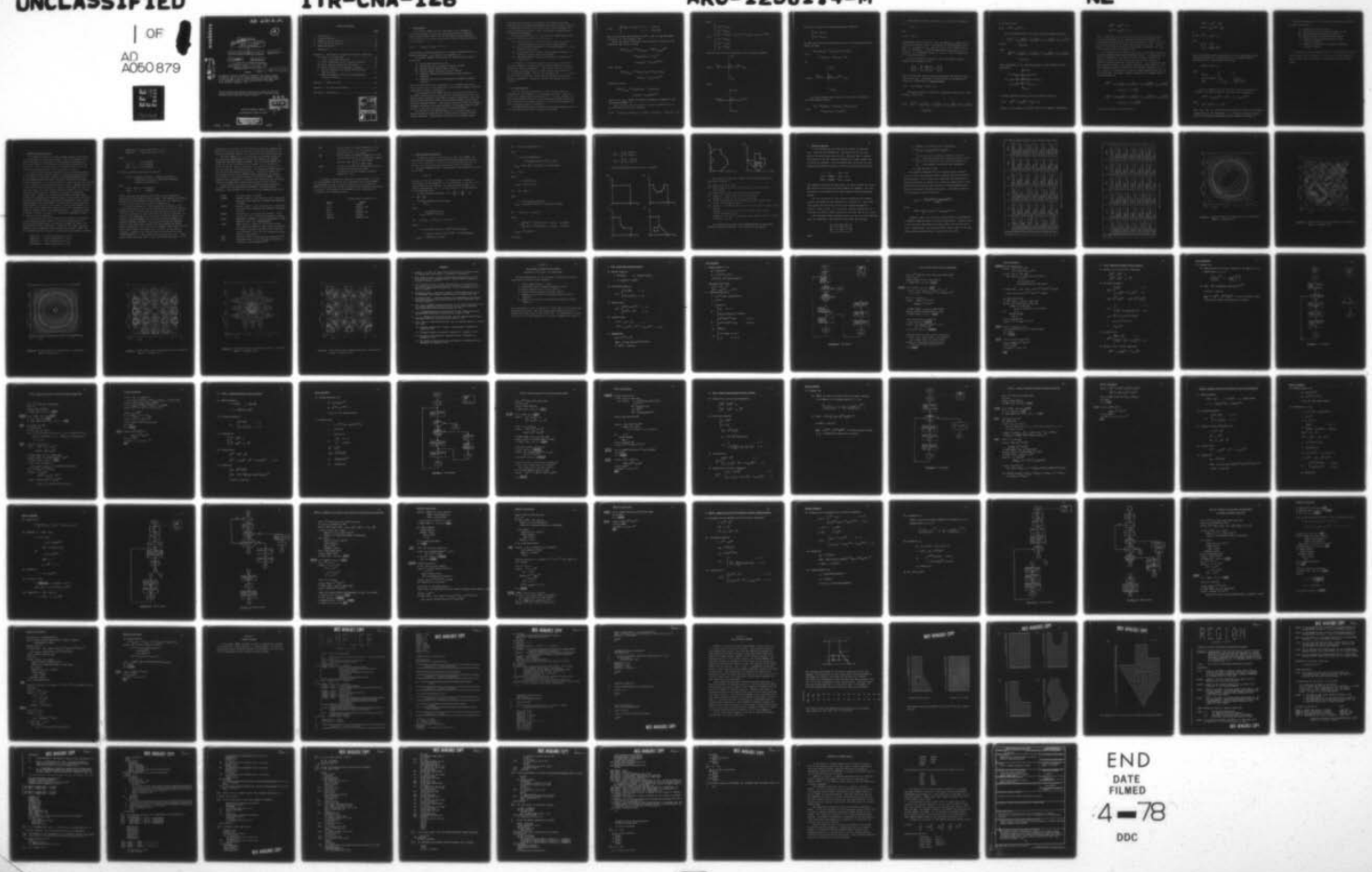
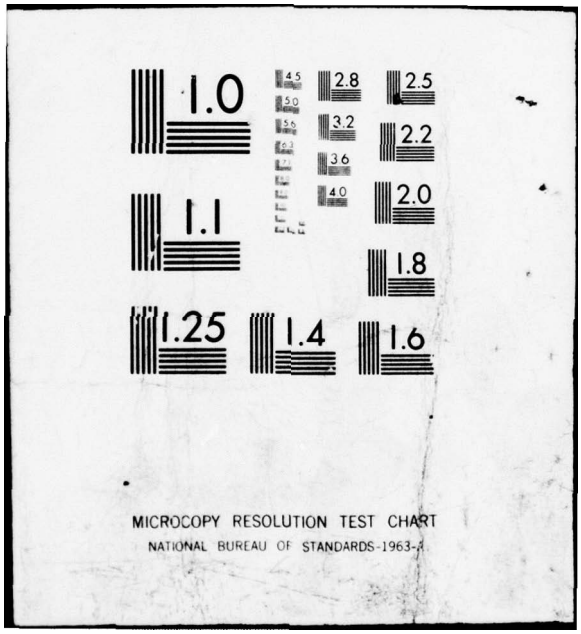
AD-A050 879 TEXAS UNIV AT AUSTIN CENTER FOR NUMERICAL ANALYSIS F/G 9/2
ITPACK REPORT: NUMERICAL STUDIES OF SEVERAL ADAPTIVE ITERATIVE --ETC(U)
AUG 77 D R KINCAID, R G GRIMES DAHC04-74-G-0198
UNCLASSIFIED ITR-CNA-126 ARO-12301.4-M NL



END
DATE FILMED
4-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ARO 12301.4-M

12 SC

6 ITPACK REPORT: NUMERICAL STUDIES OF SEVERAL ADAPTIVE ITERATIVE ALGORITHMS*

by

10 David R. Kincaid and Roger G. Grimes

12 93P.

11 August 1977

14 ITR-CNA-126

15 DAHC04-74-G-0198, NSF-MCS76-03141

18 ARO

19 12301.4-M

Abstract

Six adaptive iterative algorithms are studied for six elliptic partial differential equations on six regions compatible with subroutine REGION. An effort was made to make the resulting preliminary ITPACK code conform to the "ELLPACK Contributor's Guide--Initial Version," CSD TR 208, Purdue University, November 1, 1976.

*Work on this paper was supported in part by grant DAHC04-74-G-0198 from the Army Research Office and grant MCS76-03141 from the National Science Foundation at The University of Texas at Austin.

CENTER FOR NUMERICAL ANALYSIS

THE UNIVERSITY OF TEXAS AT AUSTIN

406 262

JOB

Table of Contents

1. Introduction

The initial ITPACK code was conceived to be a package of Fortran subroutines to solve the large sparse positive definite linear systems which arise from the five-point finite difference discretization of a general self-adjoint elliptic partial differential equation

$$(1.1) \qquad (a u_x)_x + (c u_y)_y + f u = g$$

with Dirichlet boundary conditions on a region compatible with the REGION subprogram. (For details, see Appendix 3.)

The current ITPACK code contains the following six iterative algorithms

I. Jacobi Semi-iteration (J-SI)
II. Compressed Jacobi Conjugate Gradient (CJ-CG)
III. Reduced System Semi-iteration (RS-SI)
IV. Reduced System Conjugate Gradient (RS-CG)
V. Symmetric Successive Overrelaxation Semi-iteration (SSOR-SI)
VI. Symmetric Successive Overrelaxation Conjugate Gradient (SSOR-SI)

which are developed in the monograph [1] by Hageman and Young. These methods will not be motivated in this report; however, detailed algorithms are given in Appendix 1.

Future plans for the ITPACK project are many and varied with the major limiting factor being time for implementation of the code. Various other iterative algorithms are being considered at this time. These include the Block Jacobi Semi-iterative Method and the Block Conjugate Gradient Method. Also coding schemes for mixed and Neumann boundary conditions are being developed. Yet another phase of this project is the use of various finite difference stencils.

The purpose of the ITPACK project is to develop, study, and analyze iterative algoirthms for solving elliptic partial differential equations. The principal activities are centered around

improving those iterative algorithms which involve efficient stopping tests and effective parameter determination when computing the numerical solution of the large sparse matrix problems from elliptic equations whenever finite difference or finite element procedures are employed.

It is anticipated that the code from the ITPACK project will have the following benefits and utilization:

(a) the development of ELLPACK modules which use adaptive iterative procedures to solve the linear systems

(b) add to existing knowledge of the effectiveness of various iterative algorithms

(c) allow comparisons between these iterative schemes and between iterative and direct methods

(d) the development of quality software as a research and teaching tool

In Section 2, background material relating (1.1) and basic iterative methods is given with the detailed adaptive iterative algorithms stated in Appendix 1. The overall structure of ITPACK is outlined in Section 3 with a sample of the code required to use the current ITPACK. The six test problems and test regions are set-forth in Section 4 with complete details on subroutine REGION in Appendix 3. Numerical results and figures are given in Section 5.

2. Iterative Methods

The six iterative algorithms covered by this study are developed in the monograph [1] by Hageman and Young. Consequently, we will not repeat these derivations here. We will, however, present some material related to the development of these algorithms which will aid in the understanding of the detailed statements of those procedures given in Appendix 1.

We consider the general self-adjoint elliptic partial differential equation with Dirichlet boundary conditions.

$$(2.1)\qquad \begin{cases} (au_x)_x + (cu_y)_y + fu = g \ , & (x,y) \in R \\ u = q \ , & (x,y) \in \partial R \end{cases}$$

Here a,c,f,g,q may be functions of both x and y, and the region is denoted R with boundary ∂R.

Using the central difference discretization at the grid point associated with (i,j), we have

$$(au_x)_x\Big|_{(i,j)} \approx \{(au_x)_{(i+\frac{1}{2},j)} - (au_x)_{(i-\frac{1}{2},j)}\}h^{-1}$$

$$\approx \{a_{i+\frac{1}{2},j}[u_{i+1,j} - u_{ij}]h^{-1}$$

$$-a_{i-\frac{1}{2},j}[u_{ij} - u_{i-1,j}]h^{-1}\}h^{-1}$$

Hence, we use

$$(au_x)_x\Big|_{(i,j)} \approx \{a_{i+\frac{1}{2},j}\,u_{i+1,j} + a_{i-\frac{1}{2},j}\,u_{i-1,j}$$

$$-(a_{i+\frac{1}{2},j} + a_{i-\frac{1}{2},j})u_{ij}\}h^{-2}$$

Similarly, we use

$$(cu_y)_y\Big|_{(i,j)} \approx \{c_{i,j+\frac{1}{2}}u_{i,j+1} + c_{i,j-\frac{1}{2}}u_{i,j-1}$$

$$-(c_{i,j+\frac{1}{2}} + c_{i,j-\frac{1}{2}})u_{ij}\}h^{-2}$$

Here we let u_{ij} denote the discrete variable as opposed to the continuous variable u.

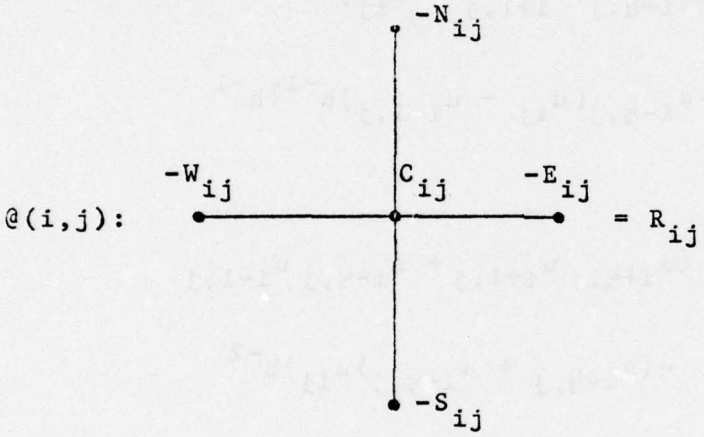
Thus at (i,j), the self-adjoint elliptic equation (2.1) is approximated by the linear equation

$$(2.2)\qquad -S_{ij}u_{i,j-1} - W_{ij}u_{i-1,j} + C_{ij}u_{ij} - E_{ij}u_{i+1,j} - N_{ij}u_{i,j+1} = R_{ij}$$

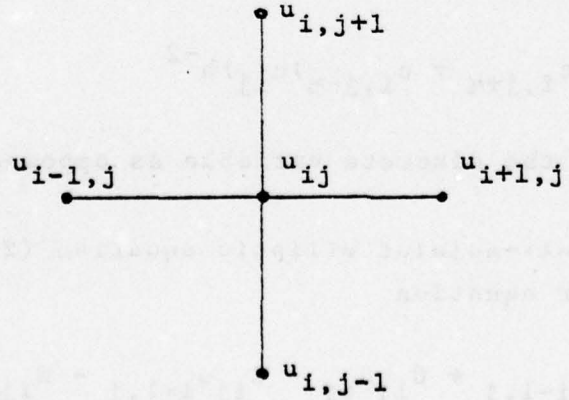
where

$$
(2.3)\quad \begin{cases} S_{ij} = c_{i,j-\frac{1}{2}} \\ W_{ij} = a_{i-\frac{1}{2},j} \\ C_{ij} = (a_{i+\frac{1}{2},j} + a_{i-\frac{1}{2},j}) + (c_{i,j+\frac{1}{2}} + c_{i,j-\frac{1}{2}}) - h^2 f_{ij} \\ E_{ij} = a_{i+\frac{1}{2},j} \\ N_{ij} = c_{i,j+\frac{1}{2}} \\ R_{ij} = -h^2 g_{ij} \end{cases}
$$

Equation (2.2) can be illustrated by the following stencils



where



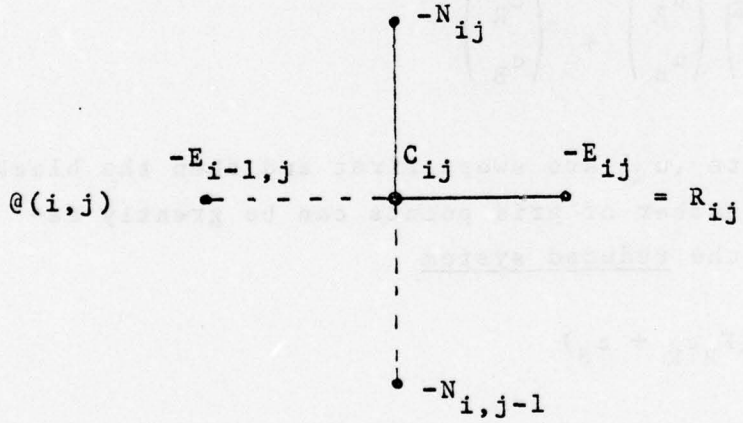
From (2.3), we have the following symmetry condition

$$\begin{cases} W_{ij} = E_{i-1,j} \\ S_{ij} = N_{i,j-1} \end{cases}$$

so that only four coefficient values need to be stored per grid point. Hence, we have

$$-N_{i,j-1}u_{i,j-1} - E_{i-1,j}u_{i-1,j} + C_{ij}u_{ij} - E_{ij}u_{i+1,j} - N_{ij}u_{i,j+1} = R_{ij} \tag{2.4}$$

and



Since only regular grid points are considered, we have for the basic linear equation

$$u_{ij} = (E_{ij}u_{i+1,j} + N_{ij}u_{i,j+1} + E_{i-1,j}u_{i-1,j} + N_{i,j-1}u_{i,j-1} + R_{ij})/C_{ij} \tag{2.5}$$

Using matrix notation, equations (2.4) and (2.5) correspond to

$$Au = b$$

and

$$u = Bu + c \tag{2.6}$$

respectively, where $D^{-1}A = I-B$ and $D = \mathrm{diag}(C_{ij})$. Notice that if the k-th equation in $Au = b$ corresponds to the grid-point (i,j) then b_k is equal to R_{ij} plus the sum of some terms in (2.2), with u replaced by q, for boundary-points adjacent to (i,j). Clearly, A is symmetric while B is not. It can be shown that A is positive definite.

When the red-black ordering is used, the basic iterative system (2.6) assumes the form

$$\begin{pmatrix} u_R \\ u_B \end{pmatrix} = \begin{pmatrix} 0 & F_R \\ F_B & 0 \end{pmatrix} \begin{pmatrix} u_R \\ u_B \end{pmatrix} + \begin{pmatrix} c_R \\ c_B \end{pmatrix}$$

where the red grid points (u_R) are swept first and then the black grid points (u_B). The number of grid points can be greatly decreased by considering the <u>reduced system</u>

$$u_B = F_B F_R u_B + (F_B c_R + c_B) \tag{2.7}$$

The basic iterative equation for algorithms based on the Jacobi method is from (2.5)

$$u_{ij}^{(n+1)} = (E_{ij}u_{i+1,j}^{(n)} + N_{ij}u_{i,j+1}^{(n)} + E_{i-1,j}u_{i-1,j}^{(n)} + N_{i,j-1}u_{i,j-1}^{(n)} + R_{ij})/C_{ij} \tag{2.8}$$

or in matrix form

$$u^{(n+1)} = Bu^{(n)} + c \tag{2.9}$$

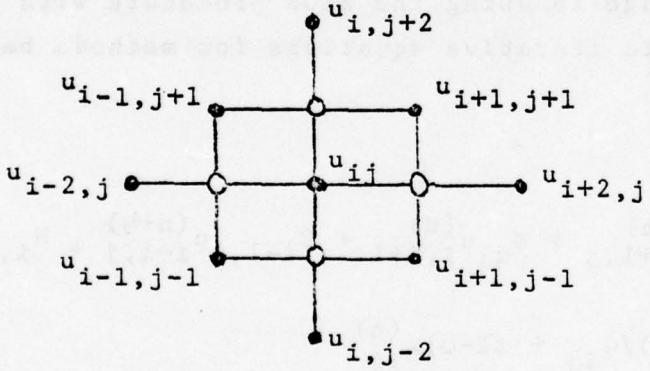
For the reduced system, the basic iterative equation would be

$$u_{ij}^{(n+1)} = (E_{ij}u_{i+1,j}^{(n+\frac{1}{2})} + N_{ij}u_{i,j+1}^{(n+\frac{1}{2})} + E_{i-1,j}u_{i-1,j}^{(n+\frac{1}{2})} + N_{i,j-1}u_{i,j-1}^{(n+\frac{1}{2})} + R_{ij})/C_{ij} \tag{2.10}$$

where

$$u_{k\ell}^{(n+\frac{1}{2})} = (E_{k\ell}u_{k+1,\ell}^{(n)} + N_{k\ell}u_{k,\ell+1}^{(n)} + E_{k-1,\ell}u_{k-1,\ell}^{(n)} + N_{k,\ell-1}u_{k,\ell-1}^{(n)} + R_{k,\ell})/C_{k,\ell}$$

This corresponds to the following stencil at each regular iterior grid point



The basic iterative equation for the resulted system is

$$u_B^{(n+1)} = F_B F_R u_B^{(n)} + F_B c_R + c_B \tag{2.11}$$

However, it is easier to consider this as two separate iterations.

$$u_R^{(n+\frac{1}{2})} = F_R u_B^{(n)} + c_R$$

$$u_B^{(n+1)} = F_B u_R^{(n+\frac{1}{2})} + c_B$$

First, a sweep of the red grid points would be done which involves a "weighted-average" of the adjacent black grid points with the results being stored in the red storage locations. This is followed by a similar sweep of the black grid points using (2.10). The net result is (2.11) with the black grid-points at iteration $n+1$ and the red grid points at iteration $n+\frac{1}{2}$.

The SSOR-SI and the SSOR-CG method use for its basic iterative equation the SSOR scheme with relaxation factor ω to accelerate the rate of convergence. The SSOR procedure involves a forward and backward sweep of all grid points with the natural ordering. A symmetric positive definite iteration matrix, $\mathcal{B}_\omega$, is obtained from this to-and-fro sweep. The natural ordering is used since the optimum relaxation factor for SSOR with the red-black ordering is $\omega = 1$, i.e., there is no advantage in using the SSOR procedure with the red-black ordering. The basic iterative equations for methods based on the SSOR method is

$$u_{ij}^{(n+\frac{1}{2})} = \omega(E_{ij}u_{i+1,j}^{(n)} + N_{ij}u_{i,j+1}^{(n)} + E_{i-1,j}u_{i-1,j}^{(n+\frac{1}{2})} + N_{i,j-1}u_{i,j-1}^{(n+\frac{1}{2})} + R_{ij})/C_{ij} + (1-\omega)u_{ij}^{(n)}$$

$$u_{ij}^{(n+1)} = \omega(E_{ij}u_{i+1,j}^{(n+1)} + N_{ij}u_{i,j+1}^{(n+1)} + E_{i-1,j}u_{i-1,j}^{(n+\frac{1}{2})} + N_{i,j-1}u_{i,j-1}^{(n+\frac{1}{2})} + R_{ij})/C_{ij} + (1-\omega)u_{i,j}^{(n+\frac{1}{2})}$$

This can be written in matrix form as

$$u^{(n+\frac{1}{2})} = \mathcal{L}_\omega u^{(n)} + k_\omega^{(F)}$$

$$u^{(n+1)} = \mathcal{U}_\omega u^{(n+\frac{1}{2})} + k_\omega^{(B)}$$

or

$$u^{(n+1)} = \mathcal{S}_\omega u^{(n)} + k_\omega \tag{2.12}$$

where

$$\begin{cases} \mathcal{S}_\omega = \mathcal{U}_\omega \mathcal{L}_\omega \\ k_\omega = \mathcal{U}_\omega k_\omega^{(F)} + k_\omega^{(B)} \end{cases}$$

The six iterative methods investigated in this study apply either Chebyshev Acceleration (Semi-iteration) or Conjugate Gradient Acceleration to a basic method of the form

$$u^{(n+1)} = \mathcal{G} u^{(n)} + k$$

where

	$\underline{\mathcal{G}}$		$\underline{k}$
Jacobi	: B	,	c
Reduced System	: $F_B F_R$	,	$F_B c_R + c_B$
SSOR	: $\mathcal{S}_\omega = \mathcal{U}_\omega \mathcal{L}_\omega$	,	$\mathcal{U}_\omega k_\omega^{(F)} + k_\omega^{(B)}$

Both the Chebyshev and the Conjugate Gradient Acceleration procedures for basic methods of this form can be written as

$$u^{(n+1)} = \rho_{n+1}(\gamma_n \delta^{(n)} + u^{(n)}) + (1 - \rho_{n+1})u^{(n-1)}$$

where

$$\delta^{(n)} = \mathcal{G} u^{(n)} + k - u^{(n)} .$$

Here ρ_{n+1} and γ_n are acceleration parameters which are determined automatically in the algorithms. As a reference for these methods and the acceleration algorithms consult Hageman and Young [1].

Detailed descriptions of the following six adaptive algorithms are given in Appendix 1.

I. Jacobi Semi-iteration (J-SI)
II. Compressed Jacobi Conjugate Gradient (CJ-CG)
III. Reduced System Semi-iteration (RS-SI)
IV. Reduced System Conjugate Gradient (RS-CG)
V. Symmetric Successive Overrelaxation Semi-Iteration (SSOR-SI)
VI. Symmetric Successive Orerrelaxation Conjugate Gradient (SSOR-CG)

We should note that procedures based on the SSOR method require twice as much work per iteration. Also, that the J-CG method requires exactly twice the number of iterations as does the RS-CG method.

3. ITPACK Structure and Use

The ITPACK collection of codes preforms various tasks which are accomplished in individual modules. The basic modules are (1) grid definition, (2) generation of the nonzero coefficients of the linear system, (3) definition of the ordering vector for the grid points, (4) initialization of the unknown vector, (5) solution by an iterative method, and (6) output of results.

The grid definition is accomplished by the subroutine REGION. In its present state, REGION accepts a polygonal parameterization of the domain of interest. However, this parameterization must be established using horizontal, vertical, and forty-five degree lines. Consequently, it is only designed to accept uniform mesh spacing. It will however allow regions with holes in them. REGION generates a rectangular grid and defines an integer array GTYPE such that for each grid point (I,J) the value of GTYPE is either 1,2, or 3 which indicates either interior, boundary, or exterior grid points, respectively. REGION also defines arrays GRIDX and GRIDY which contain the coordinates of the grid points in the x and y direction. In addition, REGION defines the minimum and maximum x and y values (AX,BX,AY,BY), the actual number of grid points in each direction (NGRIDX,NGRIDY), and the total number of grid points (NGRPTS). The remainder of the ITPACK code needs only the grid information generated by REGION and not the parameterization. A complete listing of REGION is given in Appendix 3 along with additional details on the use of this subroutine.

