our spherical coordinates the map factors h_x and h_y are

$$h_x = \cos\vartheta = \cos(y/r), \text{ and } h_y = 1.$$

For our rectangular domain we choose

$$x_a = 0 \text{ km}, \; x_b = 12{,}000 \text{ km}, \; y_a = 2{,}000 \text{ km and } y_b = 7{,}000 \text{ km}.$$

We want to force a wave solution on a zonal background field. An example of such a solution is given by

$$\phi(x,y) = 5.5 - 0.5 \tanh(y_p) + 0.5 \exp(-y_p^2) \cos\frac{2\pi x}{L} \tag{6.2}$$

where $$y_p = \frac{y - y_c}{y_w}$$

and $L = x_b - x_a = 12{,}000$ km, $y_c = 4{,}500$ km, $y_w = 1{,}500$ km. By applying Eq. (6.1). the appropriate forcing function for this solution is then

$$f(x,y) = \frac{1}{2\,y_w^2}\left[\,2\tanh(y_p) + \frac{y_w}{r}\tan\left(\frac{y}{r}\right)\right]\operatorname{sech}^2(y_p) +$$

$$\frac{1}{y_w^2}\left[\,2\,y_p^2 + \frac{y_w}{r}\tan\left(\frac{y}{r}\right)y_p - c\,\right]\exp(-y_p^2)\cos\left(\frac{2\pi x}{L}\right) \tag{6.3}$$

where $$c = 1 + \frac{2\pi^2 y_w^2}{L^2 \cos^2\left(\frac{y}{r}\right)}$$

6.1 Dirichlet Boundary Conditions in y

Our boundary conditions are taken as periodic in x such that

$$\phi(x+x_b,y) = \phi(x+x_a,y)$$

and in this first case we treat Dirichlet boundary conditions at the y boundaries, specifying the values of ϕ at $y = y_a$ and $y = y_b$. That is

$$\begin{aligned} \phi(x,y_a) &= 5.9655 + 0.0311 \cos \frac{2\pi x}{L} \\ \phi(x,y_b) &= 5.0344 + 0.0311 \cos \frac{2\pi x}{L} \end{aligned} \tag{6.4}$$

We define a grid of M = 82 by N = 51 points (x_i, y_j) adding an extra column of points at $x=x_b+\Delta x$ for the cyclic boundary conditions. The grid is then

$$\begin{aligned} x_i &= (i-1)\,\Delta x + x_a \quad \text{for } i = 1,\ldots,M \\ y_j &= (j-1)\,\Delta y + y_a \quad \text{for } j = 1,\ldots,N \end{aligned} \tag{6.5}$$

with $\Delta x = (x_b-x_a)/(M-2) = 150$ km and $\Delta y = (y_b-y_a)/(N-1) = 100$ km

Our Poisson's equation (6.1) can now be written in finite difference form as

$$\frac{1}{h_{xj}\,h_{yj}\Delta x}\left\{\frac{h_{yj}\left[\phi_{i+1,j}-\phi_{i,j}\right]}{h_{xj}\,\Delta x} - \frac{h_{yj}\left[\phi_{i,j}-\phi_{i-1,j}\right]}{h_{xj}\,\Delta x}\right\} +$$

$$\frac{1}{h_{xj}\,h_{yj}\Delta y}\left\{\frac{h_{xj+\frac{1}{2}}\left[\phi_{i,j+1}-\phi_{i,j}\right]}{h_{yj+\frac{1}{2}}\,\Delta y} - \frac{h_{xj-\frac{1}{2}}\left[\phi_{i,j}-\phi_{i,j-1}\right]}{h_{yj-\frac{1}{2}}\,\Delta y}\right\}$$

$$= f_{i,j} \tag{6.6}$$

where $h_{xj+\frac{1}{2}} = (h_{xj+1} + h_{xj})/2$ and with the boundary conditions (6.4)