The next task is that of generating the nonzero coefficients of the associated linear system. This is accomplished in the Fortran module FIVEPT which is currently designed to handle only self-adjoint elliptic operators. Therefore, the linear system is symmetric and a symmetric storage scheme can be used. These nonzero coefficients are placed in a four-column array COEF as follows:

```
COEF(IJ,1)    = center coefficient at (I,J)
COEF(IJ,2)    = north coefficient at (I,J)
COEF(IJ,3)    = east coefficient at (I,J)
COEF(IJM1,2)  = south coefficient at (I,J)
```

```
COEF(IM1J,3) = west coefficient at (I,J)
COEF(IJ,4)   = right-hand side at (I,J)
```

where

```
IJ   = I     + (J-1)*NGRIDX
IJM1 = I     + (J-2)*NGRIDX
IM1J = (I-1) + (J-1)*NGRIDX
```

The basic iterative equation then becomes

```
U(IJ) = (COEF(IJ,3)*U(IP1J) + COEF(IJ,2)*U(IJP1)
        + COEF(IM1J,3)*U(IM1J) + COEF(IJM1,2)*U(IJM1)
        + COEF(IJ,4))/COEF(IJ,1)
```

where

```
IP1J = (I+1) + (J-1)*NGRIDX
IJP1 =   I   +     J*NGRDIX .
```

FIVEPT requires the subroutine PDE which is user supplied or generated by the ELLPACK control program. PDE computes the coefficients of the self-adjoint elliptic operator at the point (x,y). A sample of the use of PDE is given in Appendix 2.

To allow extensions to three-dimensional problems, a one-dimensional array is used for the unknown vector with the elements ordered so that a linear sweep through this array is the same as proceeding through the grid points with the natural ordering. At present, four orderings have been coded and tested, namely, the natural ordering (NATORD), the red-black ordering (RBORD), the diagonal ordering (DIAGORD), and the spiral ordering (SPIRORD). Each of these subroutines defines the arrays NDXEQ and INVNDX. NDXEQ is defined in such a way that J = NDXEQ(I) means that the I-th point that is swept is actually the J-th point in the natural ordering. INVNDX is the inverse index array defined such that INVNDX(NDXEQ(I)) = I. This convention is outlined in [5]. These arrays enable the same code to use any ordering specified. It is

interesting to note that for efficiency in production software one might want to write code so that the ordering was encoded into the iterative algorithm; however, this would not allow any versatility.

The next ITPACK task is to initialize the unknown vector in the subroutine INTUNK which uses the user supplied (or ELLPACK generated) routines APXUNK and BCOND. The subroutine APXUNK computes the initial approximation (or guess) for the unknown (or solution) vector. When no information is available the value of zero is taken for the initial guess. The subroutine BCOND computes the values of the boundary grid points. Subroutine INTUNK sets the elements of the array UNKNWN, which corresponds to interior grid points, to the values supplied by subroutine APXUNK. Other elements which correspond to boundary grid points are set to values supplied by BCOND while exterior grid points are set to zero.

Certain input data besides the parameterization information for REGION and the subroutines PDE APXUNK and BCOND must be supplied. The following is a list of all the necessary input data:

LEVEL	Controls output of REGION
NGRDXD	Maximum number of grid points in the x direction. Also the first dimension of GTYPE, and dimension of GRIDX.
NGRDYD	Maximum number of grid points in the y direction. Also the second dimension of GTYPE and dimension of GRIDY.
MXNCOE	Set equal to 4 for five-point stencil. Later a value of 6 will indicate a nine-point stencil.
MXNEQ	Dimension of UNKNWN and first dimension of COEF. Commonly taken to be NGRDXD*NGRDYD.
ITMAX	Upper bound on number of iterations the user will allow the method to take before convergence. If ITMAX is reached, the method will stop and exit naturally. Note that the stopping criteria may not be satisfied.
ZETA	Tolerance level in stopping test (usually 10^{-6}).
EPSI	Tolerance level in root solving and checks in division by zero.(usually 10^{-6}).

CME — Initial guess of largest eigenvalue of the iteration matrix. If no information is known, CME = 0.0 is acceptable.

SME — Initial guess of smallest eigenvalue of the iteration matrix. If no information is known then set SME $= \begin{cases} 0.0 \text{ if CASE = FALSE} \\ -1.0 \text{ if CASE = TRUE} \end{cases}$

CASE — A logical variable to indicate which case of the adaptive procedure is used.

F — A factor used in the adaptive procedure (usually F = .75).

A workspace area must be supplied in blank common. The size of this workspace varies for each method and the variable MXNEQ. The workspace is used in various capacities, but is primarily needed for the auxillary storage utilized in the iterative algorithms. At present, the workspace array WORKSP must be dimensioned as follows for each iterative method:

Method	Minimum Value of Dimension for WORKSP
J-SI	3*MXNEQ
CJ-CG	3*MXNEQ + 200
RS-SI	2*MXNEQ
RS-CG	4*MNXEQ + 200
SSOR-SI	5*MXNEQ
SSOR-CG	6*MXNEQ + 200

4. Test Problems and Regions

In order to test the code written to date for ITPACK, six test partial differential equations with known solutions and six regions were selected. The test cases were designed so that the behavior of the six iterative algorithms could be monitored.

The test equations cover a wide range of self-adjoint operators of the form

$$L(u) = f$$

For each of the test problems, u is known over a region R. Furthermore, on the boundary of R, the function u is set to the true solution of the problem. For each iterative method, the initial approximation for u on the iterior of R was selected to be identially zero.

In the following equations, $\nabla^2 \cdot = \frac{\partial^2}{\partial x^2} \cdot + \frac{\partial^2}{\partial y^2} \cdot$ and $\nabla \cdot = \frac{\partial}{\partial x} \cdot + \frac{\partial}{\partial y} \cdot$.

The test problems are as follows:

(1) $$\nabla^2 u = f$$

where

$$f = 6xye^{x+y}(xy+x+y-3)$$
$$u_{true} = 3xye^{x+y}(x-1)(y-1)$$

(2) $$(e^{xy}u_x)_x + (e^{-xy}u_y)_y - u/(1+x+y) = f$$

where

$$f = \pi\{x \sin(\pi x) \cos(\pi y) + 3ye^{2xy} \cos(\pi x) \sin(\pi y)\}$$
$$+ \sin(\pi x) \sin(\pi y)\{(2y^2-\pi^2)e^{2xy} - \pi^2 - e^{xy}/(1+x+y)\}$$
$$u_{true} = e^{xy}\sin(\pi y) \sin(\pi x).$$

(3) $\nabla\cdot[(1 + \sin(\frac{\pi}{2}(x+y)\nabla u] = f$

where

$$f = 8[1 + \sin(\tfrac{\pi}{2}(x+y))] + \pi^2 \sin(\tfrac{\pi}{2}(x+y))[(x - \tfrac{1}{2})^2 + (y - \tfrac{1}{2})^2]$$

$$u_{true} = 2[(x - \tfrac{1}{2})^2 + (y - \tfrac{1}{2})^2]/[1 + \sin(\tfrac{\pi}{2}(x+y))].$$

(4) $\nabla^2 u = f$

where

$$f = 8(x^2+y^2-x-y)$$

$$u_{true} = 4xy(x-1)(y-1)$$

(5) $\nabla^2 u - 100u = f$

where

$$f = 300\ \cosh(20y)/\cosh(20)$$

$$u_{true} = \cosh(10x)/\cosh(10) + \cosh(20y)/\cosh(20).$$

(6) $(A(x)u_x)_x + (C(y)u_y) = f$

where

$$f = \begin{cases} (2+x)e^x - \pi^2(1+y)\sin(\pi y) + \cos(\pi y), & x,y\in[0,\tfrac{1}{2}] \\ (1-x)e^x - \pi^2(2-y)\sin(\pi y) - \cos(\pi y), & x,y\in(\tfrac{1}{2},1] \end{cases}$$

$$u_{true} = e^x + \sin(\pi y)$$

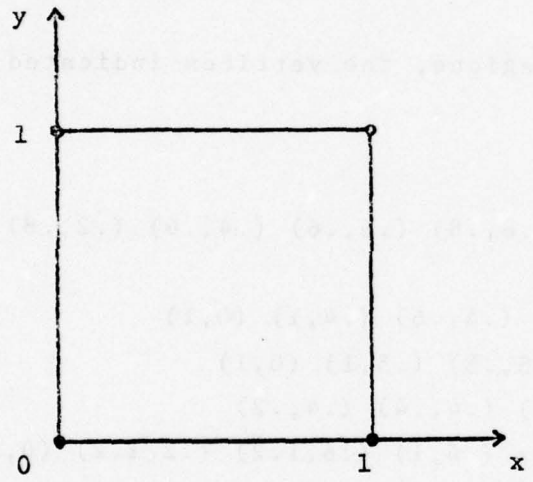
and where

$$A(x) = \begin{cases} 1+x, & x\in[0,\tfrac{1}{2}] \\ 2-x, & x\in(\tfrac{1}{2},1] \end{cases}$$

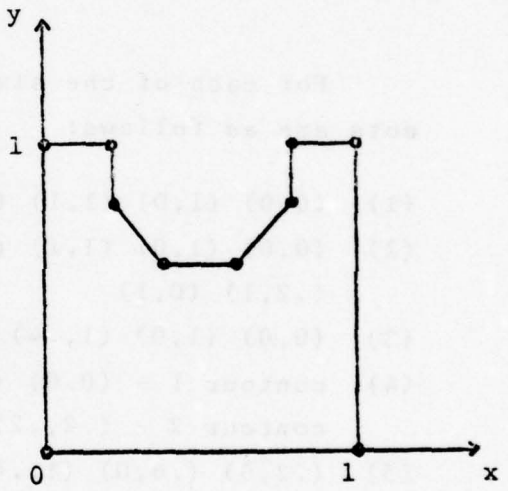
$$C(x) = \begin{cases} 1+y, & y\in[0,\tfrac{1}{2}] \\ 2-y, & y\in(\tfrac{1}{2},1] \end{cases}$$

The six test regions selected are as follows.

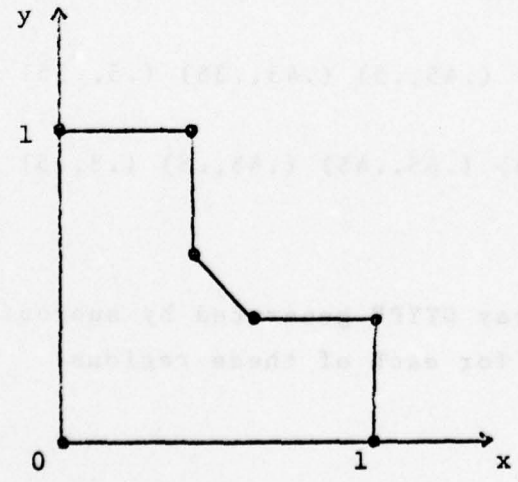
(1)



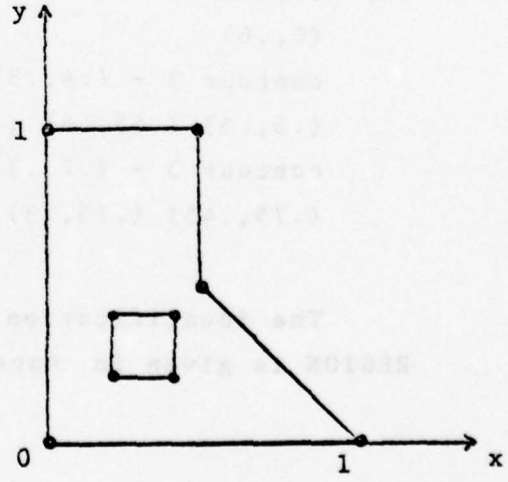
(2)



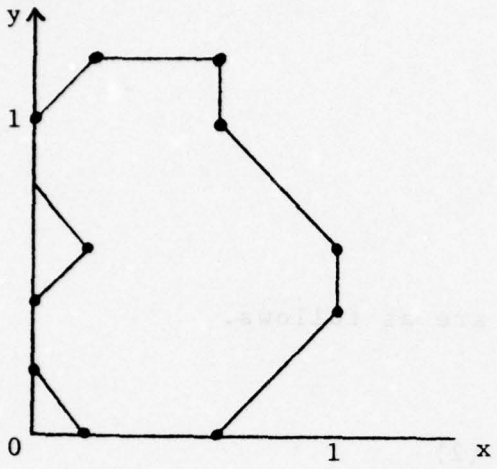
(3)



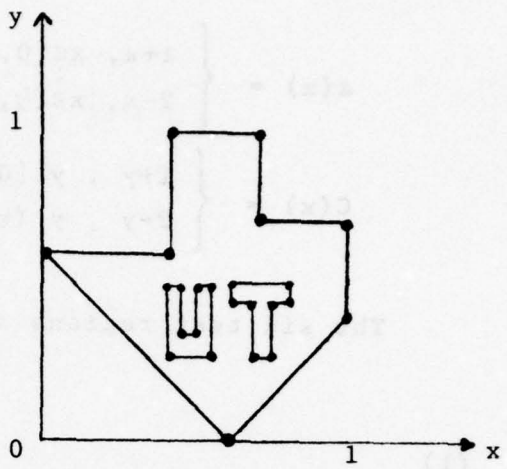
(4)



(5)



(6)



For each of the six test regions, the vertices indicated by dots are as follows:

(1) (0,0) (1,0) (1,1) (0,1)

(2) (0,0) (1,0) (1,1) (.8,1) (.8,.8) (.6,.6) (.4,.6) (.2,.8) (.2,1) (0,1)

(3) (0,0) (1,0) (1,.4) (.6,.4) (.4,.6) (.4,1) (0,1)

(4) contour 1 - (0,0) (1,0) (.5,.5) (.5,1) (0,1)
contour 2 - (.2,.2) (.2,.4) (.4,.4) (.4,.2)

(5) (.2,0) (.6,0) (1,.4) (1,.6) (.6,1) (.6,1.2) (.2,1.2) (0,1) (0,.8) (.2,.6) (0,.4) (0,.2)

(6) contour 1 - (.6,0) (1,.4) (1,.7) (.7,.7) (.7,.9) (.4,.9) (.4,.6) (0,.6)
contour 2 - (.4,.3) (.4,.5) (.45,.5) (.45,.35) (.5,.35) (.5,.5) (.55,.5) (.55,.3)
contour 3 - (.7,.3) (.7,.45) (.65,.45) (.65,.5) (.8,.5) (.8,.45) (.75,.45) (.75,.3)

The identification grid array GTYPE generated by subroutine REGION is given in Appendix 3 for each of these regions.

5. Numerical Results

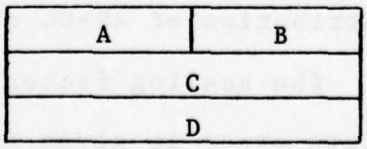
In this section we will discuss the results of numerical test runs with the ITPACK code. The first set of test runs were on the unit square with h = 1/40. This was done to compare the results of the six iterative methods on a common region over a variety of problems. All six methods were run on the six test equations described in Section 4 with the following initial data:

F = .75 CME = 0.0,
EPSI = .000001, SME = 0.0,
ZETA = .000001, CASE = .FALSE.

The red/black ordering was used with J-SI, RS-SI, RS-CG, and CJ-CG. The natural ordering was used with SSOR-SI and SSOR-CG. All runs were made on a CDC 6600 with the MNF compiler and UT2D operating system.

For the second set of test cases we considered the elliptic operator described as test equation (2) in Section 4. Each iterative method was used with the five test regions described in Section 4 with h = 1/20. All input data, initial conditions, etc. used were the same as those selected in the first set of test cases.

The following tables represent these runs and give the resulting data for comparison. Each block of the tables has the form



where

A = Number of iterations until convergence

$$B = \|u_{true} - u_{computed}\|_{D^{1/2}} / \|u_{true}\|_{D^{1/2}}$$

$$C = \begin{cases} \text{For J-SI, RS-SI, SSOR-SI, SSOR-CG, a list of iterations numbers where paramcters were changed} \\ \text{For CJ-CG and RS-CG, the iteration number where a new estimate of CME was last calculated} \end{cases}$$

D = Last estimate of CME.

For further comparison of the iterative methods, contour plots of error distributions between computed and true solutions were generated. All of these test cases used the self-adjoint operator described as problem (4) on the unit square with h = 1/20. Problem (4) was used so there would be no discretization error from the five point difference equations. The contour plots are of the function z(x,y) defined as

$$z(x,y) = \frac{|u_{true}(x,y) - u_{computed}(x,y)|}{SCALE},$$

where

$$SCALE = \max_{x,y} |u_{true}(x,y) - u_{computed}(x,y)|.$$

Figures 1 thru 4 are the error distributions at convergences of SSOR-SI, SSOR-CG, RS-SI, and RS-CG respectively. Figures 5 and 6 show the error distribution of RS-CG after five and ten iterations, respectively. The scaling factor, SCALE, which is the maximum pointwise absolute error is given for each case.

Problem	J-SI	CJ-CG	RS-SI	RS-CG	SSOR-SI	SSOR-CG
(1)	238 | 3.13384×10^{-4}	100 | 3.12327×10^{-4}	114 | 3.13105×10^{-4}	50 | 3.12327×10^{-4}	30 | 3.123832×10^{-4}	28 | 3.12329×10^{-4}
	0,2,6,14,32	24	0,2,4,8,14,34	14	0,1	0,1,3,13
	.9967537	.9966142	.9969005	.9966938	.9954121	.9956275
(2)	248 | 4.02996×10^{-4}	118 | 4.04033×10^{-4}	112 | 4.06287	59 | 4.04033×10^{-4}	32 | 4.04006×10^{-4}	28 | 4.04032×10^{-4}
	0,3,8,23,238	20	0,2,6,17	10	0,4	0,1,4,12
	.9968162	.9966055	.9967394	.9966057	.9965766	.9955249
(3)	243 | 2.32010×10^{-4}	92 | 2.33053×10^{-4}	137 | 2.32245×10^{-4}	46 | 2.33053×10^{-4}	38 | 2.33015×10^{-4}	34 | 2.33053×10^{-4}
	0,2,4,7,11,17,27,38,98	44	0,1,3,5,8,13,19,24	22	0,2,4,8	0,1,3,5,8,11,17
	.9972762	.9972372	.9970658	.9972375	.9967498	.9959858
(4)	228 | 1.08042×10^{-6}	56 | 2.13821×10^{-8}	107 | 8.16866×10^{-7}	28 | 2.13821×10^{-8}	31 | 4.48987×10^{-8}	25 | 7.26106×10^{-9}
	0,3,6,19	16	0,2,5,27	8	0,3	0,1,4
	.9967810	.9967999	.9969111	.9968003	.9957019	.9951709
(5)	98 | 9.65606×10^{-3}	74 | 9.65648×10^{-3}	49 | 9.65615×10^{-3}	37 | 9.65648×10^{-3}	19 | 9.65634×10^{-3}	16 | 9.65648×10^{-3}
	0,2,5,9,15,25,44	54	0,1,3,5,8,14,27	27	0,2	0,1,3,4
	.9807974	.9815065	.9813915	.9815068	.9292589	.9846154
(6)	251 | 9.08270×10^{-4}	106 | 9.07282×10^{-4}	127 | 9.08025×10^{-4}	53 | 9.07282×10^{-4}	41 | 9.07253×10^{-4}	31 | 9.07281×10^{-4}
	0,2,5,8,12,18,26,36,95	36	0,1,3,5,8,12,17,38	18	0,2,4,7	0,1,3,5,8
	.9973798	.9971742	.9973711	.9971745	.9967388	.9963288

Table 1. Test Problems Over the Unit Square With h = 1/40

Region	J-SI	CJ-CG	RS-SI	RS-CG	SSOR-SI	SSOR-CG
(1)	120 | 1.61596×10^{-3}	58 | 1.61685×10^{-3}	57 | 1.61597×10^{-3}	29 | 1.61685×10^{-3}	22 | 1.61663×10^{-3}	16 | 1.61685×10^{-3}
	0,3,10,98	16	0,2,7	8	0	0,1
	.98728702	.9872049	.9867997	.9872053	.9800064	.9792514
(2)	90 | 8.47542×10^{-4}	52 | 8.48297×10^{-4}	46 | 8.47410×10^{-4}	26 | 8.48298×10^{-4}	22 | 8.48228×10^{-4}	17 | 8.48279×10^{-4}
	0,2,4,7,12,23	22	0,1,2,3,5,8,23	11	0,2,7	0,1,3,6
	.9787733	.9791823	.9792092	.9791827	.9784321	.9701817
(3)	76 | 3.61644×10^{-4}	50 | 3.62336×10^{-4}	42 | 3.61739×10^{-4}	25 | 3.62335×10^{-4}	21 | 3.62428×10^{-4}	15 | 3.62306×10^{-4}
	0,2,4,7,12,37	22	0,1,2,4,7,31	11	0,2,12	0,1,2,4
	.9709087	.9709838	.9710263	.9709836	.9701155	.9594277
(4)	62 | 1.04289×10^{-4}	40 | 1.04816×10^{-4}	32 | 1.04207×10^{-4}	20 | 1.04815×10^{-4}	16 | 1.04515×10^{-4}	14 | 1.04836×10^{-4}
	0,2,5,9,16	22	0,1,3,6,17	11	0,2	0,1,3,6
	.9562094	.9572589	.9571362	.9572589	.9324005	.9384010
(5)	97 | 1.09554×10^{-3}	60 | 1.09662×10^{-3}	51 | 1.09806×10^{-3}	30 | 1.09662×10^{-3}	20 | 1.09652×10^{-3}	18 | 1.09663×10^{-3}
	0,2,4,6,8,10,14,26	22	0,1,2,3,4,6,10	11	0,1,2	0,1,2,4
	.9825412	.9826603	.9822498	.9826603	.9791120	.9730194
(6)	41 | 1.99813×10^{-4}	30 | 2.00549×10^{-4}	21 | 2.00224×10^{-4}	15 | 2.00549×10^{-4}	12 | 2.00316×10^{-4}	10 | 2.005189×10^{-4}
	0,2,5,11	20	0,1,3,9	10	0,2	0,1,3
	.9068176	.9093197	.90877104	.9093297	.8958033	.8558511

Table 2. Test Problem (2) With h = 1/20.

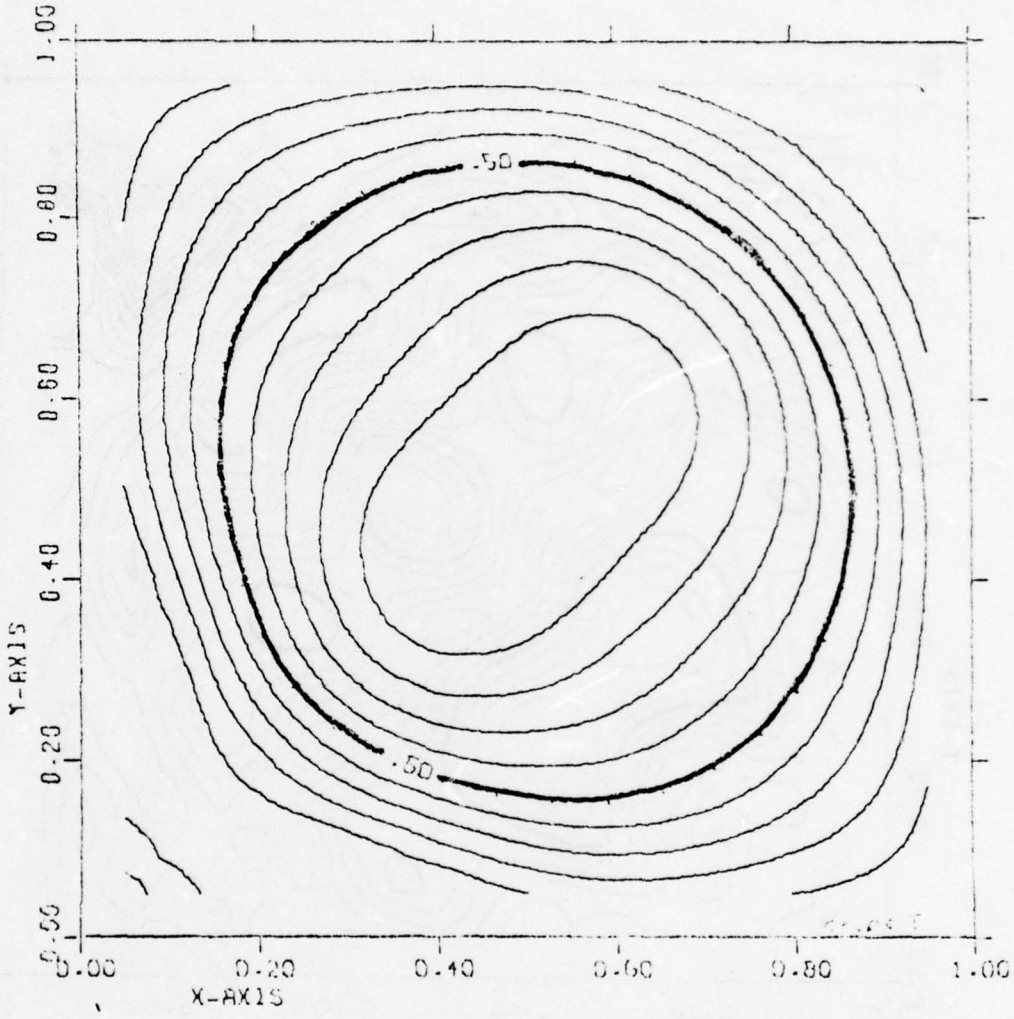


Figure 1. SSOR-SI method error distribution at convergence (SCALE = 7.529572×10^{-8})

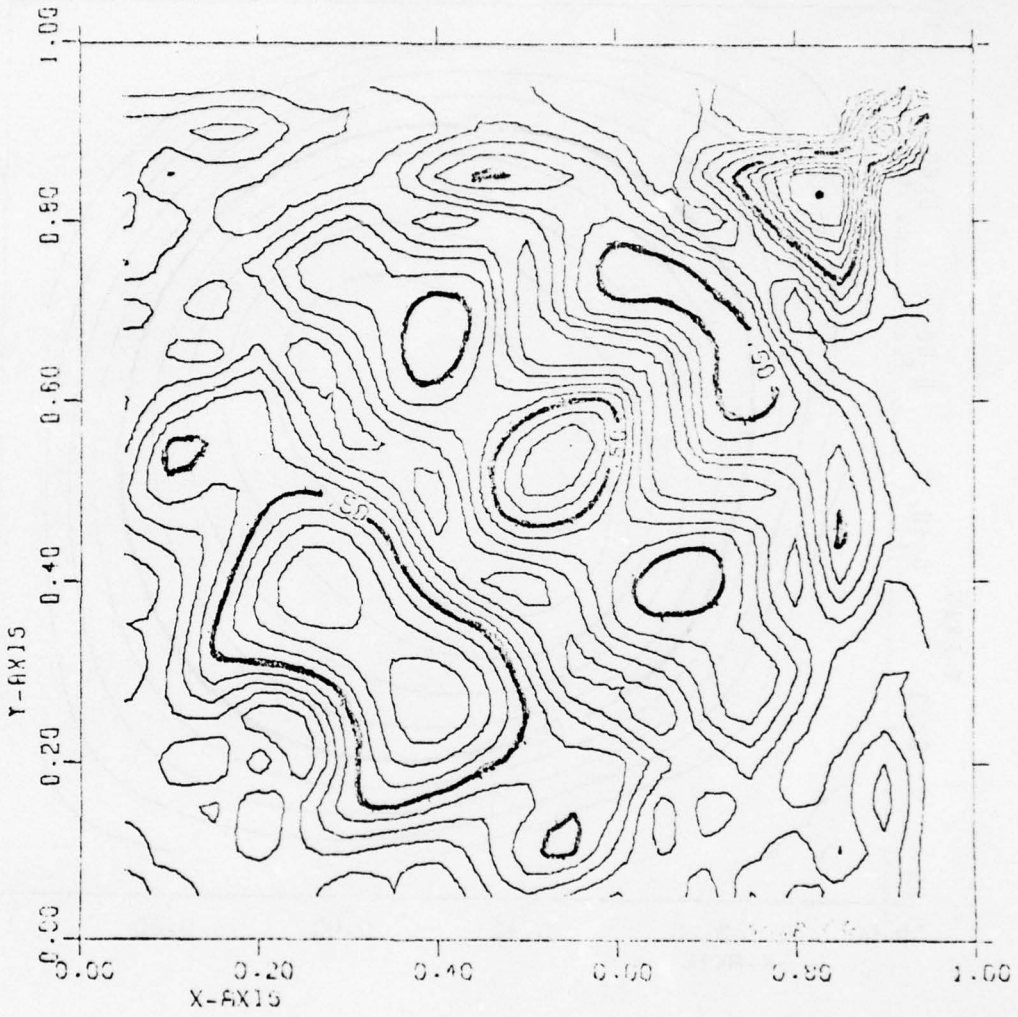


Figure 2. SSOR-CG method error distribution at convergence (SCALE = 1.583368×10^{-8})

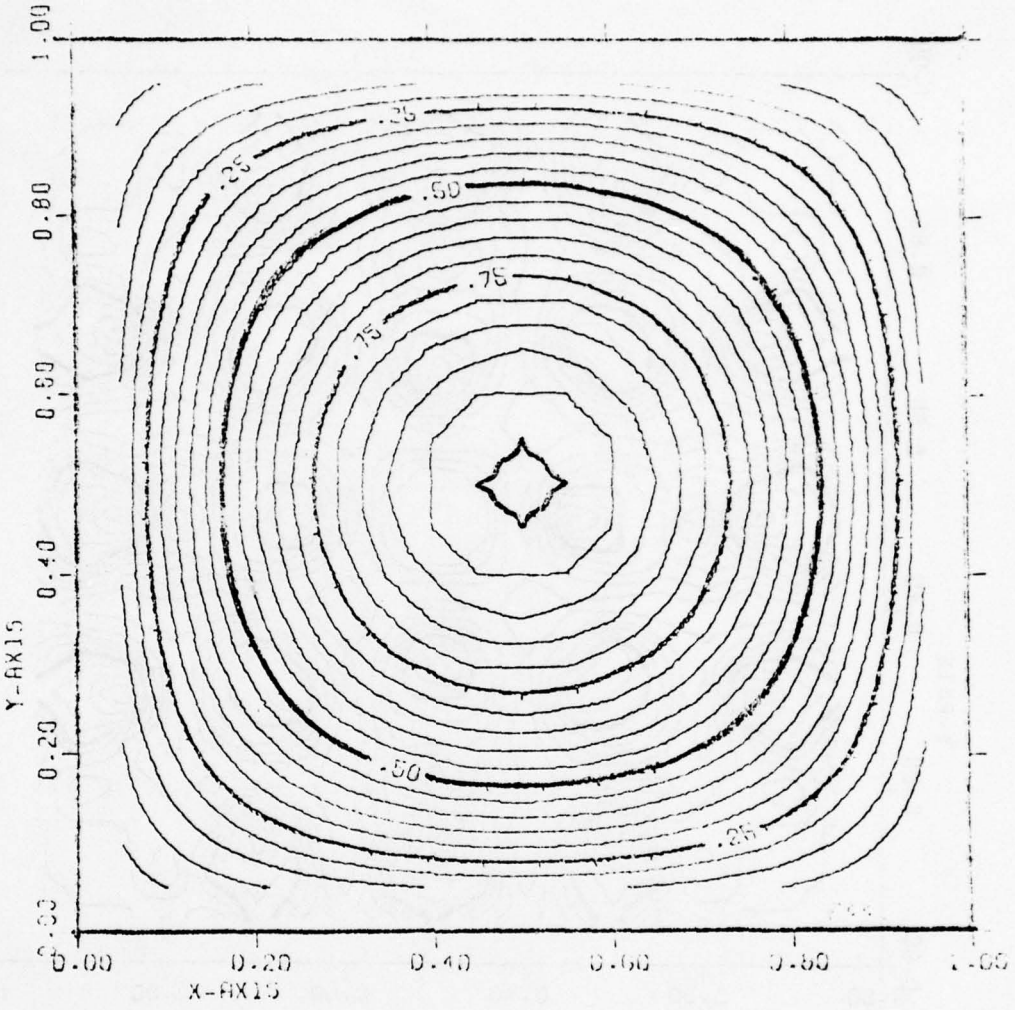


Figure 3. RS-SI method error distribution at convergence (SCALE = 2163014×10^{-7})

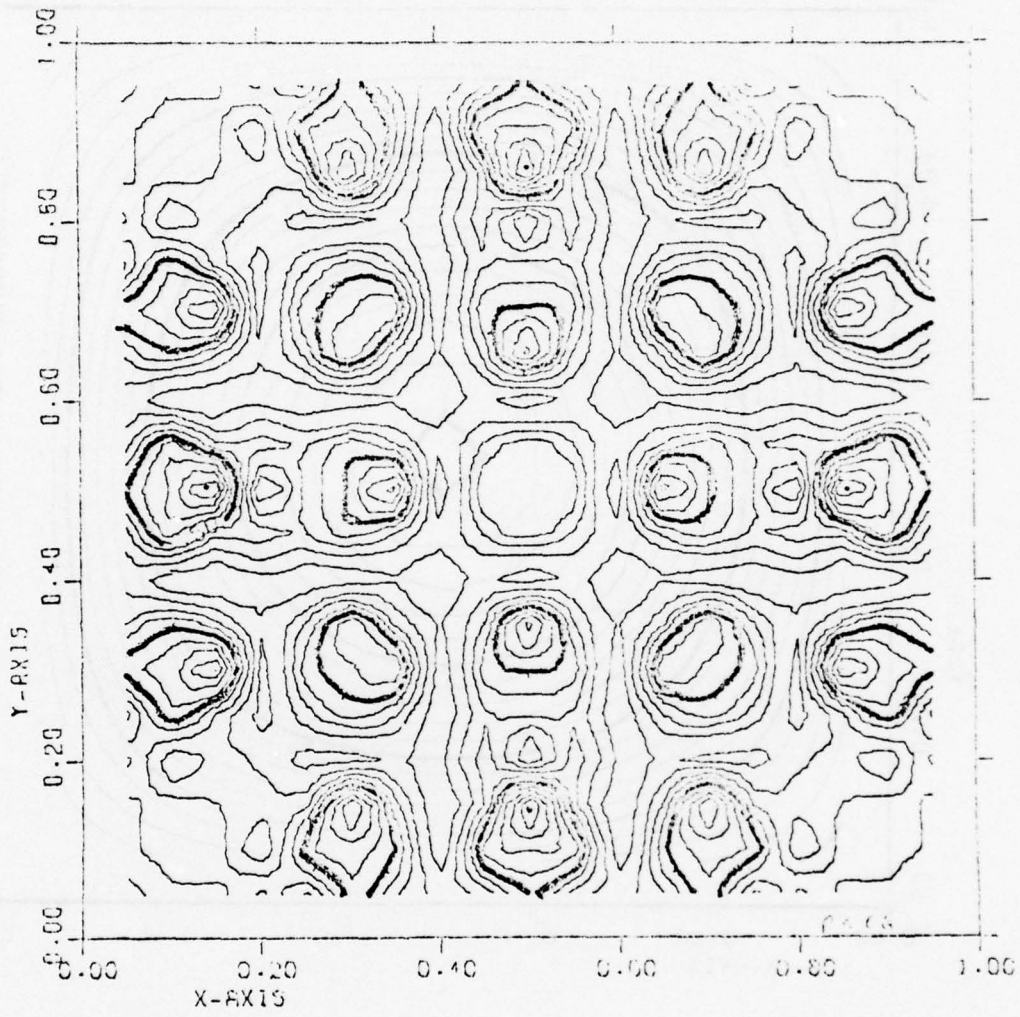


Figure 4. RS-CG method error distribution after 5 iterations (SCALE = 3.792600×10^{-3})

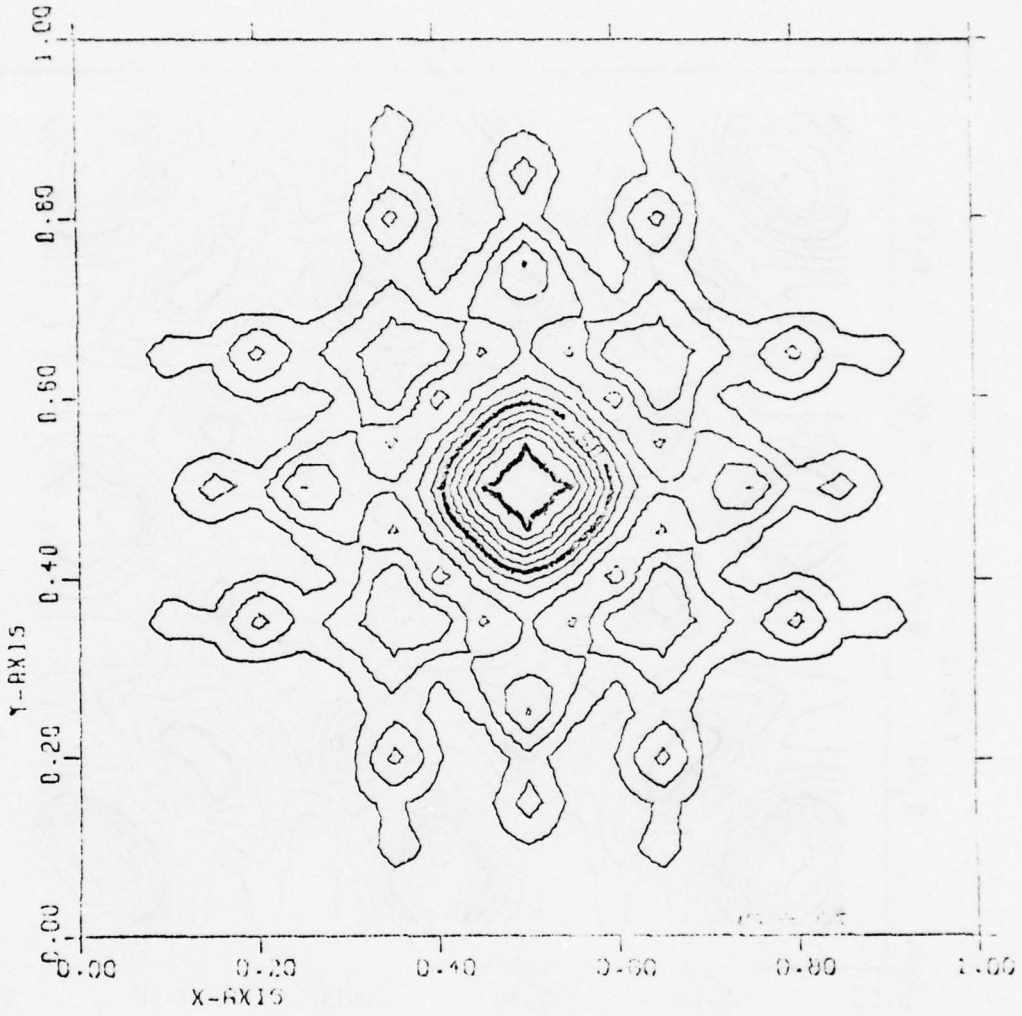


Figure 5. RS-CG method error distribution after 5 iterations (SCALE = 3.792600×10^{-3})

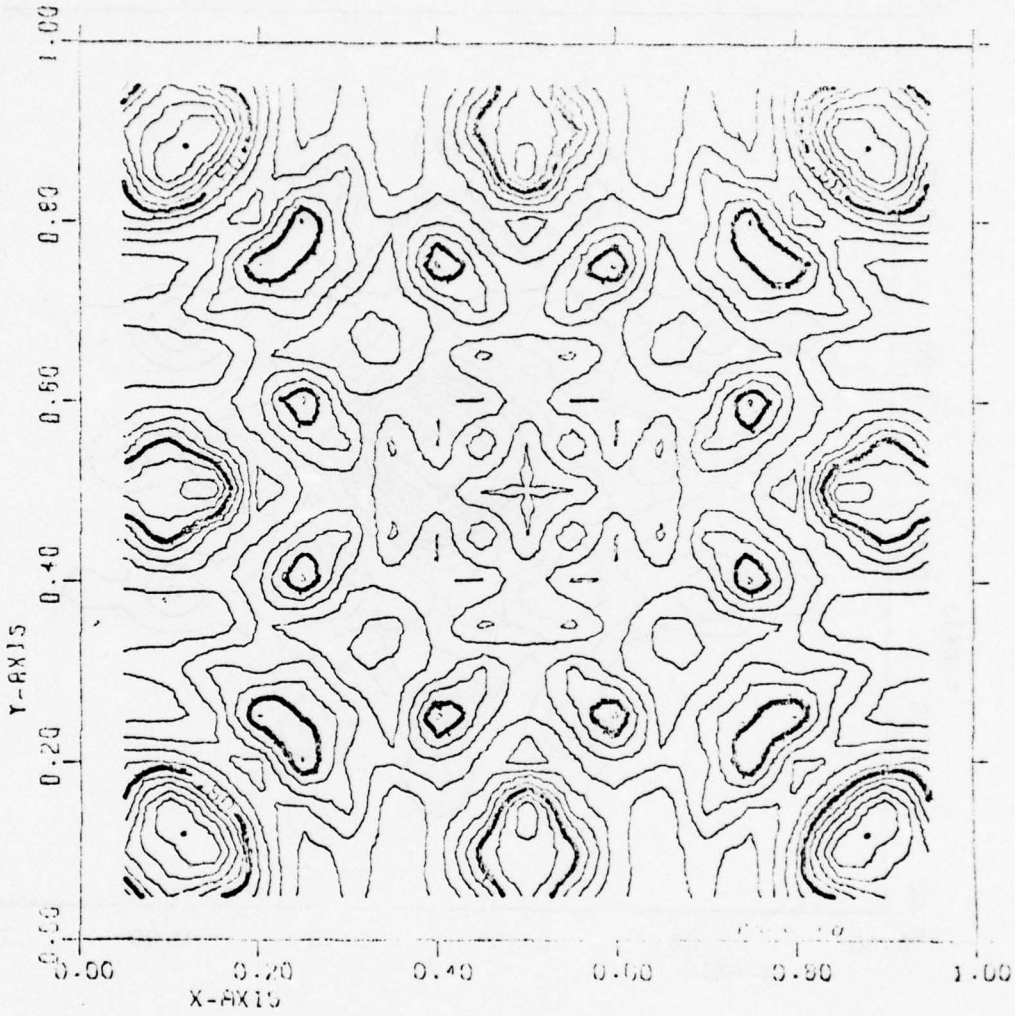


Figure 6. RS-CG method error distribution after 10 iterations (SCALE = 2.275237×10^{-6})

References

1. Hageman, L. A., and D. M. Young, "On the Effective Use of Iterative Methods for Solving Large Linear Systems," a monograph, in preparation.

2. Hayes, Linda J., and D. M. Young, "The Accelerated SSOR Method for Solving Large Linear Systems: Preliminary Report," CNA-123, Center for Numerical Analysis, UT Austin, May 1977.

3. Kincaid, David R., and D. M. Young, "The Development of a Computer Package for Solving a Class of Partial Differential Equations by Iterative Methods," Annales de l'Association internationale pour le calcul analogique-N*3-Juillet 1975, pp. 186-191.

4. Mouradoglou, Alkis J., and John H. Dauwalder, "REGION Specification Routine," V2 UTEX REGION, UTV2-01-CC026, Computation Center, UT Austin, April 1967.

5. Mouradoglou, Alkis J., "Numerical Studies on the Convergence of the Peaceman-Rachford Alternating Direction Implicit Method," TNN-67, Computation Center, UT Austin, June 1967.

6. Rice, John R.,"ELLPACK, A Cooperative Effort for the Study of Numerical Methods for Elliptic Partial Differential Equations," Computer Science Department Report CSD-TR 203, Purdue University, October 15, 1976.

7. _____,"ELLPACK Contributor's Guide--Initial Version," Computer Science Department Report CSD-TR 208, Purdue University, November 1, 1976.

8. _____, "ELLPACK 77 User's Guide--Initial Version," Computer Science Department Report CSD-TR 226, Purdue University, March 18, 1977.

9. Young, David M., M393D/CS393D--Selected Topics in Numerical Analysis, UT Austin, Fall, 1976:

   (a) "Adaptive Procedures for Chebyshev Semi-iteration," Supplement D.1, October 21, 1976,

   (b) "Conjugate Gradient Acceleration," Supplement F.1, November 3, 1976,

   (c) "The CCSI and RF-SI Method for Red/Black Matrices," Supplement G.1, November 12, 1976,

   (d) "The Compressed J-CG Method and the RS-CG Method for Red/Black Matrices," Supplement G.2, November 17, 1976.

Appendix 1

Description of Adaptive Procedures

(Equations, Flow Chart, and Algorithm)

Precise descriptions of the following six adaptive iterative algorithms are stated in this appendix.

I. Jacobi Semi-iteration (J-SI)
II. Compressed Jacobi Conjugate Gradient (CJ-CG)
III. Reduced System Semi-iteration (RS-SI)
IV. Reduced System Conjugate Gradient (RS-CG)
V. Symmetric Successive Overrelaxation Semi-iteration (SSOR-SI)
VI. Symmetric Successive Overrelaxation Conjugate Gradient (SSOR-CG)

For each method, a list of equations, a flow chart, and an algorithmic description is given. The latter description details exactly the adaptive procedure used in the ITPACK code. The mathematical derivation for each of these methods can be found in *Hageman and Young* [1].

I. J-SI: Jacobi Semi-iterative Equations

(1) Adaptive Parameters

$$\gamma = 2/(2-M_E-m_E), \qquad \sigma_E = (M_E-m_E)/(2-M_E-m_E),$$

$$r = \{1 - [1-\sigma_E^2]^{\frac{1}{2}}\}/\{1 + [1-\sigma_E^2]^{\frac{1}{2}}\}$$

(2) Acceleration Parameters

$$\rho_{n+1} = \begin{cases} 1/[1-\sigma_E^2/2], & n = s+1 \\ 1/[1-(\sigma_E/2)^2\rho_n], & n > s+1 \end{cases}$$

(3) Residual Vector

$$\delta^{(n)} = \begin{cases} \gamma\delta^{(n)} + (1-\gamma)\,\delta^{(n-1)}, & n = s+1 \\ B*u^{(n)} + c - u^{(n)}, & n > s+1 \end{cases}$$

(4) Iteration Vector

$$u^{(s+1)} = \gamma\delta^{(s)} + u^{(s)}, \qquad n = s$$

$$u^{(n+1)} = \rho_{n+1}\{\gamma\delta^{(n)} + u^{(n)}\} + (1-\rho_{n+1})u^{(n-1)}, \qquad n \geq s+1$$

(5) Stopping Test

$$d(n) = \delta^{(n)^T} * D * \delta^{(n)}$$

$$STEST = [1/(1-M_E)]\,[d(n)/(u^{(n)^T} * D * u^{(n)})]^{1/2}$$

If STEST $< \zeta$, then exit.

6. Changing Parameter Test

$$QA = [d(n)/d(s)]^{\frac{1}{2}}$$

$$QT = 2r^{(n-s)/2}/(1+r^{(n-s)})$$

If $QA \geq QT^F$, then change parameters.