$$\phi_{i,1} = 5.9655 + 0.0311 \cos \frac{2\pi x_i}{L}$$

$$\phi_{i,N} = 5.0344 + 0.0311 \cos \frac{2\pi x_i}{L} \qquad (6.7)$$

and $\phi_{1,j} = \phi_{M-1,j}$

$\phi_{M,j} = \phi_{2,j}$

The difference equation (6.6) can further be written as

$$\left[\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j} \right] + \frac{h_{xj} \Delta x^2}{h_{yj} \Delta y^2} \left(\frac{h_{xj+\frac{1}{2}} \left[\phi_{i,j+1} - \phi_{i,j} \right]}{h_{yj+\frac{1}{2}}} - \frac{h_{xj-\frac{1}{2}} \left[\phi_{i,j} - \phi_{i,j-1} \right]}{h_{yj-\frac{1}{2}}} \right)$$

$$= h_{xj}^2 \Delta x^2 f_{i,j} \qquad (6.7)$$

Setting PHI(i,j) = $\phi_{i,j}$ and defining the coefficients in the form for the difference equation (2.1) we have for i = 2,...,M-1 and j = 2,...,N-1

$$AX(i,j) = 1.$$

$$CX(i,j) = 1.$$

$$AY(i,j) = \frac{h_{xj} \left(h_{xj} + h_{xj-1} \right) \Delta x^2}{h_{yj} \left(h_{yj} + h_{yj-1} \right) \Delta y^2}$$

$$CY(i,j) = \frac{h_{xj} \left(h_{xj+1} + h_{xj} \right) \Delta x^2}{h_{yj} \left(h_{yj+1} + h_{yj} \right) \Delta y^2} \qquad (6.8)$$

$$BB(i,j) = -2 - AY(i,j) - CY(i,j)$$

$$F(i,j) = h_{xj}^2 \Delta x^2 f_{i,j}$$

For our boundary conditions we have for i = 1,...,M

$$\text{PHI}(i,1) = 5.9655 + 0.0311 \cos \frac{2\pi x_i}{L} \quad \text{with} \quad \text{A11} = 0.0$$
$$\text{PHI}(i,N) = 5.0344 + 0.0311 \cos \frac{2\pi x_i}{L} \quad \text{with} \quad \text{A1N} = 0.0 \tag{6.9}$$

and ICY = 0, A21 = A2M = 0.0 giving

$$\text{PHI}(1,j) = \text{PHI}(M-1,j) \quad \text{and} \quad \text{PHI}(M,j) = \text{PHI}(2,j) \tag{6.10}$$

As a first guess in the interior of the domain, we initially set PHI to the average of the boundary values at $y = y_a$ and $y = y_b$, giving

$$\text{PHI}(i,j) = 5.5 \quad \text{for} \quad i = 2,\ldots,M-1 \text{ and } j = 2,\ldots,N-1. \tag{6.11}$$

With a call to SEVP, the solution and the residual error are computed. The residual error gives the difference between the forcing and the discrete Laplacian of the computed solution. The maximum of the absolute values of the residual errors, divided by the maximum forcing is then computed and printed. A normalized discretization error is also computed by taking the maximum value of the absolute differences between the exact analytic solution Eq. (6.2) and the computed solution and dividing by the maximum value of the exact solution. With the Cray computer, the following values are printed:

Maximum residual error (normalized by maximum forcing) = 5.9805×10^{-11}
Maximum discretization error (normalized by maximum value of exact solution) = 8.3005×10^{-5}.