7. Rayleigh Quotient Vector

$$\tilde{u}^{(n+1)} = u^{(n)} + \delta^{(n)}$$

$$\tilde{\delta}^{(n+1)} = B*\tilde{u}^{(n+1)} + c - \tilde{u}^{(n+1)}$$

8. Computing new M_E and m_E

$$Z = (1+r^{(n-s)})\{QA + [QA^2-QT^2]^{1/2}\}/2$$

$$X = Z^{1/(n-s)}$$

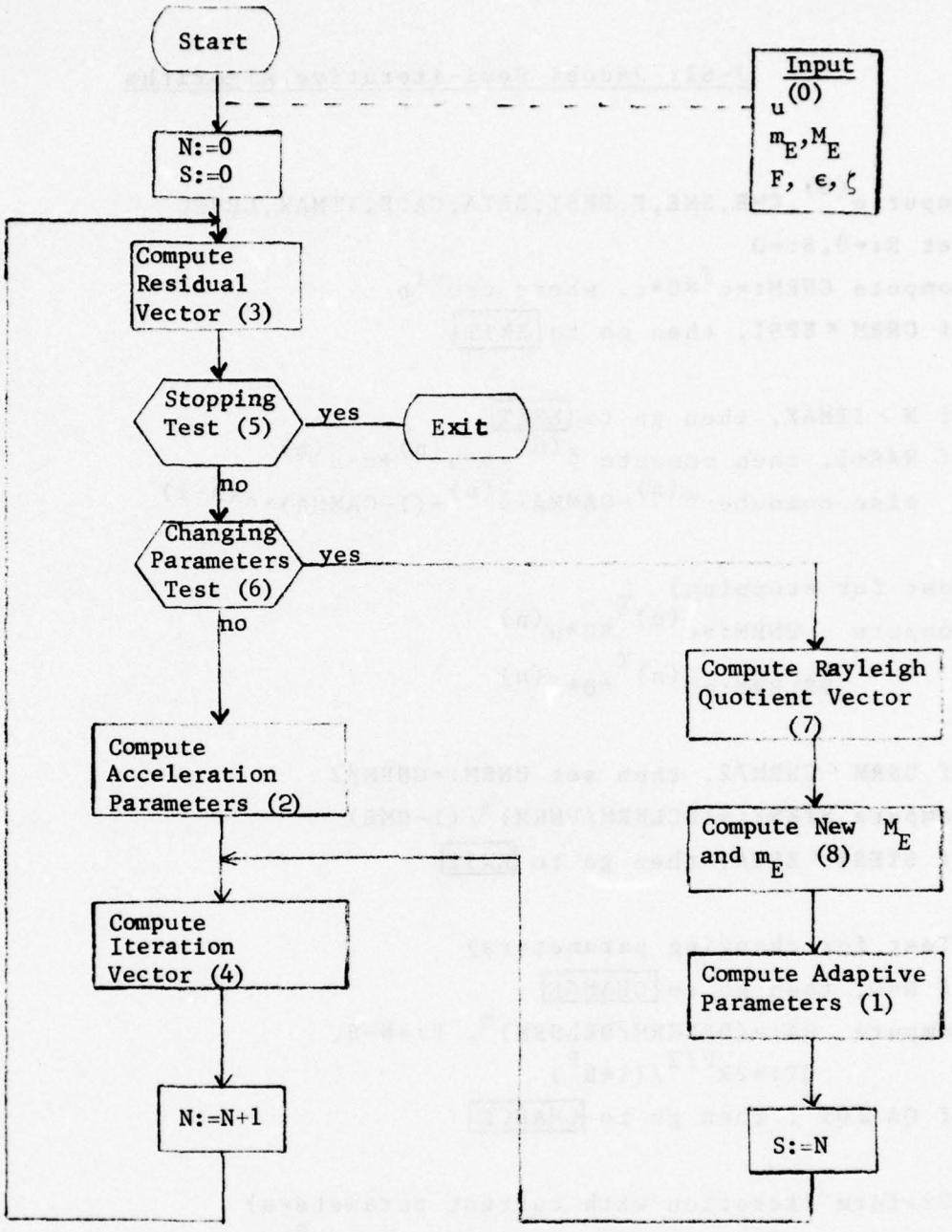
$$\sigma = (X+r/X)/(1+r)$$

$$M_1 = \begin{cases} M_E, & \text{if } n = 0 \\ [M_E+m_E+\sigma(2-M_E-m_E)]/2, & \text{otherwise} \end{cases}$$

$$M_2 = \begin{cases} (\delta^{(n)^T}*D*\tilde{\delta}^{(n+1)})/d(n), & \text{if case I} \\ (\tilde{\delta}^{(n+1)}*D*\tilde{\delta}^{(n+1)})/d(n), & \text{if case II} \end{cases}$$

$$M_E = \max\{M_1, M_2\}$$

$$M_E = \begin{cases} \text{not changed, if case I} \\ -M_E, & \text{if case II} \end{cases}$$



Flow Chart 1: J-SI Method

J-SI: Jacobi Semi-iterative Algorithm

Input $u^{(0)}$, CME, SME, F, EPSI, ZETA, CASE, ITMAX, LEVEL
Set N:=0, S:=0
Compute CNRM:=c^T*D*c, where $c=D^{-1}b$
If CNRM < EPSI, then go to EXIT

START If N > ITMAX, then go to EXIT
If N≠S+1, then compute $\delta^{(n)}=B*u^{(n)}+c-u^{(n)}$
else compute $\delta^{(n)}=GAMMA\cdot\tilde{\delta}^{(n)}+(1-GAMMA)\cdot\delta^{(n-1)}$

(Test for stopping)
Compute UNRM:=$u^{(n)^T}*D*u^{(n)}$
DELNRM:=$\delta^{(n)^T}*D*\delta^{(n)}$

If UNRM < CNRM/2, then set UNRM:=CNRM/2
Compute STEST:=$(DELNRM/UNRM)^{\frac{1}{2}}/(1-CME)$
If STEST < ZETA, then go to EXIT

(Test for changing parameters)
If N=0, then go to CHANGE
Compute QA:=$(DELNRM/DELSRM)^{\frac{1}{2}}$, P:=N-S,
QT:=$2R^{P/2}/(1+R^P)$
If QA ≥ QT^F, then go to CHANGE

(Preform iteration with current parameters)
If N=S+1, then compute RHO:=$1.0/(1-SIGE^2/2)$
else compute RHO:=$1.0/(1-RHO\cdot SIGE^2/4)$
Compute C1:=RHO·GAMMA, C2:RHO, C3=1-RHO,
$u^{(n+1)}=C1\cdot\delta^{(n)}+C2\cdot u^{(n)}+C3\cdot u^{(n-1)}$
Go to ENDIT

J-SI (continued)

CHANGE (Change parameters)

Compute $\tilde{u}^{(n+1)} = u^{(n)} + \delta^{(n)}$

$\tilde{\delta}^{(n+1)} = B * \tilde{u}^{(n+1)} + c - \tilde{u}^{(n+1)}$

If N=0, then set ZM1:=CME,

else compute $Z := (1+R^P)(QA + (QA^2 - QT^2)^{\frac{1}{2}})/2$,

$X := Z^{1/P}$

$SIG := (X + R/X)/(1+R)$

$ZM1 := (CME + SME + SIG \cdot (2 - CME - SME))/2$

If CASE=.TRUE., then compute $ZM2 := \delta^{(n)^T} * D * \tilde{\delta}^{(n+1)} / DELNRM$

else compute $ZM2 := \tilde{\delta}^{(n+1)^T} * D * \tilde{\delta}^{(n+1)} / DELNRM$

Set CME:=max{ZM1,ZM2}

If CASE=.FALSE., then set SME:=-CME

Compute SIGE:=(CME-SME)/(2-CME-SME)

GAMMA:=2/(2-CME-SME)

$R := (1 - (1 - SIGE^2)^{\frac{1}{2}}) / (1 + (1 + SIGE^2)^{\frac{1}{2}})$

Set S:=N

DELSRM:=DELNRM

RHO:=1

Print N,ZM1,ZM2,CME

Compute $u^{(n+1)} = GAMMA \cdot \delta^{(n)} + u^{(n)}$

ENDIT (End of iteration step)

Print N, $UNRM^{\frac{1}{2}}$, STEST, QA, QT^F, CME, RHO, GAMMA

Set N:=N+1

Go to START

EXIT (Exit iteration algorithm)

Compute $UNRM := u^{(n)^T} * D * u^{(n)}$

Print N, $UNRM^{\frac{1}{2}}$

If LEVEL > 2, print $u^{(n)}$

END

II. CJ-CG: Compressed Conjugate Gradient Equations

(1) Residual Vector (non-recursive computation)

$$\begin{cases} u_R^{(n)} = F_R u_B^{(n)} + c_R \\ \delta_B^{(n)} = F_B u_R^{(n)} + c_B - u_B^{(n)} \end{cases}$$

(2) Acceleration Parameters

$$\delta_R^{(n+1)} = \begin{cases} F_R \delta_B^{(0)}, & n = 0 \\ \rho_{n+1} F_R \delta_B^{(n)} + (1-\rho_{n+1}) \delta_R^{(n-1)}, & n > 0 \end{cases}$$

$$d_R(n) = \delta_R^{(n)\,T} * D_R * \delta_R^{(n)}, \qquad d_B(n) = \delta_B^{(n)\,T} * D_B * \delta_B^{(n)}$$

$$\rho_{n+1} = \begin{cases} 1, & n = 0 \\ 1 \Big/ \left[1 - (1/\rho_n)(d_B(n)/d_R(n-1))\right], & n > 0 \end{cases}$$

$$\rho_{n+2} = 1 \Big/ \left[1 - (1/\rho_{n+1})(d_R^{(n+1)}/d_B(n))\right]$$

$$\hat{\rho} = (\rho_{n+2}/\rho_n)(1-\rho_{n+1})(1-\rho_n)$$

$$\hat{\gamma} = (\rho_{n+2}\rho_{n+1}/\hat{\rho})$$

(3) Iteration Vector

$$u_B^{(n+2)} = \begin{cases} \hat{\gamma}\delta_B^{(0)} + u_B^{(0)}, & n = 0 \\ \hat{\rho}\{\hat{\gamma}\delta_B^{(n)} + u_B^{(n)}\} + (1-\hat{\rho}) u_B^{(n-2)}, & n > 0 \end{cases}$$

(4) Residual Vector (recursive computation)

$$\delta_B^{(n+2)} = \rho_{n+2} F_B * \delta_R^{(n+1)} + (1-\rho_{n+2}) \delta_B^{(n)}$$

(5) Stopping Test

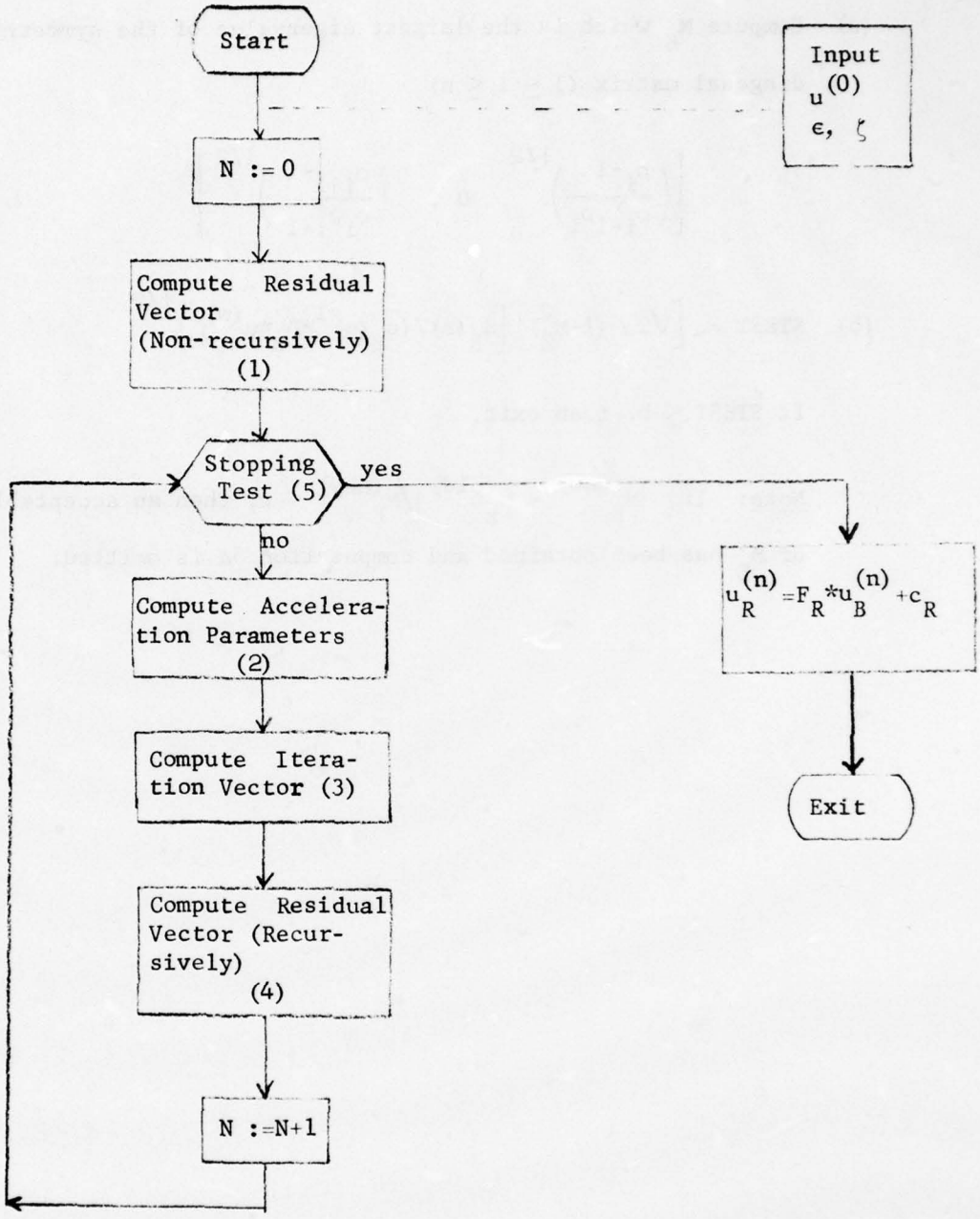
(a) Compute M_E which is the largest eigenvalue of the symmetric n x n tri-diagonal matrix ($1 \leq i \leq n$)

$$\left[\left(\frac{\rho_i - 1}{\rho_{i-1}\rho_i}\right)^{1/2}, \quad 0, \quad \left(\frac{\rho_{i+1} - 1}{\rho_i \rho_{i+1}}\right)^{1/2}\right]$$

(b) $$\text{STEST} = \left[\sqrt{2} / (1 - M_E^2)\right]\left[d_B(n) / (u_B(n)^T * D_B * u(n))\right]^{1/2}$$

If STEST $\leq \delta$, then exit.

Note: If $|M_E^{(new)} - M_E^{(old)}| / M_E^{(new)} < \epsilon$, then an acceptable estimate of M_E has been obtained and computation 5a is omitted.



Flow Chart 2: CJ-CG Method

CJ-CG: Compressed Jacobi Conjugate Gradient Algorithm

Input $u^{(0)}$, EPSI, ZETA, ITMAX, LEVEL

Set N:=0

Compute CNRM:=$k_B^T * D_B * k_B$

If CNRM < EPSI, then go to [EXIT]

[START] If N > ITMAX, then go to [EXIT]

If N < 4, then go to [ONE]

If $|CME - CMOLD|/CME < EPSI$, then go to [TWO]

[ONE] (Determine new CME)

Set CMOLD:=CME

If N=0, then set CME:=0

Else set CME:=maximum eigenvalue of the tri-diagonal matrix

$$\{[(RHO_i-1)/(RHO_i RHO_{i-1})]^{\frac{1}{2}}, 0, [(RHO_{i+1}-1)/(RHO_{i+1} RHO_i)]^{\frac{1}{2}}\},$$

for $1 \le i \le N$

[TWO] (Test for stopping)

Compute $UNRM := u_B^{(N)T} * D_B * u_B^{(N)}$

$DELNRM := \delta_B^{(N)} * D_B * \delta_B^{(N)}$

If UNRM < CNRM, then set UNRM:=CNRM

Compute $STEST := (2 \cdot DELNRM/UNRM)^{\frac{1}{2}}/(1-CME^2)$

If STEST < ZETA, then go to [EXIT]

If N=0, then set $RHO_{N+1} := 1$

else compute $RHO_{N+1} := 1/(1-DELNRM/(DELSRM \cdot RHO_N))$

Set $C1 := RHO_{N+1}$, $C1 := 1-RHO_{N+1}$

Compute: $v_R = F_R^* u_B^{(N)}$

$\delta_R^{(N+1)} = C1 \cdot v_R + C2 \cdot \delta_R^{(N-1)}$

Compute $DELSRM := \delta_R^{(N+1)T} * D_R * \delta_R^{(N+1)}$

$RHO_{N+2} := 1/(1-DELSRM/(DELNRM \cdot RHO_{N+1}))$

CJ-CG (continued)

If N=0, then set RHOHAT:=1,

else compute RHOHAT:=1+$RHO_{N+2}(1-RHO_{N+1})\cdot(1-RHO_N)/RHO_N$

Compute GAMMA:=$RHO_{N+2}\cdot RHO_{N+1}/RHOHAT$

Set C1:=RHOHAT·GAMMA, C2:=RHOHAT, C3:=1-RHOHAT

Compute $u_B^{(N+2)}=C1\cdot\delta_B^{(N)}+C2\cdot u_B^{(N)}+C3\cdot u_B^{(N-2)}$

Compute $v_B=F_B\delta_R^{(N+1)}$

Set C1:=RHO_{N+2}, C2:=1-RHO_{N+2}

Compute $\delta_B^{(N+2)}=C1\cdot v_B+C2\cdot\delta_B^{(N)}$

Print N, $UNRM^{\frac{1}{2}}$, STEST, CME, RHO, GAMMA

Set N=N+2

Go to START

EXIT Compute $u_R^{(N)}=F_R^{*}u_B^{(N)}+C_R$

$UNRM:=u^{(N)^T}*D*u^{(N)}$

Print N, $UNRM^{\frac{1}{2}}$

IF LEVEL > 2, then print $u^{(N)}$

END

III. RS-SI: Reduced System Semi-iterative Equations

(1) Adaptive Parameters

$$\gamma = 2/(2-M_E^2), \qquad \sigma_E = M_E^2/(2-M_E^2),$$

$$r = \{1 - [1-M_E^2]^{\frac{1}{2}}\}/\{1+[1-M_E^2]^{\frac{1}{2}}\}$$

(2) Acceleration Parameters

$$\rho_{n+1} = \begin{cases} 1/[1-\sigma_E^2/2], & n = s+1 \\ 1/[1-(\sigma_E/2)^2\rho_n], & n > s+1 \end{cases}$$

(3) Residual Vector

$$\begin{cases} u_R^{(n)} = F_R u_B^{(n)} + c_R \\ \delta_B^{(n)} = F_B u_R^{(n)} + c_B - u_B^{(n)} \end{cases}$$

(4) Iteration Vector

$$u_B^{(s+1)} = \gamma\delta_B^{(s)} + u_B^{(s)}$$

$$u_B^{(n+1)} = \rho_{n+1}\{\gamma\delta_B^{(n)} + u_B^{(n)}\} + (1-\rho_{n+1})u_B^{(n-1)}, \qquad n \geq s+1$$

(5) Stopping Test

$$d_B(n) = \delta_B^{(n)^T} * D_B * \delta_B^{(n)}$$

$$STEST = [\sqrt{2}/(1-M_E^2)][d_B(n)/(u_B(n)^T * D_B * u_B(n)]^{1/2}$$

If STEST $< \zeta$, then exit.

RS-SI (continued)

(6) Changing Parameters Test

$$QA = [d_B(n)/d_B(s)]^{1/2}$$

$$QT = 2r^{(n-s)}/(1+r^{2(n-s)})$$

If $QA \geq QT^F$, then change parameters.

(7) Computing new M_E

$$Z = (1+r^{2(n-s)})\{QA + [QA^2-QT^2]^{1/2}\}/2$$

$$X = Z^{1/(2(n-s))}$$

$$\sigma = (X+r/X)/(1+r)$$

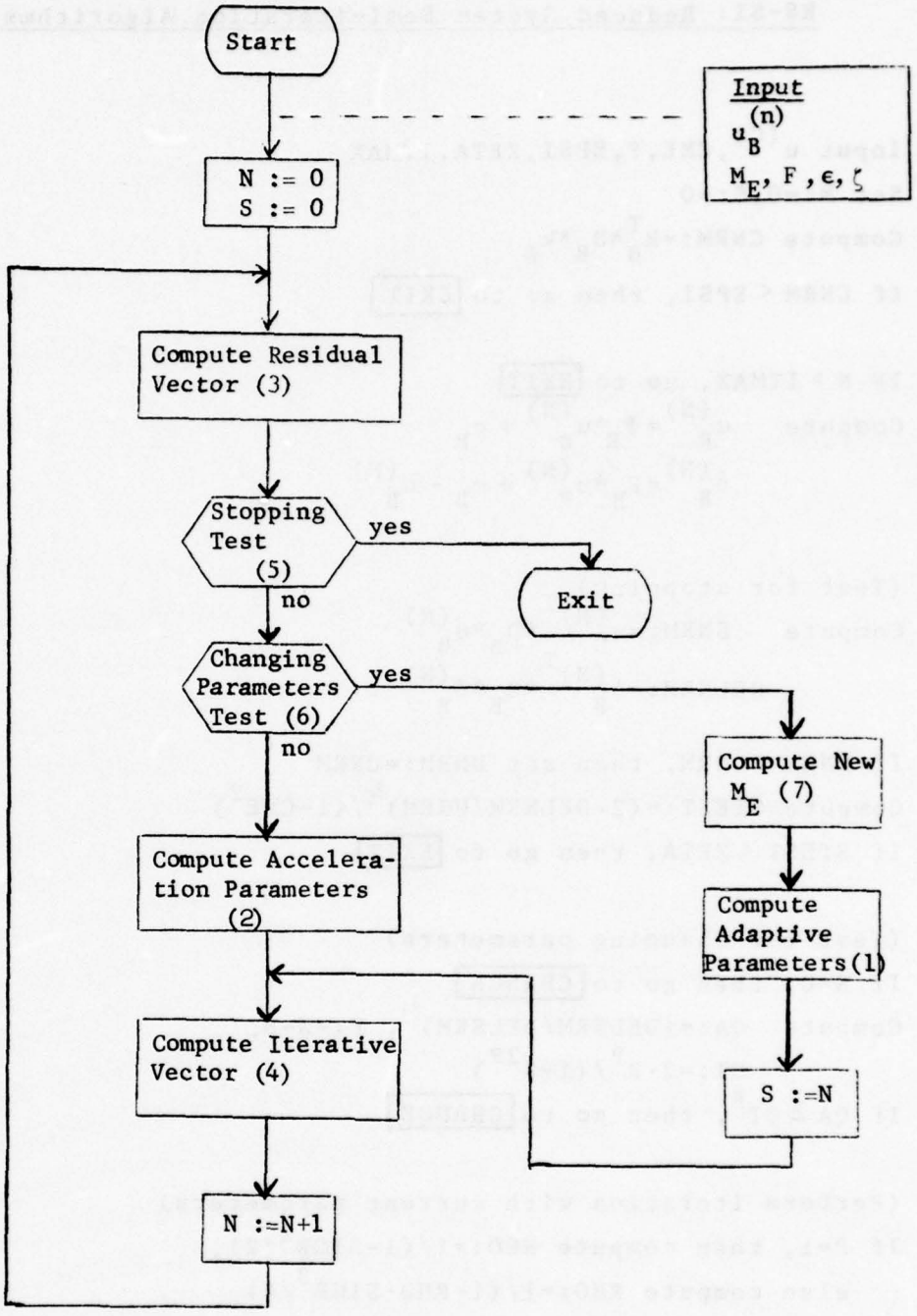
$$M_1 = \begin{cases} M_E, & \text{if } n = 0 \\ \sigma, & \text{otherwise} \end{cases}$$

$$\delta_R^{(n)} = F_R * \delta_B^{(n)}$$

$$d_R(n) = \delta_R^{(n)T} * D_R * \delta_R^{(n)},$$

$$M_2 = \left[d_R(n)/d_B(n)\right]^{1/2}$$

$$M_E = \max\{M_E, M_1, M_2\}$$



Flow Chart 3: RS-SI Method

RS-SI: Reduced System Semi-iteration Algorithms

Input $u^{(0)}$, CME, F, EPSI, ZETA, ITMAX

Set N:=0, S:=0

Compute CNRM:=$k_B^T * D_B * k_B$

If CNRM < EPSI, then go to **EXIT**

START IF N > ITMAX, go to **EXIT**

Compute $u_R^{(N)} = F_R * u_B^{(N)} + c_R$

$\delta_B^{(N)} = F_B * u_R^{(N)} + c_B - u_B^{(N)}$

(Test for stopping)

Compute UNRM:=$u_B^{(N)^T} * D_B * u_B^{(N)}$

DELNRM:=$\delta_B^{(N)^T} * D_B * \delta_B^{(N)}$

If UNRM < CNRM, then set UNRM:=CNRM

Compute STEST:=$(2 \cdot \text{DELNRM}/\text{UNRM})^{\frac{1}{2}}/(1-\text{CME}^2)$

If STEST < ZETA, then go to **EXIT**

(Test for changing parameters)

If N=0, then go to **CHANGE**

Compute QA:=$(\text{DELNRM}/\text{DELSRM})^{\frac{1}{2}}$, P:=N-S,

QT:=$2 \cdot R^P/(1+R^{2P})$

If QA $\geq$ QT^F, then go to **CHANGE**

(Perform iteration with current parameters)

If P=1, then compute RHO:=$1/(1-\text{SIGE}^2/2)$,

else compute RHO:=$1/(1-\text{RHO} \cdot \text{SIGE}^2/4)$

Set C1:=RHO·GAMMA, C2:RHO, C3:=1-RHO

Compute $u_B^{(N+1)} = C1 \cdot \delta_B^{(N)} + C2 \cdot u_B^{(N)} + C3 \cdot u_B^{(N-1)}$

Go to **ENDIT**

RS-SI (continued)

CHANGE (Change parameters)

If N=0, then set ZM1:=CME,

else compute $Z:=(1+R^{2P})(QA+(QA^2-QT^2)^{\frac{1}{2}})/2$

$X:=Z^{1/(2P)}$

$ZM1:=(X+R/X)/(1+R)$

$v_R:=F_R^*\delta_B^{(N)}$

$ZM2:=(v_R^T*D_R*v_R/DELNRM)^{\frac{1}{2}}$

Compute CME:=max{ZM1,ZM2}

Compute $SIGE:=CME^2/(2-CME^2)$

$GAMMA:=2/(2-CME^2)$

$R:=(1-(1-CME^2)^{\frac{1}{2}})/(1+(1-CME^2)^{\frac{1}{2}})$

Set S:=N

DELSRM:=DELNRM

RHO:=1

Print N,ZM1,ZM2,CME

Compute $u^{(N+1)}=GAMMA\cdot\delta^{(N)}+u^{(N)}$

ENDIT Print $N,UNRM^{\frac{1}{2}},STEST,QA,QT^F,CME,RHO,GAMMA$

Set N:=N+1

GO to START

EXIT Compute $u_R^{(N)}:=F*u_B^{(N)}+c_R$

$UNRM:=u_R^{(N)T}*D_R*u_R^{(N)}$

PRINT $N,UNRM^{\frac{1}{2}}$

If LEVEL > 2, print $u^{(N)}$

END

IV. RS-CG: Reduced System Conjugate Gradient Equations

(1) Residual Vector (non-recursive computation)

$$\begin{cases} u_R^{(n)} = F_R * u_B^{(n)} + c_R \\ \delta_B^{(n)} = F_B * u_R^{(n)} + c_B - u_B^{(n)} \end{cases}$$

(2) Acceleration Parameters

$$\begin{cases} v_R = F_R * \delta_B^{(n)} \\ v_B = F_B * v_R \end{cases}$$

$$d_B(n) = \delta_B^{(n)\,T} * D_B * \delta_B^{(n)}$$

$$\gamma_{n+1} = 1/[1-(\delta_B^{(n)\,T} * D_B * v_B)/d_B(n)]$$

$$\rho_{n+1} = \begin{cases} 1 & , \quad n = 0 \\ 1/\left[1-\left(\frac{\gamma_{n+1}}{\gamma_n \rho_n}\right)\ (d_B(n)/d_B(n-1))\right] & , \quad n > 0 \end{cases}$$

(3) Iteration Vector

$$u_B^{(n+1)} = \begin{cases} \gamma_1 \delta_B^{(n)} + u_B^{(n)} & n = 0 \\ \rho_{n+1}\ \{\gamma_{n+1}\delta_B^{(n)} + u_B^{(n)}\} + (1-\rho_{n+1})u_B^{(n-1)}, & n > 0 \end{cases}$$

(4) Residual Vector (recursive computation)

$$\delta_B^{(n+1)} = \begin{cases} \gamma_1 v_B + (1-\gamma_1)\delta_B^{(0)} & , \quad n = 0 \\ \rho_{n+1}\ \{\gamma_{n+1} v_B + (1-\gamma_{n+1})\delta_B^{(n)}\} + (1-\rho_{n+1})\delta_B^{(n-1)}, & n > 0 \end{cases}$$

RS-CG (continued)

(5) Stopping Test

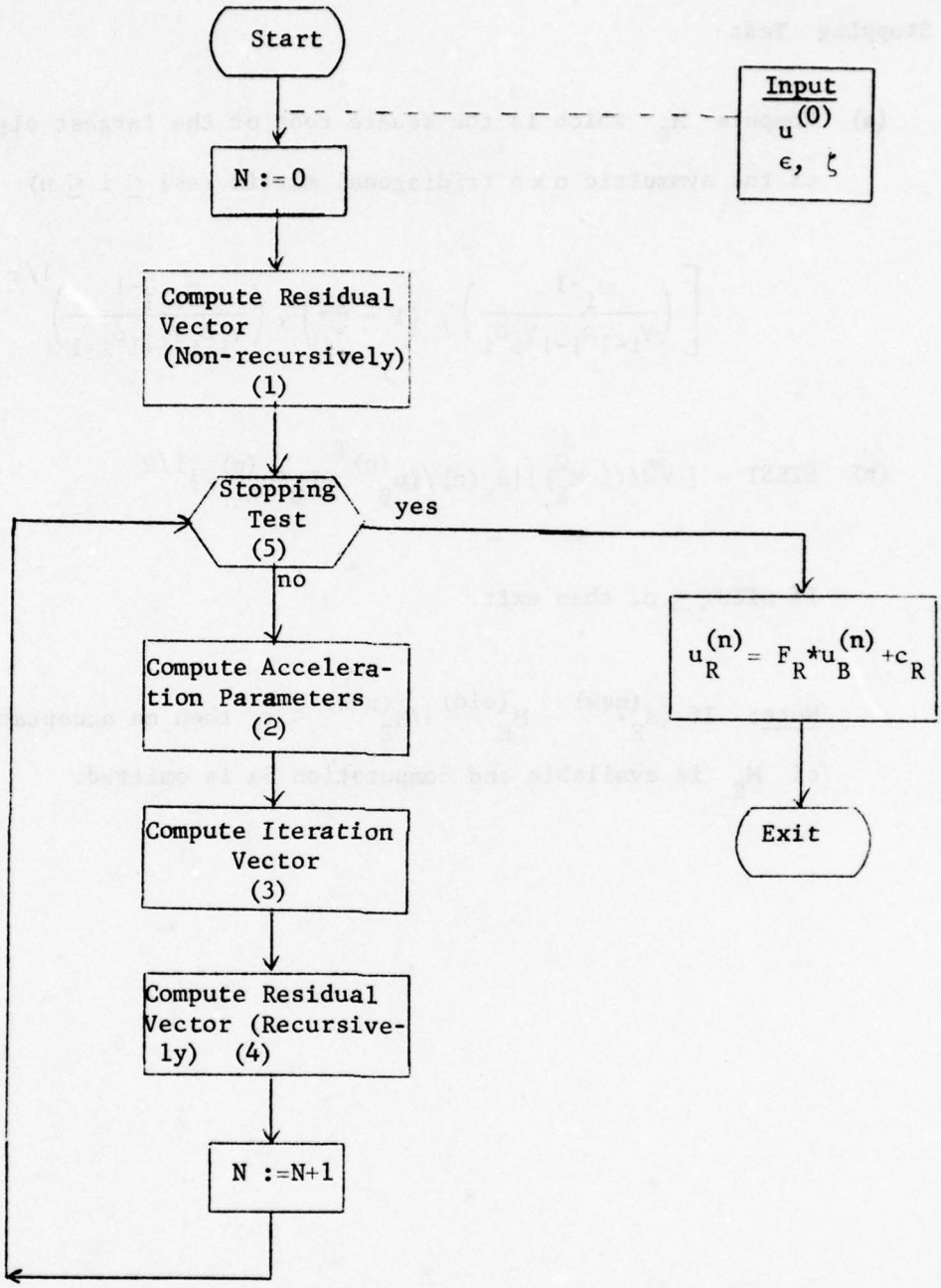
(a) Compute M_E which is the square root of the largest eigenvalue of the symmetric n x n tridiagonal matrix $(s+1 \leq i \leq n)$

$$\left[\left(\frac{\rho_i^{-1}}{\gamma_{i-1}\rho_{i-1}\gamma_i\rho_i} \right), \left(1 - \frac{1}{\gamma_i} \right), \left(\frac{\rho_{i+1}^{-1}}{\gamma_i\rho_i\gamma_{i+1}\rho_{i+1}} \right)^{1/2} \right]$$

(b) $STEST = [\sqrt{2}/(1-M_E^2)]\,[d_B(n)/(u_B^{(n)T} * D_B * u_B^{(n)})]^{1/2}$

If STEST $\leq \rho$, then exit.

Note: If $|M_E^{(new)} - M_E^{(old)}|/M_E^{(new)} < \epsilon$, then an acceptable estimate of M_E is available and computation 5a is omitted.



Flow Chart 4: RS-CG Method

RS-CG: Reduced System Conjugate Gradient Algorithm

Input $u^{(0)}$, EPSI, ZETA, ITMAX, LEVEL

Set N:=0

Compute CNRM:=$k_B^T * D * k_B$

If CNRM < EPSI, then go to [EXIT]

[START] If N > ITMAX, then go to [EXIT]

If N < 4, then go to [ONE]

If |CME-CMOLD|/CME < EPSI, then go to [TWO]

[ONE] (Determine new CME)

Set CMOLD:=CME

If N=0, then set CME:=0

else set CME=square root of the maximum eigenvalue of the tridiagonal matrix

$$\{[(RHO_i-1)/(GAMMA_{i-1}\cdot RHO_{i-1}\cdot GAMMA_i\cdot RHO_i)^{\frac{1}{2}}, (1-1/GAMMA_i),$$
$$[(RHO_{i+1}-1)/(GAMMA_i\cdot RHO_i\cdot GAMMA_{i+1}\cdot RHO_{i+1})]^{\frac{1}{2}}\}$$

[TWO] (Test for stopping)

Compute UNRM:=$u_B^{(N)^T} * D * u_B^{(N)}$

DELNRM:=$\delta_B^{(N)^T} * D * \delta_B^{(N)}$

If UNRM < CNRM, then set UNRM:=CNRM

Compute STEST:=$(2\cdot DELNRM/UNRM)^{\frac{1}{2}}/(1-CME^2)$

If STEST < ZETA, then go to EXIT

Compute: $v_B = F_B * F_R * \delta_B^{(N)}$

$GAMMA_{N+1}$:=$1/(1-\delta_B^{(N)^T} * D_B * v_B/DELNRM)$

If N=0, then RHO_1:=0

else compute RHO_{N+1}:=$1/(1-GAMMA_{N+1}DELNRM/(GAMMA_N DELSRM\cdot RHO_N))$

Set DELSRM:=DELNNM, C1:=$RHO_{N+1}\cdot GAMMA_{N+1}$, C2:=$RHO_{N+1}$, C3:=$1-RHO_{N+1}$,
C4:=$RHO_{N+1}(1-GAMMA_{N+1})$

Compute: $u_B^{(N+1)} = C1*\delta_B^{(N)} + C2*u_B^{(N)} + C3*u_B^{(N-1)}$

$\delta_B^{(N+1)} = C1*v_B + C4*\delta_B^{(N)} + C3*\delta_B^{(N-1)}$

Print N, $UNRM^{\frac{1}{2}}$, STEST, CME, RHO_{N+1}, $GAMMA_{N+1}$

Set N=N+1

Go to START

EXIT Compute $u_R^{(N)} = F*u_B^{(N)} + C_R$

$UNRM := u^{(N)^T} * D * u^{(N)}$

Print N, $UNRM^{\frac{1}{2}}$

If LEVEL > 2, then print $u^{(n)}$

END

V. SSOR-SI: Symmetric Successive Overrelaxation Semi-iterative Equations

(1) Adaptive Parameters

$$\text{SPECR} = S(\mathcal{B}_\omega), \qquad \gamma = 2/(2-\text{SPECR}), \qquad \sigma_E = \text{SPECR}/(2-\text{SPECR})$$

$$r = \{1 - [1-\text{SPECR}^2]^{1/2}\}/\{1 + [1-\text{SPECR}^2]^{1/2}\}$$

(2) Acceleration Parameters

$$\rho_{n+1} = \begin{cases} 1/[1 - \sigma_E^2/2], & n = s+1 \\ 1/[1-(\sigma_E/2)^2\rho_n], & n > s+1 \end{cases}$$

(3) Difference Vectors and Residual Vector

$$v = \mathcal{L}_\omega u^{(n)} + k_\omega^{(F)}$$

$$\Delta^{(n)} = v - u^{(n)}$$

$$\delta^{(n)} = \mathcal{Q}_\omega v + k_\omega^{(B)} - u^{(n)}$$

(4) Iteration Vector

$$u^{(n+1)} = \rho_{n+1}\{\gamma\delta^{(n)} + u^{(n)}\} + (1 - \rho_{n+1})u^{(n-1)}$$

(5) Stopping Test

$$d(n) = \Delta^{(n)^T} * D * \Delta^{(n)}$$

$$\text{STEST} = [(2-\omega)/(\omega(1-M_E)(1-\text{SPECR})^2)]^{1/2}[d(n)/(u^{(n)^T} * D * u^{(n)})]^{1/2}$$

If STEST $< \zeta$, then exit.

SSOR-SI (continued)

(6) Changing Parameter Test

$$QA = [d(n)/d(s)]^{1/2}$$

$$QT = 2r^{(n-s)/2}/(1+r^{(n-s)})$$

If $QA \geq QT^F$, then change parameters.

(7) Computing new S' and M_E

$$Z = (1+r^{(n-s)})\{QA + [QA^2-QT^2]^{1/2}\}/2$$

$$X = Z^{1/(n-s)}$$

$$\sigma = (X + r/X)/(1+r)$$

$$S_1 = \begin{cases} SPECR & , \text{ if } n = 0 \\ [SPECR + \sigma(2-SPECR)]/2, & \text{otherwise} \end{cases}$$

$$\tilde{u}^{(n+1)} = u^{(n)} + \delta^{(n)}$$

$$\tilde{\Delta}^{(n+1)} = \mathcal{L}_\omega \tilde{u}^{(n+1)} + k_\omega^{(F)} - \tilde{u}^{(n+1)}$$

$$S_2 = (\Delta^{(n)^T} * D * \tilde{\Delta}^{(n+1)})/d(n)$$

$$S' = \max\{SPECR, S_1, S_2\}$$

$$M_1 = [(1-S')(1+\bar{\beta}\omega^2) - \omega(2-\omega)]/[\omega(\omega-1-S')]$$

$$v = B*\delta^{(n)}, \qquad d(n) = \delta^{(n)^T} * D * \delta^{(n)},$$

$$M_2 = \begin{cases} (\delta^{(n)^T} * D * v)/d(n) , & \text{if case I} \\ (v^T * D * v)/d(n)^{1/2} , & \text{if case I} \end{cases}$$

$$M_E = \max\{M_E, M_1, M_2\}$$

SSOR-SI (continued)

(8) Computation of $\bar{\beta}$

$$\bar{\beta} = \max_{i,j} \{W_{ij}[E_{i-1,j} + N_{i-1,j}] + S_{ij}[E_{i,j-1} + N_{i,j-1}]\}$$

(9) Computation of ω, SPECR = $S(\mathcal{S}_\omega)$

If $M_E \leq 4\bar{\beta}$, then

$$\omega = 2/\{1 + [1-2M_E+4\bar{\beta}]^{1/2}\}$$

$$SPECR = (2-2\omega+\omega M_E)/(2-\omega M_E)$$

else

$$\omega = 2/\{1 + [1-4\bar{\beta}]^{1/2}\}$$

$$SPECR = \omega - 1$$

$$M_E = 2\sqrt{\bar{\beta}}\ , \quad \alpha = \alpha_2$$

(10) Computing ω^*

$$\omega^* = 2/\{1 + [1-4\bar{\beta}]^{1/2}\}$$

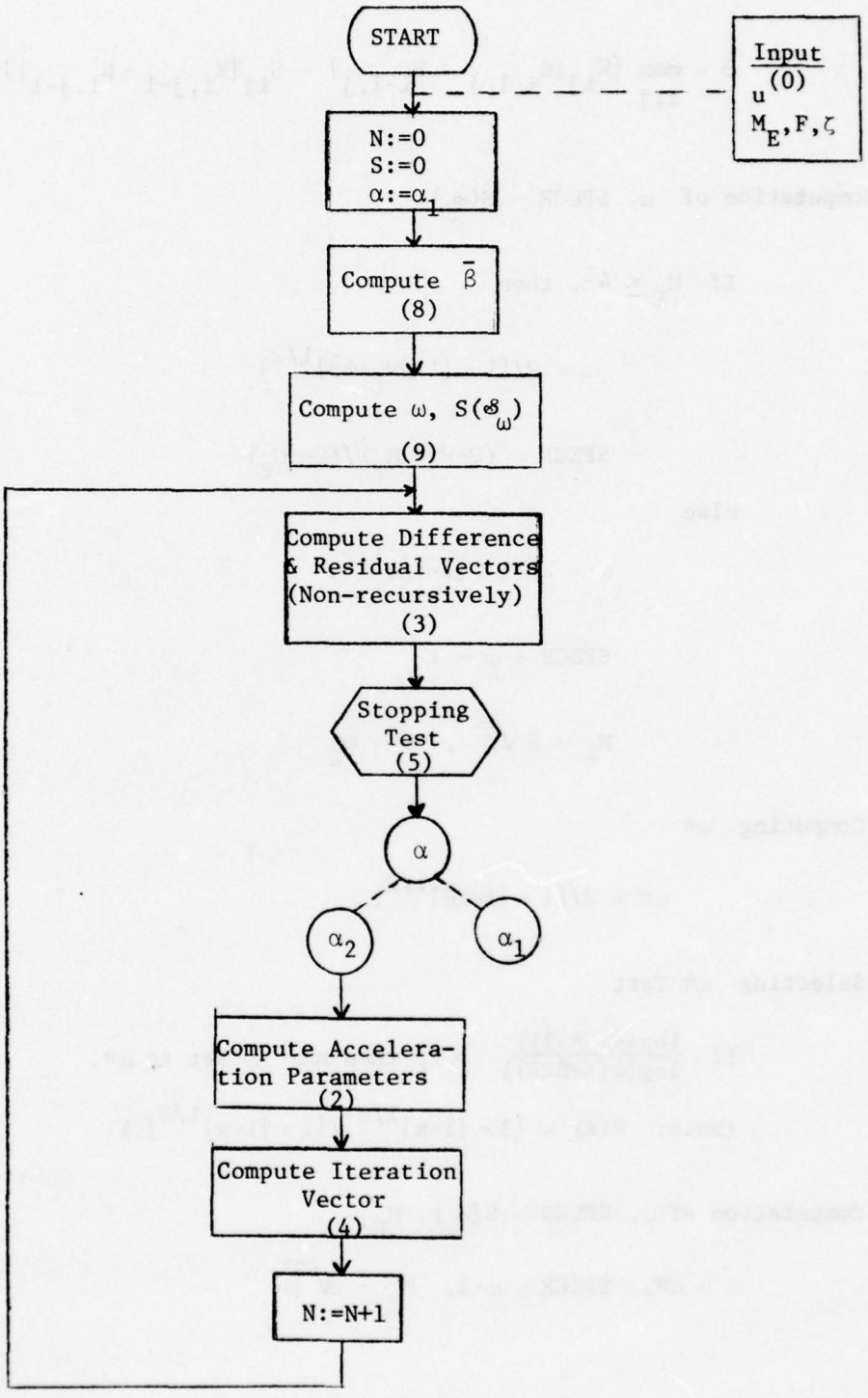
(11) Selecting ω^* Test

If $\dfrac{\log(\Phi(\omega^*-1))}{\log(\Phi(SPECR))} \geq F$, then new ω set to ω^*.

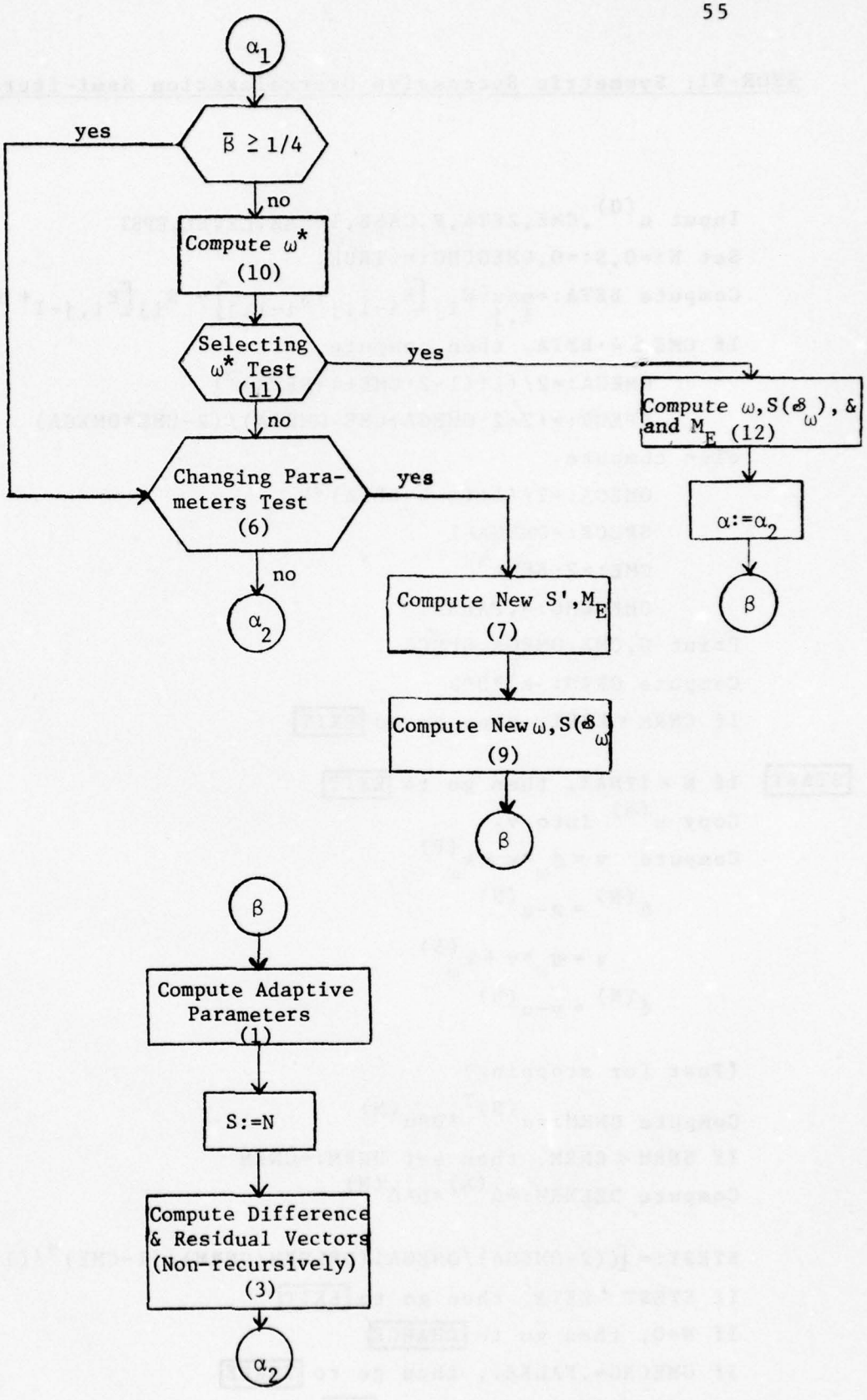
(Note: $\Phi(x) = \{1 - [1-x]^{1/2}\}/\{1 + [1-x]^{1/2}\}$.)

(12) Computation of ω, SPECR = $S(\mathcal{S}_\omega)$, M_E

$$\omega = \omega^*, \quad SPECR = \omega-1, \quad M_E = 2\sqrt{\bar{\beta}}$$



Flow Chart 5a: SSOR-SI Method



Flow Chart 5b: SSOR-SI Method

SSOR-SI: Symmetric Successive Overrelaxation Semi-iterative Algorithm

Input $u^{(0)}$, CME, ZETA, F, CASE, ITMAX, LEVEL, EPSI

Set N:=0, S:=0, OMEGCHG:=.TRUE.

Compute BETA:=$\max_{i,j}\{W_{ij}\left[E_{i-1,j}+N_{i-1,j}\right]+S_{ij}\left[E_{i,j-1}+N_{i,j-1}\right]\}$

If CME $\leq 4\cdot$BETA, then compute

OMEGA:=$2/(1+(1-2\cdot\text{CME}+4\cdot\text{BETA})^{\frac{1}{2}})$

SPECR:=$(2-2\cdot\text{OMEGA}+\text{CME}\cdot\text{OMEGA})/(2-\text{CME*OMEGA})$

else compute

OMEGA:=$2/(1+(1-4\cdot\text{BETA})^{\frac{1}{2}})$

SPECR:=OMEGA-1

CME:=$2\cdot\text{BETA}^{\frac{1}{2}}$

OMEGCHG:=.FALSE.

Print N, CME, OMEGA, SPECR

Compute CNRM:=$k^{T}*D*k$

If CNRM < EPSI, then go to [EXIT]

[START] If N > ITMAX, then go to [EXIT]

Copy $u^{(N)}$ into v.