6.2 Neumann Boundary Conditions in y

Our boundary conditions are again taken as periodic in x such that

$$\phi(x+x_b,y) = \phi(x+x_a,y)$$

but in this case we treat Neumann boundary conditions at the y boundaries, specifying the values of the normal gradient of ϕ at $y = y_a$ and $y = y_b$. From our definition of ϕ the y-gradient of ϕ at the boundaries $y = y_a$ and $y = y_b$ is

$$\frac{\partial \phi}{\partial y}(x,y_a) = \frac{-1}{2\,y_w}\,\mathrm{sech}^2 y_{pa} - \frac{y_{pa}}{y_w}\,\exp(-y_{pa}^2)\cos\frac{2\pi x}{L}$$

$$\frac{\partial \phi}{\partial y}(x,y_b) = \frac{-1}{2\,y_w}\,\mathrm{sech}^2 y_{pb} - \frac{y_{pb}}{y_w}\,\exp(-y_{pb}^2)\cos\frac{2\pi x}{L} \qquad (6.12)$$

$$\text{where } y_{pa} = \frac{y_a - y_c}{y_w} \quad \text{and} \quad y_{pb} = \frac{y_b - y_c}{y_w}$$

We define a grid of M = 82 by N = 53 points (x_i, y_j) adding an extra column of points at $x=x_b+\Delta x$ for the cyclic boundary conditions and two rows of points at $y = y_a - \Delta y/2$ and $y = y_b + \Delta y/2$ for the Neumann boundary conditions. The grid is then

$$x_i = (i-1)\,\Delta x + x_a \quad \text{for } i = 1,\ldots,M$$
$$y_j = (j-1)\,\Delta y + y_a - \Delta y/2 \quad \text{for } j = 1,\ldots,N \qquad (6.13)$$

with $\Delta x = (x_b-x_a)/(M-2) = 150$ km and $\Delta y = (y_b-y_a)/(N-2) = 100$ km. Setting PHI(i,j) = $\phi_{i,j}$, the coefficients for the difference equation are defined by Eq. (6.8) as in the case of Dirichlet boundary conditions, but with the map factors computed for the new grid. For our boundary conditions we define

$$\mathrm{F11}(i) = \left[\frac{-1}{2\,y_w}\,\mathrm{sech}^2 y_{pa} - \frac{y_{pa}}{y_w}\,\exp(-y_{pa}^2)\cos\frac{2\pi x_i}{L}\right]\Delta y$$

$$\mathrm{F1N}(i) = \left[\frac{-1}{2\,y_w}\,\mathrm{sech}^2 y_{pb} - \frac{y_{pb}}{y_w}\,\exp(-y_{pb}^2)\cos\frac{2\pi x_i}{L}\right]\Delta y \qquad (6.14)$$

with A11 = 1.0 and A1N = 1.0

and ICY = 0, A21 = A2M = 0.0 giving

$$\mathrm{PHI}(1,j) = \mathrm{PHI}(M-1,j) \quad \text{and} \quad \mathrm{PHI}(M,j) = \mathrm{PHI}(2,j) \qquad (6.15)$$

With the boundary conditions as given, the solution is indeterminate. To obtain a unique solution, we require a value for the solution at a selected point chosen on the boundary (preset as i = 2, j = 1 in the SEVP solver). For the most efficient convergence of the solver we use a zero value for PHI at this point and as the first guess, defining

PHI(i,j) = 0.0 for i = 1,...,M and j = 1,...,N. (6.16)

With a call to SEVP, the solution and the residual error are computed. The residual error gives the difference between the forcing and the discrete Laplacian of the computed solution. The maximum of the absolute values of the residual errors, divided by the maximum forcing is then computed and printed. A normalized discretization error is also computed by taking the maximum value of the absolute differences between the exact analytic solution and the computed solution and dividing by the maximum value of the exact solution. With the Cray computer, the following values are printed:

Maximum residual error (normalized by maximum forcing) = 8.0661×10^{-11}
Maximum discretization error (normalized by maximum value of exact solution)
= 8.8219×10^{-5}.

7. REFERENCES

Dahlquist, G., and A. Bjorck, 1974: Numerical Methods. Translated by N. Anderson, Prentice-Hall, 175-177.

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.

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.

Madala, R.V., 1978: An efficient direct solver for separable and non-separable elliptic equations. Mon. Wea. Rev., 106, 1735-1741.