Compute $v = \mathcal{L}_{\omega}*v + k_{\omega}^{(F)}$

$\Delta^{(N)} = v-u^{(N)}$

$v = \mathcal{U}_{\omega}*v + k_{\omega}^{(B)}$

$\delta^{(N)} = v-u^{(N)}$

(Test for stopping)

Compute UNRM:=$u^{(N)^{T}}*D*u^{(N)}$

If UNRM < CNRM, then set UNRM:=CNRM

Compute DELNRM:=$\Delta^{(N)}*D*\Delta^{(N)}$

STEST:=$\left[((2-\text{OMEGA})/\text{OMEGA})(\text{DELNRM}/\text{UNRM})/(1-\text{CME})^{\frac{1}{2}}/(1-\text{SPECR})\right]$

If STEST < ZETA, then go to [EXIT]

If N=0, then go to [CHANGE]

If OMECHG=.FALSE., then go to [THREE]

IF BETA $\geq 1/4$, then go to [ONE]

Compute OMEGAS:=2/(1+(1-4·BETA)$^{\frac{1}{2}}$)
TEMP1:=log(Φ(OMEGAS-1))
TEMP2:=log(Φ(SPECR))

where $\Phi(X)=(1-(1-X)^{\frac{1}{2}})/(1+(1-X)^{\frac{1}{2}})$.

If TEMP1/TEMP2 < F, then go to ONE

Set OMEGA:=OMEGAS
SPECR:=OMEGAS-1
OMECHG:=.FALSE.
CME:=2·BETA$^{\frac{1}{2}}$
S:=N

Print N,CME,SPECR,OMEGA
Go to TWO

ONE (Test for changing parameters)

Compute QA:=(DELNRM/DELSRM)$^{\frac{1}{2}}$, P:=N-S
QT:=2·R$^{P/2}$/(1+R^{P})
If QA ≥ QT^{F}, then go to CHANGE
else go to THREE

CHANGE (Change parameters)

If N=0, then set SIG1:=SPECR
Compute Z:=(1+R^{P})(QA+(QA^{2}-QT^{2})$^{\frac{1}{2}}$)/2
X:=Z$^{1/P}$
SIGE1:=(X+R/X)/(1+R)
SIG1:=(SPECR+SIGE1(2-SPECR))/2
Compute SIG2:= $\Delta^{(N)^T}$*D* $\tilde{\Delta}^{(N+1)}$/DELNRM
Set SME=max{SIG1,SIG2,SPECR}

(Determine new CME,OMEGA,SPECR)
Compute ZM1:=((1-SME)(1+BETA·OMEGA^{2})-OMEGA(2-OMEGA))/(OMEGA(OMEGA-1-SME)

Compute v=B*$\delta^{(n)}$
If CASE=.TRUE., then compute ZM2:=($\delta^{(n)^T}$*D*v)/($\delta^{(n)^T}$*D*$\delta^{(n)}$)
else compute ZM2:=[(v^{T}*D*v)/($\delta^{(n)^T}$*D*$\delta^{(n)}$)]$^{\frac{1}{2}}$

Compute CME:=max{CME,ZM1,ZM2}

Set S:=N

If CME $\leq$ 4BETA, then compute

OMEGA:=2/(1+(1-2CME+4BETA)$^{\frac{1}{2}}$

SPECR:=(2-2·OMEGA+CME·OMEGA)/(2-CME+OMEGA)

else compute

OMEGA:=2/(1+(1-4BETA)$^{\frac{1}{2}}$)

SPECR:=OMEGA-1

CME:=2·BETA$^{\frac{1}{2}}$

OMEGCHG:=.FALSE.

Print N,CME,OMEGA,SPECR

TWO Compute R:=(1-(1-SPECR)$^{\frac{1}{2}}$)/(1+(1-SPECR)$^{\frac{1}{2}}$)

SIGE:=SPECR/(2-SPECR)

GAMMA:=2/(2-SPECR)

RHO:=1

(Special procedure to recompute $\delta^{(n)}$ and $\Delta^{(n)}$ since OMEGA has been changed)

Copy $u^{(n)}$ into v

Compute $v = \mathcal{L}_\omega v + k_\omega^{(F)}$

$\Delta^{(n)} = v - u^{(n)}$

DELSRM: $= \Delta^{(n)^T} * D * \Delta^{(n)}$

$v = \mathcal{U}_\omega v + k_\omega^{(B)}$

$\delta^{(n)} = v - u^{(n)}$

$u^{(n+1)} = GAMMA \cdot \delta^{(n)} + u^{(n)}$

Go to ENDIT

THREE (OMEGA has not been changed)

If N=S+1, then compute RHO:=1/(1-SIGE^2/2)

else compute RHO:=1/(1-RHO·SIGE^2/4)

Set C1:=GAMMA·RHO,c2:=RHO,C3:=1-RHO

Compute $u^{(N+1)} = C1 \cdot \delta^{(N)} + C2 \cdot u^{(N)} + C3 \cdot u^{(N-1)}$

ENDIT Print N, $UNRM^{\frac{1}{2}}$, STEST, QA, QT^{F}, CME, RHO, GAMMA

Set N:=N+1

Go to START

EXIT Compute UNRM=$u^{(N)^{T}}*D*u^{(N)}$

Print N, $UNRM^{\frac{1}{2}}$

If LEVEL > 2, print $u^{(N)}$

END

VI SSOR-CG: Symmetric Successive Overrelaxation Conjugate Gradient Equations

(1) Difference Vector and Residual Vector (non-recursive computation)

$$v = \mathcal{L}_\omega u^{(n)} + k_\omega^{(F)}$$

$$\triangle^{(n)} = v - u^{(n)}$$

$$\delta^{(n)} = \mathcal{U}_\omega v + k_\omega^{(B)} - u^{(n)}$$

(2) Acceleration Parameters

$$v = \delta^{(n)} - \mathcal{L}_\omega * \delta^{(n)}$$

$$d(n) = \triangle^{(n)^T} * D * \triangle^{(n)}$$

$$\gamma_{n+1} = d(n) / (\triangle^{(n)^T} * D * v)$$

$$\rho_{n+1} = \begin{cases} 1 & , \quad n = s+1 \\ 1 \Big/ \left[1 - \left(\dfrac{\gamma_{n+1}}{\gamma_n \rho_n} \right) (d(n)/d(n-1)) \right] & , \quad n > s+1 \end{cases}$$

(3) Iteration Vector

$$u^{(n+1)} = \begin{cases} \gamma_{n+1} \delta^{(n)} + u^{(n)} & , \quad n = s+1 \\ \rho_{n+1} \{ \gamma_{n+1} \delta^{(n)} + u^{(n)} \} + (1-\rho_{n+1}) u^{(n-1)} & , \quad n > s+1 \end{cases}$$

SSOR-CG (continued)

(4) Difference Vector and Residual Vector (recursive computation)

$$\Delta^{(n+1)} = \begin{cases} \Delta^{(n)} - \gamma_{n+1} v & , \quad n = s+1 \\ \rho_{n+1}\{\Delta^{(n)} - \gamma_{n+1} v\} + (1-\rho_{n+1})\Delta^{(n-1)} & , \quad n > s+1 \end{cases}$$

$$v = \delta^{(n)} - v \qquad (= \mathcal{L}_{\omega} \delta^{(n)})$$

$$v = \mathcal{U}_{\omega} v \qquad (= \mathcal{B}_{\omega} \delta^{(n)})$$

$$\delta^{(n+1)} = \begin{cases} \gamma_{n+1} v + (1-\gamma_{n+1})\delta^{(n)} & , \quad n = s+1 \\ \rho_{n+1}\{\gamma_{n+1} v + (1-\gamma_{n+1})\delta^{(n)}\} + (1-\rho_{n+1})\delta^{(n-1)} & , \quad n > s+1 \end{cases}$$

(5) Stopping Test

$$d(n) = \Delta^{(n)^T} * D * \Delta^{(n)}$$

$$STEST = \left[(2-\omega)/(\omega(1-M_E)(1-SPECR)^2)\right]^{1/2} \left[d(n)/(u^{(n)^T} * D * u^{(n)})\right]^{1/2}$$

If STEST < ζ, then exit.

(6) Changing Parameter Test

$$\lambda_1 = -\log[\Phi(SPECR)/\Phi(SPECR/S')]$$

$$\lambda_2 = -\log[\Phi(S')]$$

If $(\lambda_1/\lambda_2) \leq F$, then change parameters.

(7a) Computation of S'

Compute S' which is the largest eigenvalue of the symmetric n x n tri-diagonal matrix ($s+1 \leq i \leq n$)

$$\left[\left(\frac{\rho_i^{-1}}{\gamma_{i-1}\rho_{i-1}\gamma_i\rho_i}\right)^{1/2}, \left(1-\frac{1}{\gamma_i}\right), \left(\frac{\rho_{i+1}^{-1}}{\gamma_i\rho_i\gamma_{i+1}\rho_{i+1}}\right)^{1/2}\right]$$

(7b) Compute new M_E

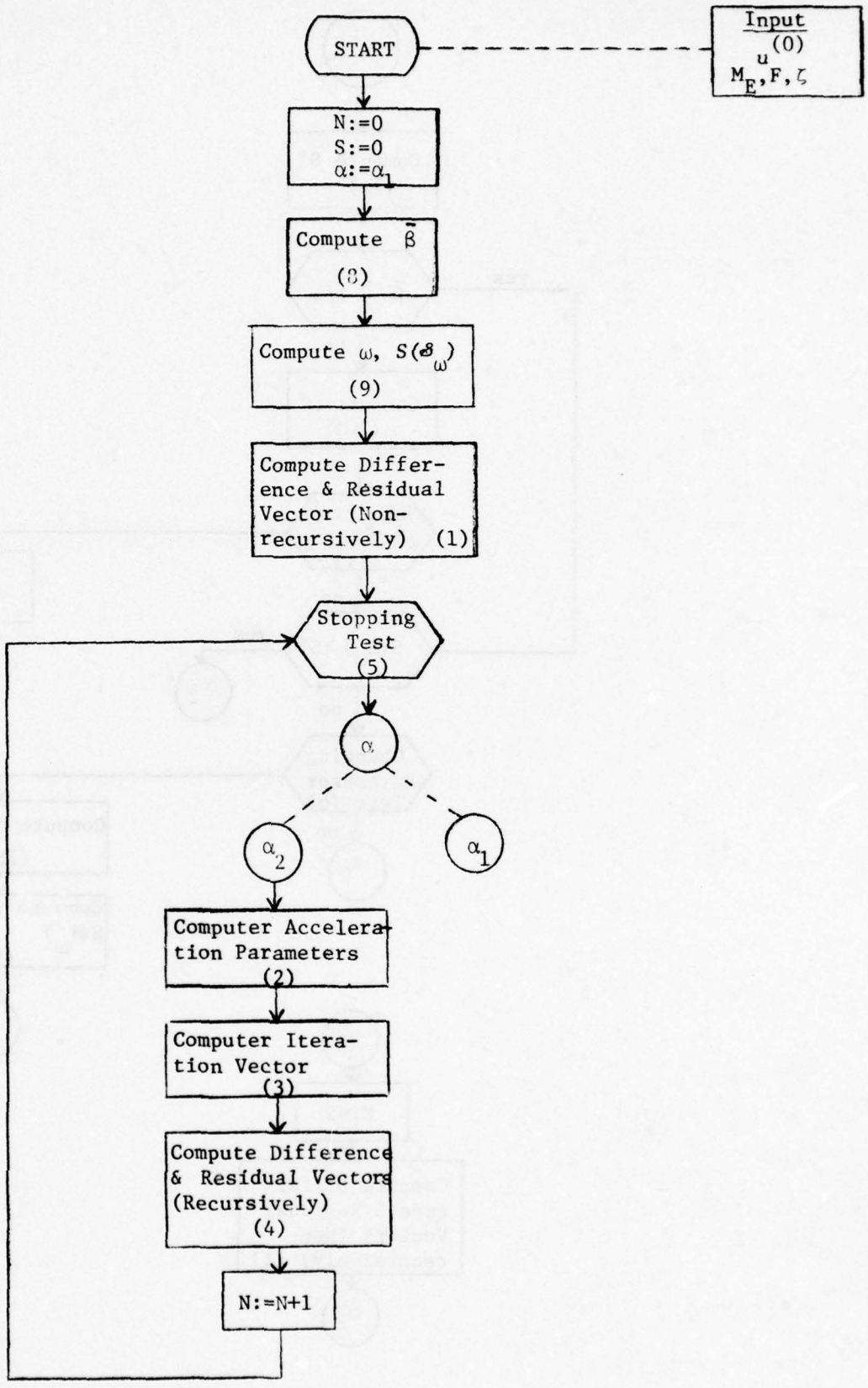
$$M_1 = [(1-S')(1+\bar{\beta}\omega^2) - \omega(2-\omega)]/[\omega(\omega-1-S')]$$

$$v = B*\delta^{(n)}, \quad d(n) = \delta^{(n)^T}*D*\delta^{(n)}$$

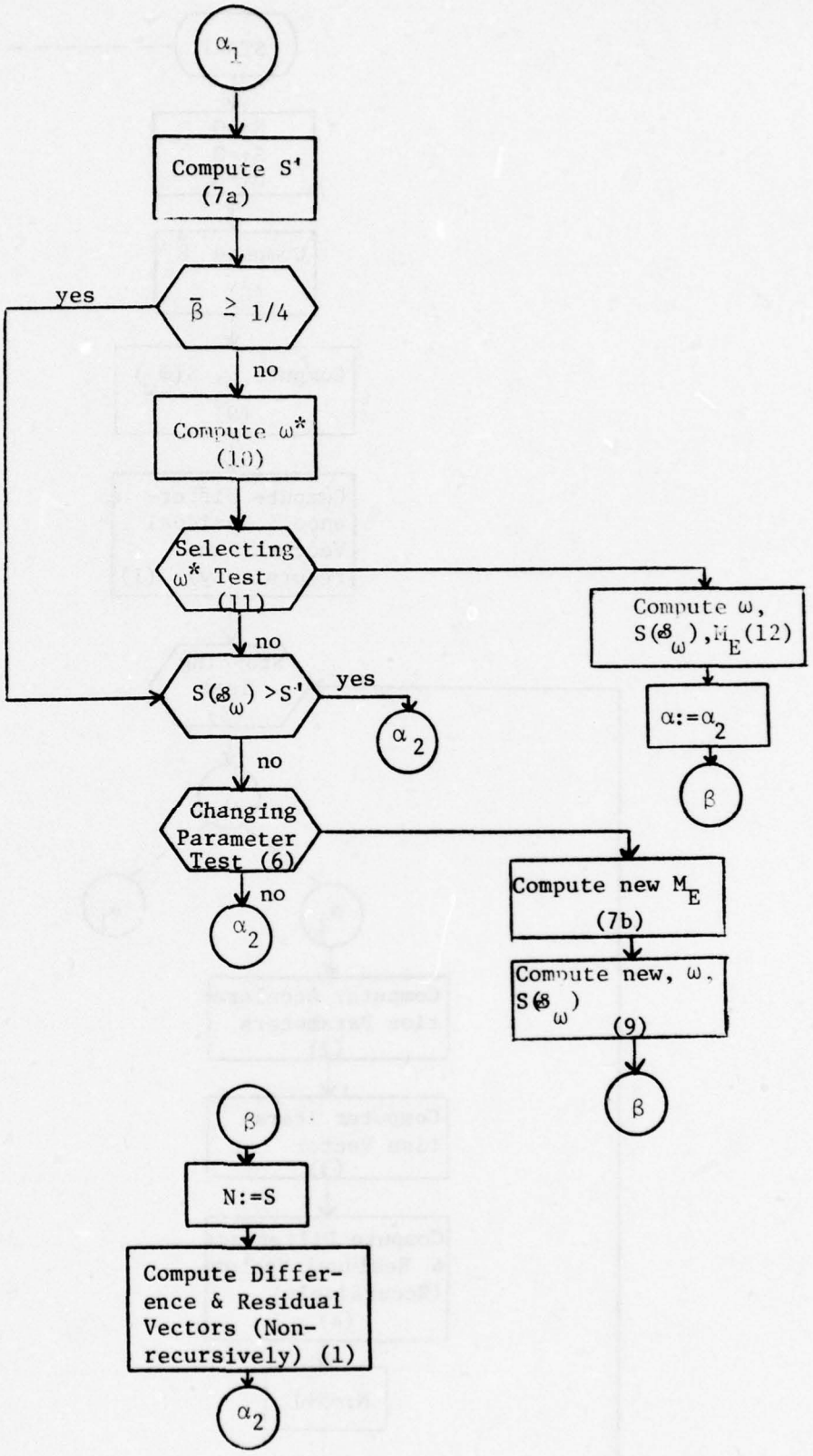
$$M_2 = \begin{cases} (\delta^{(n)^T}*D*v)/d(n), & \text{if case I} \\ \left[(v^T*D*v)/d(n)\right]^{1/2}, & \text{if case II} \end{cases}$$

$$M_E = \max\{M_E, M_1, M_2\}$$

(8)-(12) Same as SSOR-SI



Flow Chart 6a: SSOR-CG Method



Flow Chart 6b: SSOR-CG Method

SSOR-CG: Symmetric Successive Overrelaxation

Conjugate Gradient Algorithm

Input $u^{(0)}$, CME, F, ZETA, CASE, ITMAX, LEVEL, EPSI

Set N:=0, S:=0, OMEGCHG:=.TRUE.

Compute $\text{BETA}:=\max_{ij}\{W_{ij}\left[E_{i-1j}+N_{i-1j}\right]+S_{ij}\left[E_{ij-1}+N_{ij-1}\right]\}$

If $\text{CME} \leq 4\cdot\text{BETA}$, then compute

$\text{OMEGA}:=2/(1+(1-2\cdot\text{CME}+4\cdot\text{BETA})^{\frac{1}{2}})$

$\text{SPECR}:=(2-2\cdot\text{OMEGA}+\text{CME}\cdot\text{OMEGA})/(2-\text{CME}\cdot\text{OMEGA})$

else compute

$\text{OMEGA}:=2/(1+(1-4\,\text{BETA})^{\frac{1}{2}})$

$\text{SPECR}:=\text{OMEGA}-1$

$\text{CME}:=2\cdot\text{BETA}^{\frac{1}{2}}$

OMEGCHG:=.FALSE.

Print N, CME, OMEGA, SPECR

Compute $\text{CNRM}:=k^{T}*D*k$

If CNRM < EPSI, then go to EXIT

Copy $u^{(0)}$ into v.

Compute $v = \mathcal{L}_{\omega}*v + k_{\omega}^{(F)}$

$\Delta^{(0)} = v-u^{(0)}$

$v = \mathcal{U}_{\omega}*v + k_{\omega}^{(B)}$

$\delta^{(0)} = v-u^{(0)}$

START

If N > ITMAX, then go to EXIT

(Test for stopping)

Compute $\text{UNRM}:=u^{(n)^{T}}*D*u^{(n)}$

If UNRM < CNRM, then set UNRM:=CNRM

Compute $\text{DELNRM}:=\Delta^{(n)}*D*\Delta^{(n)}$

$\text{STEST}:=\left[\,(2-\text{OMEGA})/(\text{OMEGA})(\text{DELNRM}/\text{UNRM})/(1-\text{CME})\right]^{\frac{1}{2}}/(1-\text{SPECR})$

If STEST < ZETA, then go to [EXIT]
If OMEGCHG=.FALSE., then go to [THREE]
If N=0, then go to [THREE]

Else set SME:=maximum eigenvalue of the tri-diagonal matrix

$$\{[(\rho_i-1)/(\gamma_i\rho_i\gamma_{i-1}\rho_{i-1})]^{\frac{1}{2}},\ [1-1/\gamma_1]\ ,$$

$$[(\rho_{i+1}-1)/(\gamma_{i+1}\rho_{i+1}\gamma_i\rho_i)]^{\frac{1}{2}}\}\quad ,\quad \text{for } s+1 \le i \le N$$

If BETA $\ge 1/4$, then go to [ONE]
Compute OMEGAS:=$2/(1+(1-4\cdot$BETA$)^{\frac{1}{2}})$
TEMP1:=log(Φ(OMEGAS-1)
TEMP2:=log(Φ(SPECR))
where $\Phi(X) = (1-(1-X)^{\frac{1}{2}})/(1+(1-X)^{\frac{1}{2}})$
If TEMP1/TEMP2 $\le$ F, then go to [ONE]
Set OMEGA:=OMEGAS
SPECR:=OMEGAS-1
OMEGCHG:=.FALSE.
CME:=$2\cdot$BETA$^{\frac{1}{2}}$
S:=N
Print N,CME,SPECR,OMEGA
Go to [TWO]

(Test for changing parameters)
If SPECR > SME, then go to [THREE]
Compute

$$\lambda_1 = -\log\left(\frac{\Phi(\text{SPECR})}{\Phi\left(\frac{\text{SPECR}}{\text{SME}}\right)}\right)$$

$$\lambda_2 = -\log(\Phi(\text{SME}))$$

If $\lambda_1/\lambda_2 \ge$ F, then go to [THREE]

SSOR-CG (continued)

(Determine new CME,OMEGA,SPECR)

Compute ZM1:=((1-SME)(1+BETA·OMEGA^2)-OMEGA·(2-OMEGA))
/(OMEGA(OMEGA-1-SME))

Compute v=B*$\delta^{(n)}$

If CASE=.TRUE., then compute ZM2=($\delta^{(n)^T}$*D*V)/($\delta^{(n)^T}$*D*$\delta^{(n)}$)

else compute ZM2=$\left[(v^T*D*v)/(\delta^{(n)^T}*D*\delta^{(n)})\right]^{\frac{1}{2}}$

Compute CME:=max{CME,ZM1,ZM2}

Set S:=N

If CME < 4·BETA, then compute

OMEGA:=2./(1+(1-2·CME+4·BETA)$^{\frac{1}{2}}$)

SPECR:=(2-2·OMEGA+CME·OMEGA)/(2-CME·OMEGA)

else compute

OMEGA:=2./(1.+(1.-4·BETA)$^{\frac{1}{2}}$)

SPECR:=OMEGA-1

CME:=2·BETA$^{\frac{1}{2}}$

CMEGCHG:=.FALSE.

Print N,CME,OMEGA,SPECR

TWO

(Special procedure to recompute $\delta^{(n)}$ and $\Delta^{(n)}$ since OMEGA has been changed)

Copy $u^{(n)}$ into v

Compute $v = \mathcal{L}_\omega v + k_\omega^{(F)}$

$\Delta^{(n)} = v - u^{(n)}$

$v = \mathcal{U}_\omega v + k_\omega^{(B)}$

$\delta^{(n)} = v - u^{(n)}$

DELNRM = $\Delta^{(n)^T} * D * \Delta^{(n)}$

THREE

Copy $\delta^{(n)}$ into v

Compute $v = \mathcal{L}_\omega v$

$v = \delta - v$

γ_{N+1} = DELNRM/($\Delta^{(n)^T}$*D*v)

If N=S, then $\rho_{N+1}=1$

else compute

ρ_{N+1}:=1/(1-γ_{N+1}DELNRM/($\gamma_N\rho_N$DELSNM))

SSOR-CG (continued)

Set DELSNM:=DELNRM,

$C1:=\rho_{N+1}\gamma_{N+1}$, $C2:=\rho_{N+1}$, $C3:=1-\rho_{N+1}$, $C4:=\rho_{N+1}(1-\gamma_{N+1})$

Compute $u^{(n+1)}=C1*\delta^{(n)}+C2*u^{(n)}+C3*u^{(n-1)}$

$\Delta^{(n+1)}=-C1*v+C2*\Delta^{(n)}+C3*\Delta^{(n-1)}$

$v=\delta-v(=\mathcal{L}_\omega\delta^{(n)})$

$v=\mathcal{U}_\omega v(=\mathcal{S}_\omega\delta^{(n)})$

$\delta^{(n+1)}=C1*v+C4*\delta^{(n)}*C3*\delta^{(n-1)}$

Print N, $UNRM^{\frac{1}{2}}$, STEST, CME, OMEGA, SPECR, SME, ρ_{N+1}, γ_{N+1}

Set N:=N+1

Go to START

EXIT Compute $UNRM=u^{(n)^T}*D*u^{(n)}$

Print n, $UNRM^{\frac{1}{2}}$

If LEVEL > 2, print $u^{(n)}$

END

Appendix 2

Sample Problem

The current ITPACK routines can best be explained by looking at the code for a sample problem. In this appendix, the initial subroutines needed to define Problem (1) with Region (1) are given for the Compressed Jacobi Conjugate Gradient Method.

```
 SSS          M   M   PPPP    L       EEEEE   RRRR
S   S   A A   MM MM   P   P   L       E       R   R
S      A   A  M M M   P   P   L       E       R   R
 SSS   A   A  M M M   PPPP    L       EEE     RRRR
    S  AAAAA  M   M   P       L       E       R R
S   S  A   A  M   M   P       L       E       R  R
 SSS   A   A  M   M   P       LLLLL   EEEEE   R   R

                    09.40.14  24 JAN 77

      PROGRAM C ICGTST(INPUT,OUTPUT)
C
      REAL     GRIDX(21),GRIDY(21),COEF(441,4),WORKSP(1523),UNKNWN(441)
      REAL     BVALUS(4)
      INTEGER GTYPE(21,21),NDXEQ(441),INVNDX(441)
      DATA     IPTR/6LOUTPUT/,IRDR/5LINPUT/
      COMMON WORKSP
C *** BEGIN: COMMON DECK -  ITPACK
      COMMON / ITPACK / NGRPTS,NRDPTS,NBKPTS,NRDPP1,
     A                  CMUF,SMUF,ZETA,EPSI,F,GAMMA,RHO,SIGF,
     B                  HALT,CASEI,CHANGE,
     C                  RR,DELNNM,DELSNM,UDNM,TEST1,QA,QT,
     D                  IN,IS,ITMAX,
     E                  IPTR,IRDR,
     F                  OMEGA,SPECTRA,BETABAR,OMEGCHG
      LOGICAL HALT,CASEI,CHANGE,OMEGCHG
C *** END  : COMMON DECK -  ITPACK
C
C *** BEGIN: COMMON DECK - ELLPACK
      COMMON / BNDRY  / IPIECE,NBOUND,NBNDPT
      COMMON / CONSTS / IPACK1,IPACK2,IPACKB,INSIDE,HORZ,VERT,BOTH,
     A                  CORNER,INTER
      COMMON / CONTRL / DEBUG,LEVEL
      COMMON / CPDE   / CUXX,CUXY,CUYY,CUX,CUY,CU
      COMMON / EQFORM / NUMBEQ,NUMCOE
      COMMON / EQNDEX / NROW,NCOL
      COMMON / PROB   / DIM2,DIM3,POISON,LAPLAC,CONSTC,SELFAD,CROSST,
     A                  DIRICH,NEUMAN,MIXED,AX,BX,AY,BY,AZ,BZ,
     B                  NGRIDX,NGRIDY,NGRIDZ,UNIFRM,HX,HY,HZ,
     C                  ELLP77,RECTAN
      INTEGER HORZ,VERT,BOTH,CORNER,
     A        PIECE,BPTYPE,BNEIGH,BGRID
      LOGICAL DIM2,DIM3,POISON,LAPLAC,CONSTC,SELFAD,CROSST,DIRICH,
     A        NEUMAN,MIXED,UNIFRM,DEBUG,ELLP77,RECTAN,
     B        PTNAY
      REAL     AX,BX,AY,BY,AZ,BZ,HX,HY,HZ,CUXX,CUXY,CUYY,CUX,CUY,CU,
     A        XBOUND,YBOUND,BPARAM
C *** END  : COMMON DECK - ELLPACK
C
      READ(IRDR,70)  ICASES
      DO 60 IJKLM = 1,ICASES
C
C     ******************************************************************
C           INTERFACE 1:  INITIAL SITUATION
C     ******************************************************************
C
```

```
      DEBUG = .TRUE.
      LEVEL = 3
      NGRDXD = 21
      NGRDYD = 21
      MXNCOE = 4
      MXNEQ = 441
      ITMAX = 500
      ZETA = .000001
      EPSI = .000001
      CMUE = 0.0
      CASEI = .FALSE.
      F = .75
C
C     DEFINE PROBLEM ON OUTPUT FILE
C
      WRITE(IPTR,80)
      WRITE(IPTR,90) F,CMUE,ZETA,EPSI
C
C     ***************************************************************
C           INTERFACE 2:  DOMAIN PROCESSING
C     ***************************************************************
C
      CALL REGION(GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD)
C
C     ***************************************************************
C           INTERFACE 3:  EQUATION GENERATION
C     ***************************************************************
C
      CALL FIVEPT(GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,
     A            MXNEQ)
C
C     ***************************************************************
C           INTERFACE 4:  EQUATION INDEXING
C     ***************************************************************
C
      CALL RBORD(NDXEQ,MXNEQ,INVNDX)
C
C     ***************************************************************
C           INTERFACE 5:  EQUATION SOLUTION
C     ***************************************************************
C
      CALL INTUNK(GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD,MXNEQ,
     A            UNKNWN)
      CALL CJCG(GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD,COEF,MXNCOE,MXNEQ,
     A          NDXEQ,UNKNWN)
C
C     ***************************************************************
C           INTERFACE 6:  OUTPUT
C     ***************************************************************
C
      DO 50 IJ = 1,NGRPTS
      IX = MOD(IJ-1,NGRIDX) + 1
      JY = (IJ-IX)/NGRIDX + 1
      GO TO (10,20,30) GTYPE(IX,JY)
   10 CONTINUE
      WORKSP(IJ) = TRUE(GRIDX(IX),GRIDY(JY))
      GO TO 40
```

```
   20 CONTINUE
      CALL BCOND(IDUMMY,GRIDX(IX),GRIDY(JY),BVALUS)
      WORKSP(IJ) = BVALUS(4)/BVALUS(1)
      GO TO 40
   30 CONTINUE
      WORKSP(IJ) = 0.0
   40 CONTINUE
      WORKSP(IJ+NGRPTS) = WORKSP(IJ) - UNKNWN(IJ)
   50 CONTINUE
      TRUNORM = UTDV(GTYPE,NGRDXD,NGRDYD,COEF,MXNCOE,MXNEQ,NDXEQ,
     A               WORKSP(1),WORKSP(1),1,NGRPTS)
      ERRNORM = UTDV(GTYPE,NGRDXD,NGRDYD,COEF,MXNCOE,MXNEQ,NDXEQ,
     A               WORKSP(1+NGRPTS),WORKSP(1+NGRPTS),1,NGRPTS)
      TRUNORM = SQRT(TRUNORM)
      ERRNORM = SQRT(ERRNORM)
      WRITE(IPTR,100) TRUNORM,ERRNORM
      CALL VOUT(GRIDX,NGRDXD,GRIDY,NGRDYD,WORKSP(1),NGRPTS)
   60 CONTINUE
   70 FORMAT(I5)
   80 FORMAT(1H1,////30X,*THE METHOD BEING USED:   CJCG*,/30X,
     A        *THE ORDERING BEING USED:   RED-BLACK*,
     B        /30X,*CASE II OF THE ADAPTIVE PROCEDURE*,//30X,
     C        *BOUNDARY VALUES ARE SET TO:  0.0*,/30X,
     D        *INITIAL SOLUTION IS:  0.0*,/30X,
     E        *TEST PROBLEM NO. 2*)
   90 FORMAT(///30X,*STARTING ITERATIVE PARAMETERS ARE:*,/35X,
     A        *F =*,3X,F15.8,/35X,*CMUE =*,
     B        F15.8,/35X,*ZETA =*,F15.8,/35X,*EPSI =*,F15.8)
  100 FORMAT(1H1,//30X,*D TO 1/2 NORM OF TRUE SOLUTION =*,E15.8,/30X,
     A        *D TO 1/2 NORM OF THE ERROR =*,E15.8,//)
      END



      SUBROUTINE PDE(X,Y,CVALUS)
      REAL CVALUS(7)
      DATA PI/3.14159265358979/
C
C     TWO DIMENSIONS
C
C     VALUES OF EQUATION COEFFICEINTS AT (X,Y) IN ORDER:
C           UXX,UXY,UYY,UX,UY,U,RIGHT SIDE
C
      EXY = EXP(X*Y)
      CVALUS(1) = EXY
      CVALUS(2) = 0.0
      CVALUS(3) = 1.0/EXY
      CVALUS(4) = 0.0
      CVALUS(5) = 0.0
      CVALUS(6) = -1./(1. + X + Y)
      TSX = SIN(PI*X)
      TSY = SIN(PI*Y)
      TCX = COS(PI*X)
      TCY = COS(PI*Y)
      E2XY = EXY*EXY
```

```
      TEMP = PI*(X*TSX*TCY + 3.*Y*E2XY*TCX*TSY)
      TEMP1 = TSX*TSY*((2.*Y*Y-PI*PI)*E2XY - PI*PI -EXY/(1.+X+Y))
      CVALUS(7) = TEMP+TEMP1
C
      RETURN
      END



      SUBROUTINE BCOND(I,X,Y,BVALUS)
      REAL BVALUS(4)
C
C        VALUES OF BOUNDARY CONDITIONS COEFFICEINTS AT (X,Y)
C        IN THE ORDER:
C           U,UX,UY,RIGHT SIDE
C
      BVALUS(1) = 1.0
      BVALUS(2) = 0.0
      BVALUS(3) = 0.0
      BVALUS(4) = TRUE(X,Y)
C
      RETURN
      END



      FUNCTION APXUNK(X,Y)
C
C     INITIAL APPROXIMATION TO UNKNOWN VALUES
C
      APXUNK = 0.0
C
      RETURN
      END



      FUNCTION TRUE(X,Y)
      DATA PI/3.14159265358979/
C
C     TRUE SOLUTION
C
      TRUE = EXP(X*Y)*SIN(PI*X)*SIN(PI*Y)
C
      RETURN
      END
```

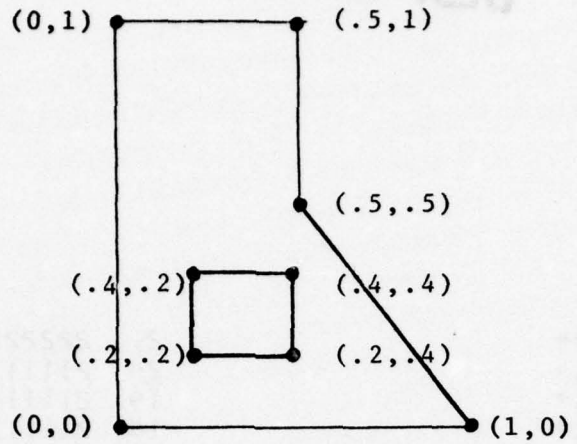
Appendix 3

The Subroutine REGION

REGION is a subroutine which superimposes a grid of size h on a region defined by closed contours. This routine constructs a two-dimensional integer array over the smallest rectangle circumscribing the possibly irregular region and denotes each grid point with integer values, namely, +1 for interior points, +2 for boundary points, +3 for exterior points. To utilize REGION, the vertices defining the boundary of each contour in the particular region are specified and ordered so that the interior of the region always lies on the left. The x and y coordinates of the endpoints of each consecutive line segment defining a contour are given as input data. The permissible line segments are those in an arbitrarily chosen xy-plane which are parallel to the x-axis or the y-axis or which form a 45° angle with an axis whose endpoints are grid points for the prescribed h.

While REGION was originally developed several years ago, it has been modified and improved recently. This recoding has removed restrictions such as the limits on the number of allowable vertices and on the number of possible contours. REGION is now coded in standard Fortran with an improved data structure and with optimized code where possible. The subroutine REGION is now compatible with code specifications outlined in the ELLPACK Contributor's Guide [5]. Hence, it is being utilized at UT Austin as an ELLPACK module to perform domain processing. While REGION is somewhat limited with regard to the types of domains it can process, it does work successfully on very complicated regions with a number of "holes" in them--all defined with horizontal, vertical, and 45° line segments.

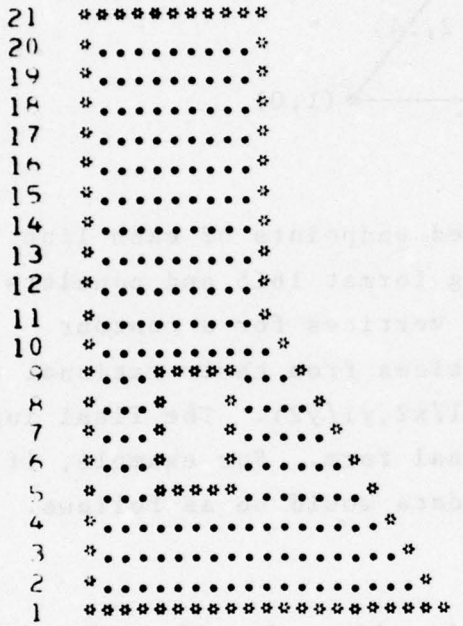
As an illustrative example of the use of subroutine REGION, consider the two-contour region (4).



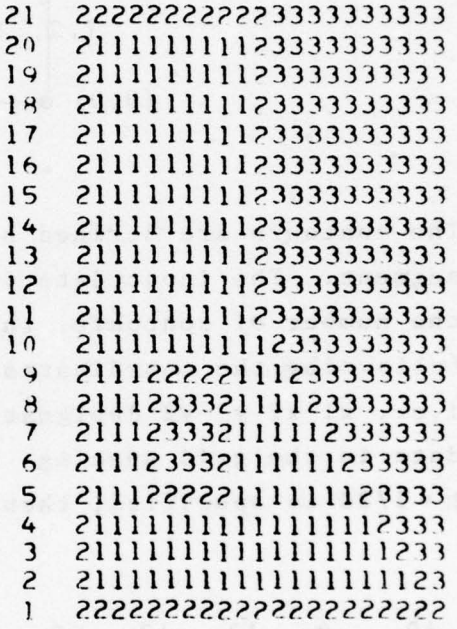
The contours are defined by the labeled endpoints of each line segment. The input data is read using format 16I5 and consists of the number of contours, the number of vertices for a contour followedby the coordinates of the vertices from their rational form, i.e., x1 x2 y1 y2 designate vertex (x1/x2,y1/y2). The final input data is the grid spacing h in rational form. For example, if h = 1/20 is specified, then the input data would be as follows.

```
2
5
0   10    0   10   10   10    0   10    5   10    5   10    5   10   10   10
0   10   10   10
4
2   10    2   10    2   10    4   10    4   10    4   10    4   10    2   10
1   20
```

The printed output from REGION and the structure of the integer array GTYPE for this input data is as follows.



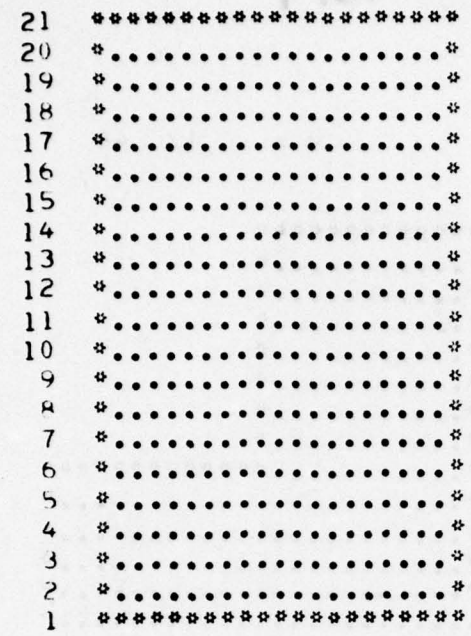
REGION Printed Output



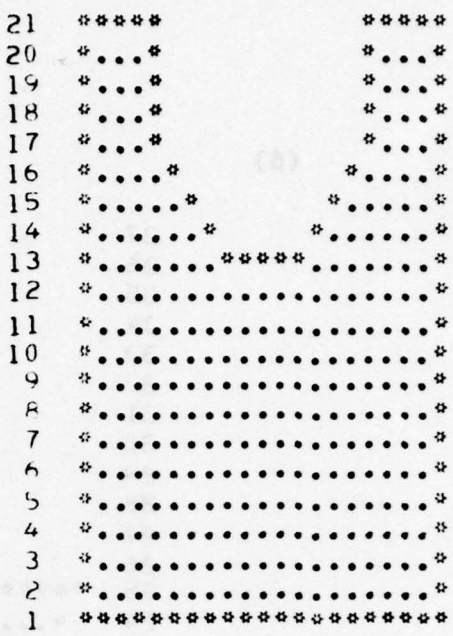
Integer Array GTYPE

The printed output from REGION for the other five test regions follows.

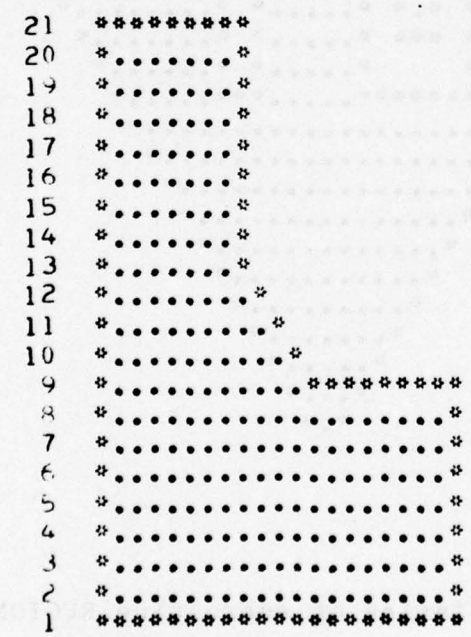
(1)



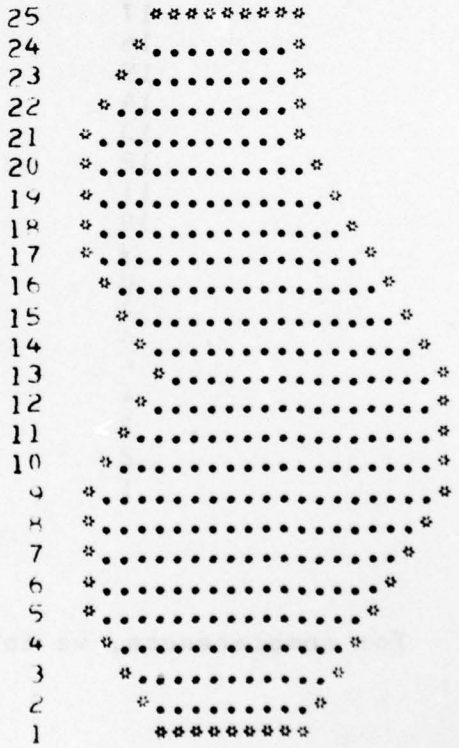
(2)



(3)

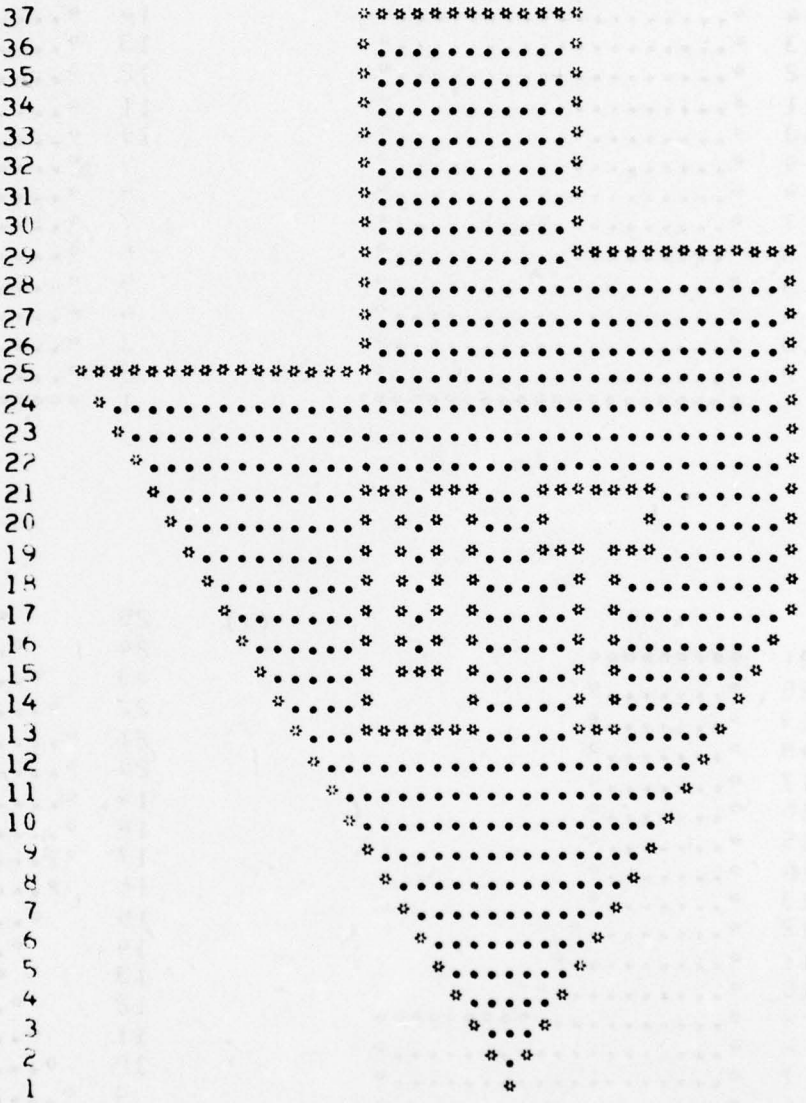


(5)



(6)