Madala, R.V., 1981: Solution of Elliptic Equations. Finite Difference Techniques for Vectorized Fluid Dynamic Calculations. D. Book, Ed., Chap. 7, Springer-Verlag, 117-135.

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.

APPENDIX: LISTING OF TEST PROGRAMS

1. Test Example with Dirichlet Boundary Condition

```
      PROGRAM DRIVR1
C
C     This program uses SEVP to solve Poisson's equation
C                LAPLACIAN OF PHI = F
C     with cyclic boundary conditions in x and Dirichlet
C     boundary conditions in y.
C
      PARAMETER (M=82, N=51)
      PARAMETER (NBLK=(N+3)/9, NBLK1=NBLK-1)
      PARAMETER (M1=M-1, M2=M-2, N1=N-1, N2=N-2)
      PARAMETER (MN=M*N)
      DIMENSION AX(M,N),AY(M,N),CX(M,N),CY(M,N),BB(M,N),
     1 RINV(M2,M2,NBLK),RINV1(M2,M2,NBLK1),RCOR(M,3),RTILDA(M2),
     2 NBSIZ2(NBLK),IS(NBLK),IE(NBLK),SUMF(NBLK),IXNDX(M),
     3 DUMMY(M2,M2),B1(M2),B2(M2)
      DIMENSION F(M,N),PHI(M,N),ERR(M,N)
      DIMENSION F11(M),F1N(M),F21(N),F2M(N)
      DIMENSION X(M),Y(N),YP1(N),YP2(N)
      DIMENSION HXU(N),HXV(N1),HYU(N),HYV(N1)
      DIMENSION AA(N),AC(N),CC(N)
      DIMENSION T1(N),T2(N),T3(N),T4(N)
      DIMENSION DUM3(M,N)
C SET COEFFICIENTS OF DIFFERENCE EQUATION AND BOUNDARY CONDITIONS TO
C ZERO INITIALLY
      DATA AX/MN*0.0/,CX/MN*0.0/,AY/MN*0.0/,CX/MN*0.0/,BB/MN*0.0/,
     1     F/MN*0.0/,F11/M*0.0/,F1N/M*0.0/,F21/N*0.0/,F2M/N*0.0/
C
      PI = 4.0*ATAN(1.0)
      AR = 6371.
      XL = 12000.0
      YL =  5000.0
C
C DEFINE GRID
      DELX = XL/FLOAT(M-2)
      DELY = YL/FLOAT(N-1)
      PRINT 990,DELX,DELY
  990 FORMAT(1H ,5HDELX=,1PE13.5,5X,5HDELY=,1PE13.5)
      DO 5 J=1,N
      Y(J)=DELY*FLOAT(J-1)+2000.
    5 CONTINUE
      DO 6 I=1,M
      X(I)=DELX*FLOAT(I-1)
    6 CONTINUE
      DELXSQ=DELX*DELX
      DELYSQ=DELY*DELY
C DEFINE MAP FACTORS
      DO 10 J=1,N
      HXU(J)=COS(Y(J)/AR)
      HYU(J)=1.0
   10 CONTINUE
```

```
      DO 20 J=1,N1
      HXV(J)=0.5*(HXU(J)+HXU(J+1))
      HYV(J)=1.0
   20 CONTINUE
C
C DEFINE FORCING FUNCTION
      XC=  0.0
      YC = 4500.0
      YW = 1500.0
      C1=YW/AR
      C2=1./(YW*YW)
      C3=2.*PI*PI*(YW/XL)*(YW/XL)
      DO 60 J=1,N
      YP1(J)=(Y(J)-YC)/YW
      YP2(J) = Y(J)/AR
   60 CONTINUE
      DO 61 J=1,N
      T1(J) = TANH(YP1(J))
      T2(J) = EXP(-YP1(J)*YP1(J))
   61 CONTINUE
      DO 62 J=1,N
      CSQ=COS(YP2(J))*COS(YP2(J))
      T3(J)=2.*T1(J)+C1*TAN(YP2(J))
      T4(J)= 2.*YP1(J)*YP1(J) - 1.0 + C1*YP1(J)*TAN(YP2(J))
     1    -  C3/CSQ
   62 CONTINUE
C  FORCING FUNCTION DEFINED AT INTERIOR POINTS OF DOMAIN ONLY
      DO 65 J=2,N-1
      DO 65 I=2,M-1
      F(I,J) = 0.5*C2*T3(J)*(1.-T1(J)*T1(J))
     1          +   C2*T4(J)*T2(J)*COS(2.*PI*(X(I)-XC)/XL)
   65 CONTINUE
C
C
C DIRICHLET BOUNDARY CONDITIONS IN Y
      A11 = 0.0
      A1N = 0.0
      DO 100 I=1,M
      PHI(I,1)=5.5-0.5*T1(1)+0.5*T2(1)*COS(2.*PI*(X(I)-XC)/XL)
      PHI(I,N)=5.5-0.5*T1(N)+0.5*T2(N)*COS(2.*PI*(X(I)-XC)/XL)
  100 CONTINUE
C
C  FOR DIRICHLET B.C. INITIALLY SET PHI IN INTERIOR TO AVERAGE
C     OF BOUNDARY VALUES
      AVPH = 0.0
      DO 110 I=2,M1
      AVPH = AVPH+PHI(I,1)+PHI(I,N)
  110 CONTINUE
      AVPH=AVPH/FLOAT(2*(M-2))
      DO 120 J=2,N1
      DO 120 I=2,M1
       PHI(I,J)= AVPH
  120 CONTINUE
C FOR CYCLIC BOUNDARY CONDITIONS
```

```
      ICYC = 1
      A21 = 0.0
      A2M = 0.0
      DO 130 J=1,N
       PHI(1,J) = PHI(M-1,J)
       PHI(M,J) = PHI(2,J)
  130 CONTINUE
C
C DEFINE COEFFICIENTS ON L.H.S. OF DIFFERENCE EQUATION.
C COEFFICIENTS DEFINED IN INTERIOR OF DOMAIN ONLY.
      DO 25  J=2,N-1
      AA(J)=((HXU(J)*HXV(J-1))/(HYV(J-1)*HYU(J)))*(DELXSQ/DELYSQ)
      CC(J)=((HXU(J)*HXV(J))/(HYV(J)*HYU(J)))*(DELXSQ/DELYSQ)
      AC(J)=-AA(J)-CC(J)
   25 CONTINUE
      DO 165 I = 2,M-1
      DO 165 J = 2,N-1
       AX(I,J) = 1.0
       CX(I,J) = 1.0
       AY(I,J) = AA(J)
       CY(I,J) = CC(J)
       BB(I,J) = AC(J)-2.0
  165 CONTINUE
C DEFINE R.H.S. OF DIFFERENCE EQUATION, SCALING FORCING FUNCTION
      DO 170 J=2,N-1
      DO 170 I=2,M-1
       CON = HXU(J)*HXU(J)*DELXSQ
       F(I,J) = F(I,J)*CON
  170 CONTINUE
      FMAX = 0.
      DO 305 J=2,N-1
      DO 305 I=2,M-1
       FMAX = AMAX1(FMAX,ABS(F(I,J)))
  305 CONTINUE
C
C
C TOLERANCE FOR SOLVER
      TOL = 1.0E-8
C
      IFLG = 1
      CALL SEVP(AX,AY,CX,CY,BB,RINV,RINV1,RCOR,NBSIZ2,IS,IE,IXNDX,
     1SUMF,DUMMY,RTILDA,PHI,F,ERR,F11,F1N,F21,F2M,M,N,M1,N1,M2,N2,
     2NBLK,NBLK1,B1,B2,A11,A1N,A21,A2M,ICYC,TOL,IFLG)
C
      PRINT 710
      PRINT 711, NBSIZ2
  710 FORMAT(1X,'BLOCK SIZE GIVEN BY NBSIZ2(1)+2 (1ST BLOCK) OR',
     1 ' NBSIZ2()+3 (OTHER BLOCKS)'/,1X,' NBSIZ2: ')
  711 FORMAT(1X,10I4)
C
C RESIDUAL ERROR OF LAPLACIAN
      ERMAX=0.0
      DO 750 I=2,M1
      DO 750 J=2,N1
```

```
      ERMAX = AMAX1(ERMAX,ABS(ERR(I,J)/FMAX))
  750 CONTINUE
      PRINT 751, ERMAX
  751 FORMAT(1X,' MAXIMUM RESIDUAL ERROR  [(FORCING - LAPLACIAN)/MAX',
     1 ' FORCING ]  =  ',1PE12.4)
C
C  DISCRETIZATION ERROR
      PMAX = 0.
      DO 3000 J=1,N
      DO 3000 I=1,M
       PHIX= 5.5-0.5*T1(J)
     1  + 0.5*T2(J)*COS(2.*PI*(X(I)-XC)/XL)
       PMAX = AMAX1(PMAX,ABS(PHIX))
       DUM3(I,J)=PHI(I,J)-PHIX
 3000 CONTINUE
      DSMAX=0.0
      DO 3007 I=2,M1
      DO 3007 J=2,N1
      DSMAX = AMAX1(DSMAX,ABS(DUM3(I,J)/PMAX))
 3007 CONTINUE
      PRINT 732, DSMAX
  732 FORMAT(1X,' MAXIMUM DISCRETIZATION ERROR  [(COMPUTED - ANALYTIC',
     1 ' PHI)/MAX ANALYTIC PHI]  =  ',1PE12.4)
C
  600 CONTINUE
      END
```

2. Test Example with Neumann Boundary Condition

```
      PROGRAM DRIVR2
C
C     This program uses SEVP to solve Poisson's equation
C               LAPLACIAN OF PHI = F
C     with Neumann boundary conditions in y
C     and Cyclic boundary conditions in x
C
      PARAMETER (M=82, N=53)
      PARAMETER (NBLK=1+(N-5+3)/9, NBLK1=NBLK-1)
      PARAMETER (M1=M-1, M2=M-2, N1=N-1, N2=N-2)
      PARAMETER (MN=M*N)
      DIMENSION AX(M,N),AY(M,N),CX(M,N),CY(M,N),BB(M,N),
     1 RINV(M2,M2,NBLK),RINV1(M2,M2,NBLK1),RCOR(M,3),RTILDA(M2),
     2 NBSIZ2(NBLK),IS(NBLK),IE(NBLK),SUMF(NBLK),IXNDX(M),
     3 DUMMY(M2,M2),B1(M2),B2(M2)
      DIMENSION F(M,N),PHI(M,N),ERR(M,N)
      DIMENSION F11(M),F1N(M),F21(N),F2M(N)
      DIMENSION X(M),Y(N),YP1(N),YP2(N)
      DIMENSION HXU(N),HXV(N1),HYU(N),HYV(N1)
      DIMENSION AA(N),AC(N),CC(N)
      DIMENSION T1(N),T2(N),T3(N),T4(N)
      DIMENSION DUM3(M,N)
C SET COEFFICIENTS OF DIFFERENCE EQUATION AND BOUNDARY CONDITIONS
C TO ZERO INITIALLY.
      DATA AX/MN*0.0/,CX/MN*0.0/,AY/MN*0.0/,CX/MN*0.0/,BB/MN*0.0/,
     1     F/MN*0.0/,F11/M*0.0/,F1N/M*0.0/,F21/N*0.0/,F2M/N*0.0/
C
      PI = 4.0*ATAN(1.0)
      AR = 6371.
      XL = 12000.0
      YL =  5000.0
C
C DEFINE GRID
      DELX = XL/FLOAT(M-2)
      DELY = YL/FLOAT(N-3)
      PRINT 990,DELX,DELY
  990 FORMAT(1H ,5HDELX=,1PE13.5,5X,5HDELY=,1PE13.5)
      DO 5 J=1,N
      Y(J)=DELY*(FLOAT(J-1)-0.5) + 2000.0
    5 CONTINUE
      DO 6 I=1,M
      X(I)=DELX*FLOAT(I-1)
    6 CONTINUE
      DELXSQ=DELX*DELX
      DELYSQ=DELY*DELY
C
C DEFINE MAP FACTORS
      DO 10 J=1,N
      HXU(J)=COS(Y(J)/AR)
      HYU(J)=1.0
   10 CONTINUE
      DO 20 J=1,N1
```

```
      HXV(J)=0.5*(HXU(J)+HXU(J+1))
      HYV(J)=1.0
   20 CONTINUE
C
C DEFINE FORCING FUNCTION
      XC=  0.0
      YC = 4500.0
      YW = 1500.0
      C1=YW/AR
      C2=1./(YW*YW)
      C3=2.*PI*PI*(YW/XL)*(YW/XL)
      DO 60 J=1,N
      YP1(J)=(Y(J)-YC)/YW
      YP2(J) = Y(J)/AR
   60 CONTINUE
      DO 61 J=1,N
      T1(J) = TANH(YP1(J))
      T2(J) = EXP(-YP1(J)*YP1(J))
   61 CONTINUE
      DO 62 J=1,N
      CSQ=COS(YP2(J))*COS(YP2(J))
      T3(J)=2.*T1(J)+C1*TAN(YP2(J))
      T4(J)= 2.*YP1(J)*YP1(J) - 1.0 + C1*YP1(J)*TAN(YP2(J))
     1    -  C3/CSQ
   62 CONTINUE
C  FORCING FUNCTION ONLY DEFINED IN INTERIOR OF DOMAIN.
      DO 65 J=2,N-1
      DO 65 I=2,M-1
      F(I,J) = 0.5*C2*T3(J)*(1.-T1(J)*T1(J))
     1          +   C2*T4(J)*T2(J)*COS(2.*PI*(X(I)-XC)/XL)
   65 CONTINUE
C
C
C NEUMANN BOUNDARY CONDITIONS IN Y
      A11 = 1.0
      A1N = 1.0
      Y11 = (Y(1)+(DELY/2.0)-YC)/YW
      Y1N = (Y(N)-(DELY/2.0)-YC)/YW
      T11 = TANH(Y11)
      T1N = TANH(Y1N)
      T21 = EXP(-Y11*Y11)
      T2N = EXP(-Y1N*Y1N)
      DO 150 I = 2,M-1
      F11(I) = - (DELY/YW)*( 0.5*(1.0-T11*T11)
     1  +Y11*T21*COS(2.*PI*(X(I)-XC)/XL) )
      F1N(I) = - (DELY/YW)*( 0.5*(1.0-T1N*T1N)
     1  +Y1N*T2N*COS(2.*PI*(X(I)-XC)/XL) )
  150 CONTINUE
C CYCLIC BOUNDARY CONDITIONS IN X
      ICYC = 1
      A21 = 0.0
      A2M = 0.0
C
C  FOR INDETERMINATE NEUMANN BOUNDARY CONDITIONS, A BOUNDARY VALUE
```

```
C  MUST BE PROVIDED AT A SINGLE BOUNDARY POINT (IX,JX) FOR THE SOLVER TO
C  OBTAIN A SOLUTION. THE BOUNDARY POINT SELECTED IN SEVP IS IX=2, JX=1.
C  FOR THE EFFICIENT CONVERGENCE OF THE SOLUTION, WE USE A ZERO VALUE
C  FOR THE SINGLE BOUNDARY POINT AND AS THE FIRST GUESS.
C
      DO 120 J=1,N
      DO 120 I=1,M
       PHI(I,J)= 0.0
  120 CONTINUE
C
C DEFINE COEFFICIENTS ON L.H.S. OF DIFFERENCE EQUATION.
C COEFFICIENTS DEFINED IN INTERIOR OF DOMAIN ONLY.
      DO 25  J=2,N-1
      AA(J)=((HXU(J)*HXV(J-1))/(HYV(J-1)*HYU(J)))*(DELXSQ/DELYSQ)
      CC(J)=((HXU(J)*HXV(J))/(HYV(J)*HYU(J)))*(DELXSQ/DELYSQ)
      AC(J)=-AA(J)-CC(J)
   25 CONTINUE
      DO 165 I = 2,M-1
      DO 165 J = 2,N-1
       AX(I,J) = 1.0
       CX(I,J) = 1.0
       AY(I,J) = AA(J)
       CY(I,J) = CC(J)
       BB(I,J) = AC(J)-2.0
  165 CONTINUE
C DEFINE R.H.S. OF DIFFERENCE EQUATION, SCALING FORCING
      DO 170 J=2,N-1
      DO 170 I=2,M-1
       CON = HXU(J)*HXU(J)*DELXSQ
       F(I,J) = F(I,J)*CON
  170 CONTINUE
      FMAX = 0.
      DO 305 J=2,N-1
      DO 305 I=2,M-1
       FMAX = AMAX1(FMAX,ABS(F(I,J)))
  305 CONTINUE
C
C
C TOLERANCE FOR SOLVER
      TOL = 1.0E-8
C
      IFLG = 1
      CALL SEVP(AX,AY,CX,CY,BB,RINV,RINV1,RCOR,NBSIZ2,IS,IE,IXNDX,
     1SUMF,DUMMY,RTILDA,PHI,F,ERR,F11,F1N,F21,F2M,M,N,M1,N1,M2,N2,
     2NBLK,NBLK1,B1,B2,A11,A1N,A21,A2M,ICYC,TOL,IFLG)
C
      PRINT 710
  710 FORMAT(1X,'BLOCK SIZE GIVEN BY NBSIZ2(1)+2 (1ST BLOCK) OR',
     1 ' NBSIZ2()+3 (OTHER BLOCKS)'/,1X,' NBSIZ2: ')
      PRINT 711, NBSIZ2
  711 FORMAT(1X,10I4)
C
C RESIDUAL ERROR OF LAPLACIAN
      FNORM=FMAX
```

```
      ERMAX=0.0
      DO 750 I=2,M1
      DO 750 J=2,N1
      ERMAX = AMAX1(ERMAX,ABS(ERR(I,J)/FNORM))
  750 CONTINUE
      PRINT 751, ERMAX
  751 FORMAT(1X,' MAXIMUM RESIDUAL ERROR  [(FORCING - LAPLACIAN)/MAX',
     1 ' FORCING ]  =  ',1PE12.4)
C
C  DIFFERENCE AT PRESCRIBED BOUNDARY POINT (IX,JX)
      PHI21=5.5-0.5*T1(1)+0.5*T2(1)*COS(2.*PI*(X(2)-XC)/XL)
      PDIFF = PHI21 - PHI(2,1)
C  DISCRETIZATION ERROR
      PMAX = 0.
      DO 3000 J=1,N
      DO 3000 I=1,M
       PHIX= 5.5-0.5*T1(J)
     1  + 0.5*T2(J)*COS(2.*PI*(X(I)-XC)/XL)
       DUM3(I,J)=PHI(I,J)-PHIX+PDIFF
       PMAX = AMAX1(PMAX,ABS(PHIX))
 3000 CONTINUE
      PNORM = PMAX
      DSMAX=0.0
      DO 3007 I=2,M1
      DO 3007 J=2,N1
      DSMAX = AMAX1(DSMAX,ABS(DUM3(I,J)/PNORM))
 3007 CONTINUE
      PRINT 732, DSMAX
  732 FORMAT(1X,' MAXIMUM DISCRETIZATION ERROR  [(COMPUTED - ANALYTIC',
     1 ' PHI)/MAX ANALYTIC PHI]  =  ',1PE12.4)
C
  600 CONTINUE
      END
```