H = 1/ 40



For completeness, we now give the listing of subroutine REGION.

```
RRRR    EEEEE    GGGG    III     000    N   N
R   R   E       G         I     0   0   NN  N
R   R   E       G         I     0  00   N N N
RRRR    EEE     G  GG     I     0 0 0   N  NN
R R     E       G   G     I     00  0   N   N
R  R    E       G   G     I     0   0   N   N
R   R   EEEEE    GGGG    III     000    N   N

                11.42.52  07 JUN 77

      SUBROUTINE REGION (GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD)
C
C     ------------------------------------------------------------------
C     FUNCTION : SUPERIMPOSES A MESH OF SIZE HX=HY=H1/H2 ON A REGION
C                DEFINED BY CLOSED CONTOURS AND CONSTRUCTS AN INTEGER
C                ARRAY WHICH DESCRIBES EACH MESH POINT ON THE SMALLEST
C                RECTANGLE CIRCUMSCRIBING THE POSSIBLY IRREGULAR REGION
C                AS AN INTERIOR POINT (+1), AN BOUNDARY POINT (+2), OR
C                AN EXTERIOR POINT (+3).
C
C     USAGE :    CALL REGION (GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD)
C
C     PARAMETERS :
C
C     GTYPE  - GTYPE IS AN NGRDXD BY NGRDYD INTEGER ARRAY USED TO
C              INDICATE THE TYPE OF POINT ON THE GRID.  THE NUMBERS
C              1, 2, OR 3 INDICATE RESPECTIVELY INTERIOR, BOUNDARY,
C              OR EXTERIOR POINTS OF THE GRID.
C
C     NGRDXD - NGRDXD IS THE ROW DIMENSION OF THE ARRAY GTYPE AS
C              SPECIFIED IN THE CALLING PROGRAM.
C
C     NGRDYD - NGRDYD IS THE COLUMN DIMENSION OF THE ARRAY GTYPE
C              SPECIFIED IN THE CALLING PROGRAM.
C
C     GRIDX  - GRIDX IS AN ARRAY OF LENGTH NGRDXD DIMENSIONED IN THE
C              CALLING PROGRAM.  UPON LEAVING REGION IT CONTAINS THE
C              X COORDINATES OF THE MESH LINES STARTING IN THE
C              LOWER LEFT HAND CORNER.
C
C     GRIDY  - GRIDY IS AN ARRAY OF LENGTH NGRDYD DIMENSIONED IN THE
C              CALLING PROGRAM.  UPON LEAVING REGION IT CONTAINS THE
C              Y COORDINATES OF THE MESH LINES STARTING IN THE
C              LOWER LEFT HAND CORNER.
C
C
C     OTHER PARAMETERS PASSED IN LABELED COMMON ARE:
C
C     LEVEL  = 0     NO PRINTING FROM REGION
C            = 1     THE INPUT DATA ONLY IS PRINTED.
C            = 2     THE GRAPH OF THE REGION ONLY IS PRINTED.
C            = 3     PRINT BOTH INPUT DATA AND GRAPH OF REGION
C
C     DEBUG  IS A LOGICAL DEBUGGING PARAMETER.  IF TRUE THEN LEVEL
C            IS RESET TO 3.  IF FALSE NO ACTION IS TAKEN.
C
```

```
C     NGRIDX IS THE NUMBER OF MESH POINTS IN THE X-DIRECTION OF THE
C            CIRCUMSCRIBED RECTANGLE.  THIS IS COMPUTED IN REGION.
C
C     NGRIDY IS THE NUMBER OF MESH POINTS IN THE Y-DIRECTION OF THE
C            CIRCUMSCRIBED RECTANGLE.  THIS IS COMPUTED IN REGION.
C
C     NGRPTS IS THE NUMBER OF TOTAL MESH POINTS OF THE CIRCUMSCRIBED
C            RECTANGLE.  THIS IS COMPUTED IN REGION.
C
C     HX,HY  ARE THE MESH SIZE FOR THE GRID.  REGION READS H1 AND
C            H2 FROM DATA AND COMPUTES HX=HY=H1/H2.  HX AND HY ARE
C            THEN RETURNED TO THE CALLING PROGRAM.
C
C     AX,BX  ARE THE MINIMUM AND MAXIMUM VALUES OF THE X COORDINATE.
C            REGION COMPUTES THESE AND RETURNS THEM IN LABELED COMMON.
C
C     AY,BY  ARE THE MINIMUM AND MAXIMUM VALUES OF THE Y COORDINATE.
C            REGION COMPUTES THESE AND RETURNS THEM IN LABELED COMMON.
C
C
C     PARAMETERS USING BLANK COMMON ARE:
C
C     L      IS DESCRIBED BELOW
C
C
C     OTHER PARAMETERS :
C
C     H1,H2  ARE PARAMETERS INDICATING THE UNIFORM MESH SIZE
C            IN RATIONAL FORM.  THESE AS WELL AS L ARE READ IN AS
C            DATA AND ARE DESCRIBED FURTHER BELOW.
C
C
C     THE DEFINITION OF THE REGION IS AS FOLLOWS:
C
C         KN = NUMBER OF CONTOURS IN THE REGION.
C         L(5,K) = NUMBER OF VERTICES ON THE K--TH CONTOUR -- 3 OR MORE
C         L(1,I)/L(2,I) = THE X COORDINATE OF THE I-TH VERTEX.
C         L(3,I)/L(4,I) = THE Y COORDINATE OF THE I-TH VERTEX.
C         H1/H2 = THE MESH SIZE TO BE CONSIDERED.
C
C     NOTES: (1) THE ARRAY L USES BLANK COMMON FURNISHED IN THE
C                CALLING PROGRAM
C            (2) THE INPUT DATA FOR THE VERTICES MUST BE ORDERED SO
C                THE INTERIOR OF THE REGION ALWAYS LIES TO THE LEFT
C            (3) REQUIRED SUBROUTINES -- RTREG,IABS,MODS
C
C
C     WRITTEN OR MODIFIED BY                          DATE
C
C     ROGER G. GRIMES AND DAVID R. KINCAID            JUNE 1977
C     ROGER G. GRIMES AND DAVID R. KINCAID            FEBRUARY 1977
C     JAMES D. SULLIVAN AND DAVID R. KINCAID          FEBRUARY 1976
C     ALKIS J. MOURADOGLOU AND JOHN H. DAUWALDER      APRIL 1967
C
C                  CENTER FOR NUMERICAL ANALYSIS/COMPUTATION CENTER
C                  UNIVERSITY OF TEXAS AT AUSTIN
C
```

```
C     REFERENCES:
C
C     (1)     KINCAID,DAVID R. AND ROGER G. GRIMES, ***** CNA REPORT ***
C
C     (2)     ALKIS J. MOURADOGLOU AND JOHN H. DAUWALDER, ΞREGION
C             SPECIFICATION ROUTINEΞ, V2 UTEX REGION, UTV2-01-CC026,
C             COMPUTATION CENTER, UT-AUSTIN, APRIL 1967.
C
C     (3)     A. J. MOURADOGLOU, ΞNUMERICAL STUDIES ON THE CONVERGENCE
C             OF THE PEACEMAN-RACHFORD ALTERNATING DIRECTION IMPLICIT
C             METHODΞ, TNN-67, COMPUTATION CENTER, UT-AUSTIN, JUNE 1967.
C
C     ------------------------------------------------------------------
C
      REAL GRIDX(NGRDXD),GRIDY(NGRDYD)
      INTEGER GTYPE(NGRDXD,NGRDYD)
      INTEGER R,S,RS1,SSAVE,SET,P,Q,FLAG,H1,H2
      COMMON LETTER(3),NUMBER(3),L(5,1)
C
C *** BEGIN: COMMON DECK -  ITPACK
C *** END  : COMMON DECK -  ITPACK
C
C *** BEGIN: COMMON DECK - ELLPACK
C *** END  : COMMON DECK - ELLPACK
C
      NUMBER(1)=2
      NUMBER(2)=1
      NUMBER(3)=3
      LETTER(1)=1H*
      LETTER(2)=1H.
      LETTER(3)=1H
      MP1=NGRDXD-1
      MQ1=NGRDYD-1
      READ (IRDR,610) KN
      READ (IRDR,610) KC01
      L(5,1)=KC01
      READ (IRDR,610) (L(1,J),L(2,J),L(3,J),L(4,J),J=1,KC01)
      IF (KN.LE.1) GO TO 20
      DO 10 K=2,KN
         READ (IRDR,610) L(5,K)
         KC02=KC01+1
         KC01=KC01+L(5,K)
C
C***  IT IS ASSUMED THAT L(5,K) > 0 FOR K=1,2,...,KN AND THAT KN > 0.
C
   10 READ (IRDR,610) (L(1,J),L(2,J),L(3,J),L(4,J),J=KC02,KC01)
C
C***  THE READING OF THE COORDINATES OF THE VERTICES IS WITH 16I5 FORMAT
C***  L(1,I),L(2,I),L(3,I),L(4,I),L(1,I+1),L(2,I+1),L(3,I+1),...
C
   20 READ (IRDR,610) H1,H2
      KJ2=0
      IF (DEBUG) LEVEL=3
      IF (MOD(LEVEL,2).EQ.0) GO TO 40
C
C***  PRINT INPUT DATA
C
```

```
      WRITE (IPTR,620)
      KJ2=0
      DO 30 K=1,KN
         KJ1=KJ2+1
         KJ2=KJ2+L(5,K)
         WRITE (IPTR,630) K
         WRITE (IPTR,640)
         WRITE (IPTR,660) (L(1,I),L(2,I),I=KJ1,KJ2)
         WRITE (IPTR,650)
   30 WRITE (IPTR,660) (L(3,I),L(4,I),I=KJ1,KJ2)
      KJ2=0
   40 DO 60 I=1,KN
         KJ1=KJ2+2
         KJ2=KJ2+L(5,I)
         IF (KJ1.GT.KJ2) STOP 1
         DO 50 KS=KJ1,KJ2
            KS1=KS-1
            IF (L(1,KS)*L(2,KS1).EQ.L(2,KS)*L(1,KS1).OR.L(3,KS)*L(4,KS1)
     1      .EQ.L(4,KS)*L(3,KS1).OR.IABS((L(1,KS)*L(2,KS1)-L(1,KS1)*L(2,
     2      KS))*L(4,KS)*L(4,KS1)).EQ.IABS(L(2,KS)*L(2,KS1)*(L(3,KS)*L(4
     3      ,KS1)-L(3,KS1)*L(4,KS)))) GO TO 50
            WRITE (IPTR,670) KS,I
            STOP 2
   50    CONTINUE
         KJ1=KJ1-1
         DO 60 J=1,KN
            IF (L(1,KJ1)*L(2,KJ2).EQ.L(1,KJ2)*L(2,KJ1).OR.L(3,KJ1)*L(4,KJ2)
     1    .EQ.L(3,KJ2)*L(4,KJ1).OR.IABS((L(1,KJ1)*L(2,KJ2)-L(1,KJ2)*L(2,K
     2    J1))*L(4,KJ1)*L(4,KJ2)).EQ.IABS(L(2,KJ1)*L(2,KJ2)*(L(3,KJ1)*L(4
     3    ,KJ2)-L(3,KJ2)*L(4,KJ1)))) GO TO 60
            WRITE (IPTR,670) KJ1,J
            STOP 3
   60 CONTINUE
C
C***  PICK THE MAX AND MIN OF THE COORDINATES OF THE VERTICES OF
C***  ALL CONTOURS.
C***    MINX1/MINX2 =  MIN OF X COORDINATES
C***    MINY1/MINY2 =  MIN OF Y COORDINATES
C***    MAXX1/MAXX2 =  MAX OF X COORDINATES
C***    MAXY1/MAXY2 =  MAX OF Y COORDINATES
C
      MAXX1=L(1,1)
      MINX1=MAXX1
      MAXX2=L(2,1)
      MINX2=MAXX2
      MAXY1=L(3,1)
      MINY1=MAXY1
      MAXY2=L(4,1)
      MINY2=MAXY2
C
C***  MINX1  =  MAXX1  =  X ( 1 , 1 )
C***  MINX2  =  MAXX2  =  X ( 2 , 1 )
C***  MINY1  =  MAXY1  =  X ( 3 , 1 )
C***  MINY2  =  MAXY2  =  X ( 4 , 1 )
C
      IF (KCO1.LE.1) STOP 4
      DO 100 I=2,KCO1
```

```
            IF (MAXX1*L(2,I).GE.MAXX2*L(1,I)) GO TO 70
            MAXX1=L(1,I)
            MAXX2=L(2,I)
            GO TO 80
   70       IF (MINX1*L(2,I).LE.MINX2*L(1,I)) GO TO 80
            MINX1=L(1,I)
            MINX2=L(2,I)
   80       IF (MAXY1*L(4,I).GE.MAXY2*L(3,I)) GO TO 90
            MAXY1=L(3,I)
            MAXY2=L(4,I)
            GO TO 100
   90       IF (MINY1*L(4,I).LE.MINY2*L(3,I)) GO TO 100
            MINY1=L(3,I)
            MINY2=L(4,I)
  100    CONTINUE
         IF (H2*(MAXX1*MINX2-MAXX2*MINX1).LE.MP1*H1*MINX2*MAXX2) GO TO 110
         WRITE (IPTR,680)
         STOP 5
  110    IF (H2*(MAXY1*MINY2-MAXY2*MINY1).LE.MQ1*H1*MINY2*MAXY2) GO TO 120
         WRITE (IPTR,690)
         STOP 6
C
C***     CHECK THAT ALL BOUNDARY POINTS ARE INTEGRAL MULTIPLES OF H
C
  120    DO 150 I=1,KCO1
C
C***     AT THIS POINT KCO1 = THE TOTAL NUMBER OF VERTICES.
C
            MM=H2*(L(1,I)*MINX2-L(2,I)*MINX1)
            NN=H1*MINX2*L(2,I)
            KK=MM/NN
            IF (KK*NN.EQ.MM) GO TO 140
  130       WRITE (IPTR,700)
            STOP 7
  140       L(1,I)=KK
            MM=H2*(L(3,I)*MINY2-L(4,I)*MINY1)
            NN=H1*MINY2*L(4,I)
            KK=MM/NN
            IF (KK*NN.NE.MM) GO TO 130
            L(2,I)=KK
  150    CONTINUE
C
C***     DETERMINE NGRIDX AND NGRIDY.
C
         NGRXM1=L(1,1)
         NGRYM1=L(2,1)
         IF (KCO1.LT.2) STOP 10
         DO 170 I=2,KCO1
            IF (NGRXM1.GT.L(1,I)) GO TO 160
            NGRXM1=L(1,I)
  160       IF (NGRYM1.GT.L(2,I)) GO TO 170
            NGRYM1=L(2,I)
  170    CONTINUE
         NGRIDX=NGRXM1+1
         NGRXP1=NGRIDX+1
         NGRIDY=NGRYM1+1
         NGRYP1=NGRIDY+1
```

```
C
C***  SET THE ARRAY =GTYPE= TO ZERO.
C
      DO 180 J=1,NGRIDY
      DO 180 I=1,NGRIDX
  180 GTYPE(I,J)=0
C
C***  DEFINE THE BOUNDARY POINTS WHICH ARE NOT VERTICES
C***  IN THE ARRAY GTYPE
C
      KJ2=0
      DO 480 K=1,KN
         KJ1=KJ2+1
         KJ2=KJ2+L(5,K)
      DO 480 J=KJ1,KJ2
         IF (J.NE.KJ1) GO TO 190
         IPP=1+L(1,KJ2)
         IQP=1+L(2,KJ2)
         GO TO 200
  190    IPP=1+L(1,J-1)
         IQP=1+L(2,J-1)
  200    IP=1+L(1,J)
         IQ=1+L(2,J)
         IF (J.NE.KJ2) GO TO 210
         IPN=1+L(1,KJ1)
         IQN=1+L(2,KJ1)
         GO TO 220
  210    IPN=1+L(1,J+1)
         IQN=1+L(2,J+1)
  220    CALL RTREG (IPP,IQP,IP,IQ,R)
         CALL RTREG (IPN,IQN,IP,IQ,S)
         IF (GTYPE(IP,IQ).EQ.0) GO TO 230
         WRITE (IPTR,710) J,K
         STOP 11
  230    MODR=MOD(R+4,8)
         MODS=MOD(S+4,8)
         IF (R.LE.S) GO TO 250
         IF (MODR.LE.MODS) GO TO 240
         GTYPE(IP,IQ)=+1
         GO TO 280
  240    GTYPE(IP,IQ)=+10
         GO TO 280
  250    IF (R.NE.S) GO TO 260
         WRITE (IPTR,720) J,K
         STOP 12
  260    IF (MODR.LT.MODS) GO TO 270
         GTYPE(IP,IQ)=-1
         GO TO 280
  270    GTYPE(IP,IQ)=-10
  280    IF (J.NE.KJ1) GO TO 290
         RS1=R
         SSAVE=S
         GO TO 480
  290    IF (IABS(IPP-IP).LE.1.AND.IABS(IQP-IQ).LE.1) GO TO 470
         MODR=MOD(R+4,8)
         MODS=MOD(SSAVE+4,8)
         IF (R.LE.SSAVE) GO TO 300
```

```
            SET=+10
            IF (MODR.GT.MODS) SET=+1
            GO TO 310
  300       SET=-10
            IF (MODR.GT.MODS) SET=-1
  310       IPM1=IABS(IPP-IP)-1
            IQM1=IABS(IQP-IQ)-1
            IF (IPP.GE.IP) GO TO 370
            IF (IQP.GE.IQ) GO TO 330
            DO 320 N=1,IPM1
  320       GTYPE(IPP+N,IQP+N)=SET
            GO TO 470
  330       IF (IQP.NE.IQ) GO TO 350
            DO 340 N=1,IPM1
  340       GTYPE(IPP+N,IQ)=SET
            GO TO 470
  350       DO 360 N=1,IPM1
  360       GTYPE(IPP+N,IQP-N)=SET
            GO TO 470
  370       IF (IPP.NE.IP) GO TO 410
            IF (IQP.GE.IQ) GO TO 390
            DO 380 N=1,IQM1
  380       GTYPE(IP,IQP+N)=SET
            GO TO 470
  390       DO 400 N=1,IQM1
  400       GTYPE(IP,IQP-N)=SET
            GO TO 470
  410       IF (IQP.GE.IQ) GO TO 430
            DO 420 N=1,IPM1
  420       GTYPE(IPP-N,IQP+N)=SET
            GO TO 470
  430       IF (IQP.NE.IQ) GO TO 450
            DO 440 N=1,IPM1
  440       GTYPE(IPP-N,IQ)=SET
            GO TO 470
  450       DO 460 N=1,IPM1
  460       GTYPE(IPP-N,IQP-N)=SET
  470       SSAVE=S
            IF (J.NE.KJ2) GO TO 480
            IPP=IP
            IQP=IQ
            IP=IPN
            IQ=IQN
            R=RS1
            J=J+1
C
C***  THE DO LOOP INDEX J HAS JUST BEEN REDEFINED INSIDE THE LOOP.
C
            GO TO 290
  480   CONTINUE
        DO 540 J=1,NGRIDY
C
C***  SET INTERIOR AND EXTERIOR POINTS SCANNING LEFT TO RIGHT.
C
            FLAG=0
            SET=3
            DO 510 I=1,NGRIDX
```

```
            IF (GTYPE(I,J).EQ.0) GO TO 490
            FLAG=1
            GO TO 510
  490       IF (FLAG.EQ.0) GO TO 500
            FLAG=0
            SET=2
            IF (GTYPE(I-1,J).GE.0) SET=3
  500       GTYPE(I,J)=SET
  510    CONTINUE
C
C***  CHECK CONSISTENCY OF CONTOUR ORIENTATIONS SCANNING RIGHT TO LEFT.
C
         FLAG=0
         SET=3
      DO 540 K=1,NGRIDX
         I=NGRXP1-K
         IF (GTYPE(I,J).EQ.2) GO TO 520
         IF (GTYPE(I,J).EQ.3) GO TO 520
         FLAG=1
         GO TO 540
  520    IF (FLAG.EQ.0) GO TO 530
         FLAG=0
         SET=2
         IF (IABS(GTYPE(I+1,J)).EQ.1) SET=3
  530    IF (GTYPE(I,J).EQ.SET) GO TO 540
         WRITE (IPTR,730) I,J
         STOP 13
  540 CONTINUE
C
C***  SET THE VALUES OF THE ARRAY ΞGTYPEΞ.
C
      DO 550 J=1,NGRIDY
      DO 550 I=1,NGRIDX
         K=GTYPE(I,J)
         IF ((K.NE.2).AND.(K.NE.3)) K=1
  550 GTYPE(I,J)=LETTER(K)
      IF (LEVEL.LT.2) GO TO 570
C
C     PRINT THE REGION IDENTIFICATION GRID.
C
      WRITE (IPTR,620)
      WRITE (IPTR,740) H1,H2
      MAXAB=NGRIDX
      DO 560 K=1,NGRIDY
         J=NGRYP1-K
  560 WRITE (IPTR,750) J,(GTYPE(I,J),I=1,MAXAB)
  570 CONTINUE
      DO 580 J=1,NGRIDY
      DO 580 I=1,NGRIDX
         IF(GTYPE(I,J).EQ.LETTER(1)) GTYPE(I,J) = NUMBER(1)
         IF(GTYPE(I,J).EQ.LETTER(2)) GTYPE(I,J) = NUMBER(2)
         IF(GTYPE(I,J).EQ.LETTER(3)) GTYPE(I,J) = NUMBER(3)
  580 CONTINUE
      NGRPTS=NGRIDX*NGRIDY
      HX=FLOAT(H1)/FLOAT(H2)
      HY=HX
      AX=FLOAT(MINX1)/FLOAT(MINX2)
```

```
      BX=FLOAT(MAXX1)/FLOAT(MAXX2)
      AY=FLOAT(MINY1)/FLOAT(MINY2)
      BY=FLOAT(MAXY1)/FLOAT(MAXY2)
      DO 590 ISET=1,NGRIDX
  590 GRIDX(ISET)=AX+FLOAT(ISET-1)*HX
      DO 600 ISET=1,NGRIDY
  600 GRIDY(ISET)=AY+FLOAT(ISET-1)*HY
      RETURN
C
  610 FORMAT (16I5)
  620 FORMAT (1H1,////)
  630 FORMAT (///25X,13HCONTOUR  NO. ,I5,/)
  640 FORMAT (/,1H0,4X,25HX COORDINATES OF VERTICES)
  650 FORMAT (/,1H0,4X,25HY COORDINATES OF VERTICES)
  660 FORMAT (10X,8(1X,I4,1H/,I4,2X))
  670 FORMAT (1H2,///,10X,23HTHE COORDINATES OF THE ,I5,18H TH VERTEX ON
     1 THE ,I5,12H TH CONTOUR ,/,10X,55HDEFINES WITH THE PREVIOUS VERTEX
     2 A SEGMENT WHICH MAKES ,/,10X,68HWITH THE X-AXIS AN ANGLE OTHER TH
     3AN  N*(PI/4) WHERE N IS AN INTEGER ,//)
  680 FORMAT (1H2,///,10X,36HTOO MANY MESH POINTS IN X DIRECTION ,//)
  690 FORMAT (1H2,///,10X,36HTOO MANY MESH POINTS IN Y DIRECTION ,//)
  700 FORMAT (1H2,///,10X,20HH IS NOT ACCEPTABLE ,///)
  710 FORMAT (1H2,10X,4HTHE ,I5,18H TH VERTEX OF THE ,I5,24H TH CONTOUR
     1ALREADY SET ,///)
  720 FORMAT (1H2,10X,4HTHE ,I5,18H TH VERTEX OF THE ,I5,24H TH CONTOUR
     1GIVES R = S ,///)
  730 FORMAT (1H2,///,10X,56HINCONSISTENT CONTOUR ORIENTATION FOR MESH P
     1OINT WITH P =I3,2X,3HQ =I3)
  740 FORMAT (25X,31HTHE REGION IDENTIFICATION GRID ,//,30X,22H. ARE INT
     1ERIOR POINTS ,/,30X,22H* ARE BOUNDARY POINTS ,/,30X,28H(BLANK) ARE
     2 EXTERIOR POINTS ,//,35X,4HH = ,I4,1H/,I4,//)
  750 FORMAT (3X,I5,2X,101A1)
C
      END




      SUBROUTINE RTREG (P1,Q1,P2,Q2,RS)
      INTEGER P1,P2,Q1,Q2,RS
C
      IF (P1-P2) 10,50,90
C
C***  P1 < P2.
C
   10 IF (Q1-Q2) 20,30,40
   20 RS=5
      RETURN
   30 RS=4
      RETURN
   40 RS=3
      RETURN
C
C***  P1 = P2.
C
   50 IF (Q1-Q2) 60,70,80
```

```
   60 RS=6
      RETURN
   70 WRITE (IPTR,130)
      STOP 14
   80 RS=2
      RETURN
C
C***  P1 > P2.
C
   90 IF (Q1-Q2) 100,110,120
  100 RS=7
      RETURN
  110 RS=0
      RETURN
  120 RS=1
      RETURN
C
  130 FORMAT (1H2,10X, 42H(RTREG) TWO VERTICES HAVE THE SAME P AND Q,//)
C
      END
```

Addendum to ITPACK Report

A new version of ITPACK (August 1977) has added capabilities to the version covered in this report. These capabilities are a new solution method, Symmetric SOR Partially Adaptive (hereafter referred to as SSOR-PA), a constant coefficient switch, and an adaptive/nonadaptive switch.

SSOR-PA is similar to symmetric SOR Semi-iterative (SSOR-SI) except it applies the adaptive process only to the spectral radius, SPECR. To use SSOR-PA a good choice of CME and OMEGA must be known, *a priori*.

The constant coefficient switch is a logical, variable CONSTC. If the user wants to solve an Elliptic Partial Differential Equation with constant coefficients, then CONSTC should be set to .TRUE. in the main program. This will allow the user to set MXNCOE=1 and reduce the storage allocation by 3 full vectors. The array COEF would then contain only the right-hand side of the equation and the constant coefficients would be saved in a labeled common block.

The adaptive/non-adaptive switch is the logical variable, ADAPT. If the user already has a good parameter selection and does not wish to change parameters adaptively, ADAPT should be set to .FALSE. This switch is available in all solution methods except SSOR-PA. As SSOR-SI and SSOR-PA are identical in the non-adaptive case, this option was added only to SSOR-SI as it required less storage allocation than SSOR-PA.

Another change in the new version of ITPACK is the initialization of scalars, parameters, and switches that are needed in ITPACK. To interface with the ELLPACK Control Program these variables could not be initialized in the main program and must instead be initialized from an input file. The following variables are still initialized in the main program:

NGRDXD	CONSTC
NGRDYD	DEBUG
MXNCOE	LEVEL
MXNEQ	

The variables which are initialized from the input file are:

ITMAX	F
ZETA	CME
EPSI	SME
ADAPT	OMEGA
CASEI	SPECR

The input file for these variables consists of two lines and must be added to the end of the input file for the REGION subprogram (see Appendix 3). The first line is the same for all solution methods. ITMAX, ZETA, EPSI, ADAPT and CASEI are read in a I10, 2F10.2, 2L10 format.

The variables on the second line of the input data line depend on the solution method but all are read in 4F10.2 format. For J-SI or RS-SI the variables, in the order they appear, are F, CME, and SME. For SSOR-CG, SSOR-PA, and SSOR-SI the variables are F, CME, OMEGA, and SPECR. In the adaptive case (ADAPT=.TRUE.) RS-CG and CJ-CG need no second line, while in the non-adaptive case (ADAPT=.FALSE.) CME is read from the second data line.

A sample of the above for the SSOR-PA solution method follows.

```
Columns -      10          20          30        40        50
              100     .000001     .000001      TRUE     FALSE
              .75          .95        1.64       0.0
```

ITPACK would then set:

ITMAX=100	F=.75
ZETA=.000001	CME=.95
EPSI=.000001	OMEGA=1.64
ADAPT=.TRUE.	SPECR=0.0
CASEI=.FALSE.	

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ITR-CNA-126	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): ITPACK Report: Numerical Studies of Several Adaptive Iterative Algorithms	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): David R. Kincaid and Roger G. Grimes	8. CONTRACT OR GRANT NUMBER(s): DAHC04 74 G 0198
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center ror Numerical Analysis, The University of Texas at Austin, Austin, Texas 78712	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: P-12301-M
11. CONTROLLING OFFICE NAME AND ADDRESS: The University of Texas at Austin, Austin, Texas 78712	12. REPORT DATE: August 1977
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

ITPACK, ELLPACK, elliptic partial differential equation, iterative algorithm, adaptive parameter determination

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Six adaptive iterative algorithms are studied for six elliptic partial differential equations on six regions compatible with subroutine REGION. An effort was made to make the resulting preliminary ITPACK code conform to the "ELLPACK Contributor's Guide--Initial Version," CSD TR 208, Purdue University, November 1, 1976.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE