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 DAH004-74-G-0198 from the Army Research Office and grant MCS76-03141 from the National Science Foundation at The University of Texas at Austin.
Table of Contents

1. Introduction.................................................................1
2. Iterative Methods..........................................................2
3. ITPACK Structure and Use.................................................11
4. Test Problems and Regions..............................................15
5. Numerical Results.........................................................19

References............................................................................29

Appendix 1 - Description of Adaptive Procedures (Equations,
Flow Chart, and Algorithm)....................................................30
I. J-SI: Jacobi Semi-iterative.................................................31
II. CJ-CG: Compressed Jacobi Conjugate Gradient...............36
III. RS-SI: Reduced System Semi-iteration............................41
IV. RS-CG: Reduced System Conjugate Gradient....................46
V. SSOR-SI: Symmetric Successive Overrelaxation
   Semi-iteration.............................................................51
VI. SSOR-CG: Symmetric Successive Overrelaxation
    Conjugate Gradient...................................................60

Appendix 2 - Sample Problem...............................................69

Appendix 3 - The Subroutine REGION....................................74

Addendum to ITPACK Report.................................................88
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

\[(a_{uu})_x + (cu_y)_y + fu = g\]

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

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

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

\[
(au_x)_x |_{(i,j)} \approx (au_x)_{(i,\frac{1}{2},j)} - (au_x)_{(i-\frac{1}{2},j)} h^{-1}
\]

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

Hence, we use

\[
(au_x)_x |_{(i,j)} \approx \{a_{i+\frac{1}{2},j}u_{i+1,j} + a_{i-\frac{1}{2},j}u_{i-1,j} - (a_{i+\frac{1}{2},j} + a_{i-\frac{1}{2},j})u_{ij}\} h^{-2}
\]

Similarly, we use

\[
(cu_y)_y |_{(i,j)} \approx \{c_{i,j+\frac{1}{2}}u_{i,j+1} + c_{i,j-\frac{1}{2}}u_{i,j-1} - (c_{i,j+\frac{1}{2}} + c_{i,j-\frac{1}{2}})u_{ij}\} h^{-2}
\]

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

\begin{equation}
S_{ij}u_{i,j-1} - W_{ij}u_{i-1,j} + C_{ij}u_{ij} - K_{ij}u_{i+1,j} - K_{ij}u_{i,j+1} + K_{ij}u_{i,j+1} - K_{ij} = \eta
\end{equation}
where
\[
\begin{align*}
S_{ij} &= c_{i,j-\frac{1}{2}} \\
W_{ij} &= a_{i-\frac{1}{2},j} \\
C_{ij} &= (a_{i+\frac{1}{2},j} + a_{i-\frac{1}{2},j}) + (c_{i,j+\frac{1}{2}} + c_{i,j-\frac{1}{2}}) - h^2 f_{ij} \\
E_{ij} &= a_{i+\frac{1}{2},j} \\
N_{ij} &= c_{i,j+\frac{1}{2}} \\
R_{ij} &= -h^2 g_{ij}
\end{align*}
\]

Equation (2.2) can be illustrated by the following stencils:

\[\text{@}(i,j): \quad \begin{array}{c}
-N_{ij} \\
-W_{ij} \\
C_{ij} \\
E_{ij} \\
=S_{ij} \\
=R_{ij}
\end{array}\]

where
From (2.3), we have the following symmetry condition

\[
\begin{align*}
W_{ij} &= E_{i-1,j} \\
S_{ij} &= N_{i,j-1}
\end{align*}
\]

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

\[
\begin{align*}
-N_{i,j-1}u_{i,j-1} - E_{i-1,j}u_{i-1,j} + C_{ij}u_{ij} \\
-E_{ij}u_{i+1,j} - N_{ij}u_{i,j+1} &= R_{ij}
\end{align*}
\]

(2.4)

and

\[
\begin{align*}
&\text{Since only regular grid points are considered, we have for the basic linear equation}
\end{align*}
\]

\[
u_{ij} = (E_{ij}u_{i+1,j} + N_{ij}u_{i,j+1} + E_{i-1,j}u_{i-1,j} \\
+ N_{i-1,j}u_{i-1,j} + R_{ij})/C_{ij}
\]

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

\[ Au = b \]

and

\[ (2.6) \quad u = Bu + c \]

respectively, where \( D^{-1}A = I - B \) and \( D = \text{diag}(C_{ij}) \). 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 reduced system

\[ (2.7) \quad u_B = F_B F_R u_B + (F_B c_R + c_B) \]

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

\[ (2.8) \quad u_{ij}^{(n+1)} = E_{ij} u_{i+1,j}^{(n)} + N_{ij} u_{i,j+1}^{(n)} + E_{i-1,j} u_{i-1,j}^{(n)} + N_{i,j-1} u_{i,j-1}^{(n)} + R_{ij} \frac{1}{C_{ij}} \]
or in matrix form

\begin{equation}
\mathbf{u}(n+1) = \mathbf{Bu}(n) + \mathbf{c}
\end{equation}

For the reduced system, the basic iterative equation would be

\begin{equation}
\mathbf{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})} + \mathbf{R}_{ij}/C_{ij})
\end{equation}

where

\begin{equation}
\mathbf{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)} + \mathbf{R}_{k,\ell}/C_{k,\ell})
\end{equation}

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

The basic iterative equation for the resulted system is

\begin{equation}
\mathbf{u}_B^{(n+1)} = \mathbf{F_R F_C u}_B^{(n)} + \mathbf{F_R c}_R + \mathbf{c}_B
\end{equation}

However, it is easier to consider this as two separate iterations.
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 \( \omega \) 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, \( A \), 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 are

\[
\begin{align*}
u_{ij}^{(n+\frac{1}{2})} &= \omega(E_{ij}u_{i+1,j}^{(n)} + N_{ij}u_{i,j+1}^{(n)} + E_{i-1,j}u_{i-1,j}^{(n)} + N_{i,j-1}u_{i,j-1}^{(n)} + R_{ij})/C_{ij} + (1-\omega)u_{ij}^{(n)} \\
u_{ij}^{(n+1)} &= \omega(E_{ij}u_{i+1,j}^{(n+1)} + N_{ij}u_{i,j+1}^{(n+1)} + E_{i-1,j}u_{i-1,j}^{(n+1)} + N_{i,j-1}u_{i,j-1}^{(n+1)} + R_{ij})/C_{ij} + (1-\omega)u_{ij}^{(n+\frac{1}{2})}
\end{align*}
\]

This can be written in matrix form as
\[ u(n + \frac{1}{2}) = \phi_{\omega} u(n) + k_{\omega} \]
\[ u(n + 1) = \phi_{\omega} u(n + \frac{1}{2}) + k_{\omega} \]

or

\[ u(n + 1) = \phi_{\omega} u(n) + k_{\omega} \]

where

\[
\begin{align*}
\phi_{\omega} &= \phi_{\omega} l_{\omega} \\
k_{\omega} &= \phi_{\omega} k_{\omega} + k_{\omega}
\end{align*}
\]

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

\[ u(n + 1) = \gamma u(n) + k \]

where

\[
\begin{align*}
\gamma &= B \\
\nu &= c \\
F_{B}F_{R} &= F_{B}c_{R} + c_{E}
\end{align*}
\]

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

\[ \delta(n) = \gamma u(n) + k - u(n) \]

Here \( \rho_{n+1} \) and \( \gamma_{n} \) 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 Overrelaxation 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 performs 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:

\[
\begin{align*}
\text{COEF(IJ,1)} & = \text{center coefficient at (I,J)} \\
\text{COEF(IJ,2)} & = \text{north coefficient at (I,J)} \\
\text{COEF(IJ,3)} & = \text{east coefficient at (I,J)} \\
\text{COEF(IJM1,2)} & = \text{south coefficient at (I,J)}
\end{align*}
\]
COEF(IM1J,3) = west coefficient at (I,J)
COEF(IJ,4) = right-hand side at (I,J)

where

\[
\begin{align*}
I & = I + (J-1) \times NGRIDX \\
IJM1 & = I + (J-2) \times NGRIDX \\
IM1J & = (I-1) + (J-1) \times NGRIDX
\end{align*}
\]

The basic iterative equation then becomes

\[
U(IJ) = (COEF(IJ,3) \times U(IP1J) + COEF(IJ,2) \times U(IJP1) + COEF(IM1J,3) \times U(IM1J) + COEF(IJM1,2) \times U(IJM1) + COEF(IJ,4)) / COEF(IJ,1)
\]

where

\[
\begin{align*}
IP1J & = (I+1) + (J-1) \times NGRIDX \\
IJP1 & = I + J \times NGRDIX
\end{align*}
\]

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
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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL</td>
<td>Controls output of REGION</td>
</tr>
<tr>
<td>NGRDXD</td>
<td>Maximum number of grid points in the x direction.</td>
</tr>
<tr>
<td></td>
<td>Also the first dimension of GTYPE, and dimension of GRIDX.</td>
</tr>
<tr>
<td>NGRDYD</td>
<td>Maximum number of grid points in the y direction.</td>
</tr>
<tr>
<td></td>
<td>Also the second dimension of GTYPE and dimension of GRIDY.</td>
</tr>
<tr>
<td>MXNCOE</td>
<td>Set equal to 4 for five-point stencil. Later a value of 6 will indicate a</td>
</tr>
<tr>
<td></td>
<td>nine-point stencil.</td>
</tr>
<tr>
<td>MXNEQ</td>
<td>Dimension of UNKNWN and first dimension of COEF.</td>
</tr>
<tr>
<td></td>
<td>Commonly taken to be NGRDXD*NGRDYD.</td>
</tr>
<tr>
<td>ITMAX</td>
<td>Upper bound on number of iterations the user will allow the method to take</td>
</tr>
<tr>
<td></td>
<td>before convergence. If ITMAX is reached, the method will stop and exit</td>
</tr>
<tr>
<td></td>
<td>naturally. Note that the stopping criteria may not be satisfied.</td>
</tr>
<tr>
<td>ZETA</td>
<td>Tolerance level in stopping test (usually $10^{-6}$).</td>
</tr>
<tr>
<td>EPSI</td>
<td>Tolerance level in root solving and checks in division by zero. (usually</td>
</tr>
<tr>
<td></td>
<td>$10^{-6}$).</td>
</tr>
</tbody>
</table>
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 auxiliary storage utilized in the iterative algorithms. At present, the workspace array WORKSP must be dimensioned as follows for each iterative method:

<table>
<thead>
<tr>
<th>Method</th>
<th>WORKSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>J-SI</td>
<td>3*MXNEQ</td>
</tr>
<tr>
<td>CJ-CG</td>
<td>3*MXNEQ + 200</td>
</tr>
<tr>
<td>RS-SI</td>
<td>2*MXNEQ</td>
</tr>
<tr>
<td>RS-CG</td>
<td>4*MXNEQ + 200</td>
</tr>
<tr>
<td>SSOR-SI</td>
<td>5*MXNEQ</td>
</tr>
<tr>
<td>SSOR-CG</td>
<td>6*MXNEQ + 200</td>
</tr>
</tbody>
</table>
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 interior of \( R \) was selected to be identically zero.

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

The test problems are as follows:

(1) \[ \nabla^2 u = f \]

where

\[ f = 6xye^{x+y}(x+y+1) \]
\[ u_{\text{true}} = 3xye^{x+y}(x-1)(y-1) \]

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

where

\[ f = \pi\{x \sin(\pi x) \cos(\pi y) + 3ye^{2xy} \cos(\pi x) \sin(\pi y)\} \]
\[ + \sin(\pi x) \sin(\pi y)\{(2y^2-\pi^2)e^{2xy} - \pi^2 - e^{xy}/(1+x+y)\} \]
\[ u_{\text{true}} = e^{xy}\sin(\pi y) \sin(\pi x). \]
(3) \[ \nabla^2 [(1 + \sin(\frac{\pi}{2} (x+y)) \nabla u)] = f \]

where

\[
\begin{align*}
    f &= 8 \left[ 1 + \sin(\frac{\pi}{2} (x+y)) \right] \\
    &\quad + \pi^2 \sin(\frac{\pi}{2} (x+y)) \left[ (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \right] \\
    \end{align*}
\]

\[
\begin{align*}
    u_{\text{true}} &= 2 \left[ (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \right] / \left[ 1 + \sin(\frac{\pi}{2} (x+y)) \right].
\end{align*}
\]

(4) \[ \nabla^2 u = f \]

where

\[
\begin{align*}
    f &= 8 \left( x^2 + y^2 - x - y \right) \\
    u_{\text{true}} &= 4xy(x-1)(y-1)
\end{align*}
\]

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

where

\[
\begin{align*}
    f &= 300 \cosh(20y) / \cosh(20) \\
    u_{\text{true}} &= \cosh(10x) / \cosh(10) + \cosh(20y) / \cosh(20)
\end{align*}
\]

(6) \[ (A(x)u_x)_x + (C(y)u_y)_y = f \]

where

\[
\begin{align*}
    f &= \begin{cases} 
        (2+x)e^x - \pi^2 (1+y) \sin(\pi y) + \cos(\pi y), & x, y \in [0, \frac{1}{2}] \\
        (1-x)e^x - \pi^2 (2-y) \sin(\pi y) - \cos(\pi y), & x, y \in [\frac{1}{2}, 1]
    \end{cases} \\
    u_{\text{true}} &= e^x + \sin(\pi y)
\end{align*}
\]

and \[ u_{\text{true}} = \]
The six test regions selected are as follows.

(1) \quad (2)

\begin{align*}
A(x) &= \begin{cases} 
1+x, & x \in [0, \frac{1}{2}] \\
2-x, & x \in (\frac{1}{2}, 1]
\end{cases} \\
C(x) &= \begin{cases} 
1+y, & y \in [0, \frac{1}{2}] \\
2-y, & y \in (\frac{1}{2}, 1]
\end{cases}
\end{align*}
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:

\[
F = .75, \quad CME = 0.0, \\
EPSI = .000001, \quad SME = 0.0, \\
ZETA = .000001, \quad CASE = .FALSE.
\]

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

where
A = Number of iterations until convergence

\[ B = \frac{\|u^\text{true} - u^\text{computed}\|_D}{\|u^\text{true}\|_D} \]

For J-SI, RS-SI, SSOR-SI, SSOR-CG, a list of
iterations numbers where parameters were changed.

C = \{ \begin{align*}
\text{For CJ-CG and RS-CG, the iteration number where} \\
\text{a new estimate of CME was last calculated}
\end{align*} \}

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

\[ z(x,y) = \frac{|u^{\text{true}}(x,y) - u^{\text{computed}}(x,y)|}{\text{SCALE}} \]

where

\[ \text{SCALE} = \max_{x,y} |u^{\text{true}}(x,y) - u^{\text{computed}}(x,y)|. \]

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

Table 1. Test Problems Over the Unit Square With h = 1/40
<table>
<thead>
<tr>
<th>Region</th>
<th>J-SI</th>
<th>CJ-CG</th>
<th>RS-SI</th>
<th>RS-CG</th>
<th>SSOR-SI</th>
<th>SSOR-CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>120</td>
<td>1.61596 x 10^-3</td>
<td>58</td>
<td>1.61685 x 10^-3</td>
<td>57</td>
<td>1.61597 x 10^-3</td>
</tr>
<tr>
<td></td>
<td>0,3,10,98</td>
<td>16</td>
<td>0,2,7</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.98728702</td>
<td>0.9872099</td>
<td>0.9867997</td>
<td>0.9872053</td>
<td>0.980064</td>
<td>0.9792514</td>
</tr>
<tr>
<td>(2)</td>
<td>90</td>
<td>8.47542 x 10^-4</td>
<td>52</td>
<td>8.48297 x 10^-4</td>
<td>46</td>
<td>8.47410 x 10^-4</td>
</tr>
<tr>
<td></td>
<td>0,2,4,7,12,23</td>
<td>22</td>
<td>0,1,2</td>
<td>3,5,8,23</td>
<td>11</td>
<td>0,2,7</td>
</tr>
<tr>
<td></td>
<td>0.9787733</td>
<td>0.9791823</td>
<td>0.9792082</td>
<td>0.9791827</td>
<td>0.9784321</td>
<td>0.9701817</td>
</tr>
<tr>
<td>(3)</td>
<td>76</td>
<td>3.61644 x 10^-4</td>
<td>50</td>
<td>3.62336 x 10^-4</td>
<td>42</td>
<td>3.61739 x 10^-4</td>
</tr>
<tr>
<td></td>
<td>0,2,4,7,12,37</td>
<td>22</td>
<td>0,1,2,4,7,31</td>
<td>11</td>
<td>0,2,12</td>
<td>0,1,2,4</td>
</tr>
<tr>
<td></td>
<td>0.9709087</td>
<td>0.9709838</td>
<td>0.9710263</td>
<td>0.9709836</td>
<td>0.9701155</td>
<td>0.9594277</td>
</tr>
<tr>
<td>(4)</td>
<td>62</td>
<td>1.04289 x 10^-4</td>
<td>40</td>
<td>1.04816 x 10^-4</td>
<td>32</td>
<td>1.04207 x 10^-4</td>
</tr>
<tr>
<td></td>
<td>0,2,5,9,16</td>
<td>22</td>
<td>0,1,1,6,17</td>
<td>11</td>
<td>0,2</td>
<td>0,1,3,6</td>
</tr>
<tr>
<td></td>
<td>0.9562094</td>
<td>0.9572589</td>
<td>0.9571162</td>
<td>0.9572589</td>
<td>0.9324005</td>
<td>0.9384010</td>
</tr>
<tr>
<td>(5)</td>
<td>97</td>
<td>1.09554 x 10^-3</td>
<td>60</td>
<td>1.09662 x 10^-3</td>
<td>51</td>
<td>1.055806 x 10^-3</td>
</tr>
<tr>
<td></td>
<td>0,2,4,6,8,10,14,26</td>
<td>22</td>
<td>0,1,2,3,4,6,10</td>
<td>11</td>
<td>0,1,2</td>
<td>0,1,2,4</td>
</tr>
<tr>
<td></td>
<td>0.9825412</td>
<td>0.9826603</td>
<td>0.9822498</td>
<td>0.9826603</td>
<td>0.9791120</td>
<td>0.9730194</td>
</tr>
<tr>
<td>(6)</td>
<td>41</td>
<td>1.99813 x 10^-4</td>
<td>30</td>
<td>2.00549 x 10^-4</td>
<td>21</td>
<td>2.00224 x 10^-4</td>
</tr>
<tr>
<td></td>
<td>0,2,5,11</td>
<td>20</td>
<td>0,1,3,9</td>
<td>10</td>
<td>0,2</td>
<td>0,1,3</td>
</tr>
<tr>
<td></td>
<td>0.9068176</td>
<td>0.9093197</td>
<td>0.90877104</td>
<td>0.9093297</td>
<td>0.8958033</td>
<td>0.8558511</td>
</tr>
</tbody>
</table>

Table 2. Test Problem (2) With h = 1/20.
Figure 1. SSOR-SI method error distribution at convergence (SCALE = 7.529572 x 10^{-8})
Figure 2. SSOR-CG method error distribution at convergence
(Scale = $1.583368 \times 10^{-8}$)
Figure 3. RS-SI method error distribution at convergence

(Scale = 2163014 x 10^{-7})
Figure 4. RS-CG method error distribution after 5 iterations
(Scale = 3.792600 x 10^{-3})
Figure 5. RS-CG method error distribution after 5 iterations
(Scale = 3.792600 x 10^{-3})
Figure 6. RS-CG method error distribution after 10 iterations (SCALE = 2.275237 x 10^-6)
References


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,
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

\[
\gamma = \frac{2}{(2-M_{E_m})}, \quad \sigma_E = \frac{(M_m-E_m)/(2-M_{E_m})}{E_m},
\]

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

(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) Residual Vector

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

(4) Iteration Vector

\[
u(s+1) = \gamma \delta(s) + u(s), \quad n = s
\]

\[
u(n+1) = \rho_{n+1} \{\gamma \delta(n) + u(n)\} + (1 - \rho_{n+1}) u(n-1), \quad n > s+1
\]

(5) Stopping Test

\[
d(n) = \delta(n)^T \ast D \ast \delta(n)
\]

\[
STEST = \left[\frac{1}{(1-M_{E_m})}\right][d(n)/\langle u(n)^T \ast D \ast u(n) \rangle]^{1/2}
\]

If STEST < \(\zeta\), then exit.
6. Changing Parameter Test

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

\[
QT = 2r(n-s)/(1+r(n-s))
\]

If \( QA \geq QT \), then change parameters.

7. Rayleigh Quotient Vector

\[
\tilde{u}(n+1) = u(n) + \delta(n)
\]

\[
\tilde{v}(n+1) = B^{*}\tilde{u}(n+1) + c - \tilde{u}(n+1)
\]

8. Computing new \( M_E \) and \( m_E \)

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

\[
X = Z^{1/(n-s)}
\]

\[
\alpha = (X+r/X)/(1+r)
\]

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

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

\[
M_E = \max\{M_1, M_2\}
\]

\[
M_E = \begin{cases} 
\text{not changed, if case I} \\
-M_E, & \text{if case II}
\end{cases}
\]
Flow Chart 1: J-SI Method
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 $N \neq S+1$, then compute $\delta(n)=B*u(n)+c-u(n)$

else compute $\delta(n)=\text{GAMMA} \cdot \tilde{\delta}(n) + (1-\text{GAMMA}) \cdot \delta(n-1)$

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

If $STEST < ZETA$, then go to [TEST]

(Test for changing parameters)

If $N=0$, then go to [CHANGE]

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

$QT:=2P/((1+R^P))$

If $QA \geq QT^F$, 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 $C1:=RHO \cdot \text{GAMMA}, C2:=RHO, C3:=1-RHO,$

$u(n+1)=C1 \cdot \delta(n)+C2 \cdot u(n)+C3 \cdot u(n-1)$

Go to [ENDIT]
J-SI (continued)

**CHANGE**

(Change parameters)

Compute \( \vec{u}_n(n+1) = u_n(n) + \delta(n) \)

\( \vec{\delta}(n+1) = B\vec{u}_n(n+1) + C - \vec{u}_n(n+1) \)

If \( N=0 \), then set \( ZM1 := CME \),
else compute \( Z := (1+R^P)(QA+(QA^2-Q^2T)^2)/2 \),
\( X := Z^{1/P} \)
\( SIG := (X+R/X)/(1+R) \)
\( ZM1 := (CME+SME+SIG\cdot(2-CME-SME))/2 \)

If CASE = .TRUE., then compute \( ZM2 := \delta(n)^T \cdot D \cdot \delta(n+1)/DELNRM \)
else compute \( ZM2 := \delta(n+1)^T \cdot D \cdot \delta(n+1)/DELNRM \)

Set \( CME := \max\{ZM1, ZM2\} \)
If CASE = .FALSE., then set \( SME := -CME \)
Compute \( SIGE := (CME-SME)/(2-CME-SME) \)
\( GAMMA := 2/(2-CME-SME) \)
\( R := (1-(1-SIGE^2)^{1/2})/(1+(1+SIGE^2)^{1/2}) \)
Set \( S := N \)
\( DELSRM := DELNRM \)
\( RHO := 1 \)
Print \( N, ZM1, ZM2, CME \)
Compute \( u(n+1) = GAMMA \cdot \delta(n) + u(n) \)

**END**
(End of iteration step)
Print \( N, UNRM, STEST, QA, QT, CME, RHO, GAMMA \)
Set \( N := N+1 \)
Go to **START**

**EXIT**
(Exit iteration algorithm)
Compute \( UNRM := u(n)^T \cdot D \cdot u(n) \)
Print \( N, UNRM \)
If LEVEL > 2, print \( u(n) \)

**END**
II. CJ-CG: Compressed Conjugate Gradient Equations

(1) Residual Vector (non-recursive computation)
\[
\begin{align*}
\mathbf{u}_R^{(n)} &= F_R \mathbf{u}_B^{(n)} + c_R \\
\delta_B^{(n)} &= F_B \mathbf{u}_R^{(n)} + c_B - \mathbf{u}_B^{(n)}
\end{align*}
\]

(2) Acceleration Parameters
\[
\begin{align*}
\delta_R^{(n+1)} &= \begin{cases} 
F_R \delta_B^{(0)}, & n = 0 \\
\rho_{n+1} F_R \delta_B^{(n)} + (1-\rho_{n+1}) \delta_R^{(n-1)}, & n > 0
\end{cases} \\
d_R^{(n)} &= \delta_R^{(n)} * \delta_R^{(n)} \\
d_B^{(n)} &= \delta_B^{(n)} * \delta_B^{(n)}
\end{align*}
\]
\[
\begin{align*}
\rho_{n+1} &= \begin{cases} 
1, & n = 0 \\
\frac{1}{1 - (1/\rho_n) (d_B^{(n)}/d_R^{(n-1)})}, & n > 0
\end{cases} \\
\hat{\rho} &= (\rho_{n+2}/\rho_n) (1-\rho_{n+1}) (1-\rho_n) \\
\hat{\gamma} &= (\rho_{n+2} \rho_{n+1}/\hat{\rho})
\end{align*}
\]

(3) Iteration Vector
\[
\begin{align*}
\mathbf{u}_B^{(n+2)} &= \begin{cases} 
\phi_B^{(0)} + \mathbf{u}_B^{(0)}, & n = 0 \\
\rho \mathbf{u}_B^{(n)} + (1-\hat{\rho}) \mathbf{u}_B^{(n-2)}, & n > 0
\end{cases}
\end{align*}
\]

(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 $n \times n$ tridiagonal matrix ($1 \leq i \leq n$)

$$
\begin{bmatrix}
\left(\frac{\rho_i-1}{\rho_i-1\rho_i}\right)^{1/2}, & 0, & \left(\frac{\rho_{i+1}-1}{\rho_{i+1}\rho_{i+1}}\right)^{1/2}
\end{bmatrix}
$$

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

If $\text{STEST} \leq 6$, then exit.

Note: If $\frac{|M_E^{(\text{new})} - M_E^{(\text{old})}|}{M_E^{(\text{new})}} < \epsilon$, then an acceptable estimate of $M_E$ has been obtained and computation 5a is omitted.
Flow Chart 2: CJ-CG Method

Start

N := 0

Compute Residual Vector (Non-recursively) (1)

Stopping Test (5)

yes

Compute Acceleration Parameters (2)

no

Compute Iteration Vector (3)

Compute Residual Vector (Recursively) (4)

N := N + 1

Input

u(0)

ε, ζ

\[ u_R^{(n)} = F_R \cdot u_B^{(n)} + c_R \]

Exit
CJ-CG: Compressed Jacobi Conjugate Gradient Algorithm

Input \( u(0), \text{EPSI}, \text{ZETA}, \text{ITMAX}, \text{LEVEL} \)

Set \( N := 0 \)

Compute \( CNRM := k^T_B * D_B * k_B \)

If \( CNRM < \text{EPSI} \), then go to Exit

**START**

If \( N > \text{ITMAX} \), then go to Exit

If \( N < 4 \), then go to One

If \( |\text{CME} - \text{CMOLD}| / \text{CME} < \text{EPSI} \), then go to Two

**One**

(Determine new CME)

Set \( \text{CMOLD} := \text{CME} \)

If \( N = 0 \), then set \( \text{CME} := 0 \)

Else set \( \text{CME} := \text{maximum eigenvalue of the tri-diagonal matrix} \)

\[
\{(\text{RHO}_i - 1)/(\text{RHO}_i \cdot \text{RHO}_i - 1)\}^{1/2}, 0, \{(\text{RHO}_{i+1} - 1)/(\text{RHO}_{i+1} \cdot \text{RHO}_{i+1} - 1)\}^{1/2} \}
\]

for \( 1 \leq i \leq N \)

**Two**

(Test for stopping)

Compute \( \text{UNRM} := u_B^T * D_B \cdot u_B \)

\( \text{DELNRM} := \delta_B^T * D_B \cdot \delta_B \)

If \( \text{UNRM} < \text{CNRM} \), then set \( \text{UNRM} := \text{CNRM} \)

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

If \( \text{STEST} < \text{ZETA} \), then go to Exit

If \( N = 0 \), then set \( \text{RHO}_{N+1} := 1 \)

else compute \( \text{RHO}_{N+1} := 1 / (1 - \text{DELNRM} / (\text{DELSRM} \cdot \text{RHO}_N)) \)

Set \( C1 := \text{RHO}_{N+1}, C1 := 1 - \text{RHO}_{N+1} \)

Compute:

\( v_R = \{-u_R^B \} \)

\( \delta_{(N+1)} = C1 \cdot v_R + C2 \cdot \delta_{(N-1)} \)

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

\( \text{RHO}_{N+2} := 1 / (1 - \text{DELSRM} / (\text{DELNRM} \cdot \text{RHO}_{N+1})) \)
CJ-CG (continued)

If N=0, then set RHOHAT:=1,
else compute RHOHAT:=1+RHO\textsuperscript{N+2}\cdot\left(1-RHO\textsuperscript{N+1}\right)/RHO\textsuperscript{N+1}/RHOHAT
Compute GAMMA:=RHO\textsuperscript{N+2}\cdot RHO\textsuperscript{N+1}/RHOHAT
Set C1:=RHOHAT\cdot 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 \(v_B=F_B\cdot\delta^{(N+1)}\)
Set C1:=RHO\textsuperscript{N+2}, C2:=1-RHO\textsuperscript{N+2}
Compute \(\delta_B^{(N+2)}=C1\cdot v_B+C2\cdot\delta_B^{(N)}\)
Print N, UNRM\textsuperscript{1/2}, STEST, CME, RHO, GAMMA
Set N=N+2
Go to \textbf{START}

\textbf{EXIT}
Compute \(u_R^{(N)}=F_R^{T}u_B^{(N)}+C_R\)
UNRM:=\(u^{(N)}\cdot D\cdot u^{(N)}\)
Print N, UNRM\textsuperscript{1/2}
\textbf{IF LEVEL > 2, then print }u^{(N)}
\textbf{END}
III. RS-SI: Reduced System Semi-iterative Equations

(1) Adaptive Parameters

\[ \gamma = \frac{2}{(2-M_E^2)}, \quad \sigma_E = \frac{M_E^2}{(2-M_E^2)}, \]
\[ r = \frac{1 - [1-M_E^2]^{1/2}}{1 + [1-M_E^2]^{1/2}} \]

(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) Residual Vector

\[ u_R^{(n)} = F_R u_B^{(n)} + c_R \]
\[ \delta_B^{(n)} = F_B u_R^{(n)} + c_B - u_B^{(n)} \]

(4) Iteration Vector

\[ u_B^{(s+1)} = \gamma \delta_B^{(s)} + u_B^{(s)} \]
\[ u_B^{(n+1)} = \rho_{n+1} \{ \gamma \delta_B^{(n)} + u_B^{(n)} \} + (1-\rho_{n+1}) u_B^{(n-1)}, \quad n \geq s+1 \]

(5) Stopping Test

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

If STEST < \( \xi \), then exit.
RS-SI (continued)

(6) Changing Parameters Test

\[ QA = \left( \frac{d_B(n)}{d_B(s)} \right)^{1/2} \]

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

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

(7) Computing new \( M_E \)

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

\[ X = Z^{1/(2(n-s))} \]

\[ \sigma = (X-r/X)/(1+r) \]

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

\[ \delta_R(n) = F \ast \delta(n) \]

\[ d_R(n) = \delta_R(n)^T \ast \delta_R(n)^* \]

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

\[ M_E = \max\{M_E, M_1, M_2\} \]
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 $CNRM:=y^TK^T B k_B$

If $CNRM < EPSI$, then go to EXIT

START

IF $N > ITMAX$, go to EXIT

Compute $u_{(N)}^R = F^R * u_{(N)}^B + c_R$
$\delta_{(N)}^B = F_B^* u_{(N)}^R + c_B - u_{(N)}$

(Test for stopping)
Compute $UNRM:=u_{(N)}^T B^* d_B^* u_{(N)}$
$DELNRM:=\delta_{(N)}^T B^* d_B^* \delta_{(N)}$

If $UNRM < CNRM$, then set $UNRM:=CNRM$

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

(Test for changing parameters)
If $N=0$, then go to CHANGE

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

(Perform iteration with current parameters)
If $P=1$, then compute $RHO:=1/(1-SIGE^2/2)$,
else compute $RHO:=1/(1-RHO \cdot SIGE^2/4)$
Set $C1:=RHO \cdot GAMMA, C2:=RHO, C3:=1-RHO$

Compute $u_{(N+1)}^B = C1 \cdot \delta_{(N)}^B + C2 \cdot u_{(N)}^B + C3 \cdot u_{(N-1)}^B$

Go to ENDIT
RS-SI (continued)

(Change parameters)

If \( N = 0 \), then set \( ZM1 := CME \),
else compute

\[
Z := \frac{(1+R^2P)(QA+(QA^2-QT^2)^{1/2})}{2}
\]

\[
X := Z^{1/(2P)}
\]

\[
ZM1 := \frac{(X+R/X)/(1+R)}{2}
\]

\[
\nu := \frac{F^*5(N)}{R^B}
\]

\[
ZM2 := \left(\frac{\nu^T \ast D \ast \nu}{\text{DELNRM}}\right)^{1/2}
\]

Compute \( CME := \max\{ZM1, ZM2\} \)

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

\[
\text{GAMMA} := 2/(2-CME^2)
\]

\[
R := \left(1-(1-CME^2)^{1/2}\right)/(1+(1-CME^2)^{1/2})
\]

Set \( S := N \)

\[
\text{DELSRM} := \text{DELNRM}
\]

\[
\text{RHO} := 1
\]

Print \( N, ZM1, ZM2, CME \)
Compute \( u^{(N+1)} := \text{GAMMA} \cdot \delta(N) + u^{(N)} \)

END

Print \( N, \text{UNRM}^{1/2}, \text{TEST}, QA, QT^F, CME, \text{RHO}, \text{GAMMA} \)
Set \( N := N+1 \)
GO to START

EXIT

Compute \( u^R := F^*u_B^R + c_R \)

\[
\text{UNRM} := u^R \ast D_R \ast u^R
\]

PRINT \( N, \text{UNRM}^{1/2} \)
If \( \text{LEVEL} > 2 \), print \( u^{(N)} \)

END
IV. RS-CG: Reduced System Conjugate Gradient Equations

(1) Residual Vector (non-recursive computation)

\[
\begin{align*}
\mathbf{u}_R^{(n)} &= \mathbf{F}_R \mathbf{u}_R^{(n)} + \mathbf{c}_R \\
\mathbf{\delta}_B^{(n)} &= \mathbf{F}_B \mathbf{u}_R^{(n)} + \mathbf{c}_B - \mathbf{u}_B^{(n)}
\end{align*}
\]

(2) Acceleration Parameters

\[
\begin{align*}
\mathbf{v}_R &= \mathbf{F}_R \mathbf{\delta}_B^{(n)} \\
\mathbf{v}_B &= \mathbf{F}_B \mathbf{\delta}_B^{(n)} \\
\mathbf{d}_B(n) &= \mathbf{\delta}_B^{(n)} \mathbf{D}_B \mathbf{\delta}_B^{(n)} \\
\gamma_{n+1} &= \frac{1}{1 - (\mathbf{\delta}_B^{(n)} \mathbf{D}_B \mathbf{\delta}_B^{(n)}) / \mathbf{d}_B(n)} \\
\rho_{n+1} &= \begin{cases} 
1, & n = 0 \\
1 / \left[ 1 - \frac{(\gamma_{n+1})}{(\gamma_n \rho_n)} \left( \frac{\mathbf{d}_B(n)}{\mathbf{d}_B(n-1)} \right) \right], & n > 0
\end{cases}
\]

(3) Iteration Vector

\[
\begin{align*}
\mathbf{u}_B^{(n+1)} &= \begin{cases} 
\gamma_1 \mathbf{\delta}_B^{(n)} + \mathbf{u}_B^{(n)}, & n = 0 \\
\rho_{n+1} \left( \gamma_{n+1} \mathbf{\delta}_B^{(n)} + \mathbf{u}_B^{(n)} \right) + (1 - \rho_{n+1}) \mathbf{u}_B^{(n-1)}, & n > 0
\end{cases}
\]

(4) Residual Vector (recursive computation)

\[
\begin{align*}
\mathbf{\delta}_B^{(n+1)} &= \begin{cases} 
\gamma_1 \mathbf{v}_B + (1 - \gamma_1) \mathbf{\delta}_B^{(0)}, & n = 0 \\
\rho_{n+1} \left( \gamma_{n+1} \mathbf{v}_B + (1 - \gamma_{n+1}) \mathbf{\delta}_B^{(n)} \right) + (1 - \rho_{n+1}) \mathbf{\delta}_B^{(n-1)}, & n > 0
\end{cases}
\]
RS-CG (continued)

(5) Stopping Test

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

$$
\begin{bmatrix}
\left( \frac{\rho_i - 1}{\sqrt{y_{i-1} \rho_i - 1 y_{i+1} \rho_i}} \right), & \left( 1 - \frac{1}{y_i} \right), & \left( \frac{\rho_{i+1} - 1}{\sqrt{y_{i+1} \rho_i - 1 y_{i+1} \rho_{i+1}}} \right)^{1/2}
\end{bmatrix}
$$

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

If $\text{STEST} \leq \rho$, then exit.

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

N := 0

Compute Residual Vector (Non-recursively) (1)

Stopping Test (5)

yes

Compute Acceleration Parameters (2)

Compute Iteration Vector (3)

Compute Residual Vector (Recursively) (4)

N := N + 1

Input

\( u^{(0)} \), \( \epsilon, \zeta \)

\[ u_R^{(n)} = F_R^* u_B^{(n)} + c_R \]

Exit

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 \( \text{EXIT} \)

START
If \( N > ITMAX \), then go to \( \text{EXIT} \)
If \( N < 4 \), then go to \( \text{ONE} \)
If \( |CME-CMOLD| / CME < EPSI \), then go to \( \text{TWO} \)

ONE
(Determine new CME)
Set \( CMOLD := CME \)
If \( N = 0 \), then set \( CME := 0 \)
else set \( CME = \text{square root of the maximum eigenvalue of the tridiagonal matrix} \)
\[
\left\{ \left( \frac{RHO_{i-1}}{GAMMA_i} \right)^{1/2}, \left( (1 - 1/GAMMA_i) \right), \left( \frac{RHO_{i+1}}{GAMMA_i} \right)^{1/2} \right\}
\]

TWO
(Test for stopping)
Compute \( UNRM := u_B^T(\vec{N}) * D * u_B(\vec{N}) \)
\( DELNRM := \delta_B^T(\vec{N}) * D * \delta_B(\vec{N}) \)
If \( UNRM < CNRM \), then set \( UNRM := CNRM \)
Compute \( STEST := (2 \cdot DELNRM/UNRM)^{1/2} / (1 - CME^2) \)
If \( STEST < ZETA \), then go to \( \text{EXIT} \)
Compute:
\[
\begin{align*}
\delta_B & = F_B * F_B \delta_B(\vec{N}) \\
\Gamma_{N+1} & = 1 / (1 - \delta_B^T(\vec{N}) * D_B * \delta_B(\vec{N}) )
\end{align*}
\]
If \( N = 0 \), then \( RHO_{1} := 0 \)
else compute \( RHO_{N+1} := 1 / (1 - \Gamma_{N+1} \cdot DELNRM / (\Gamma_{N+1} \cdot DELSRM \cdot RHO_{N+1} )) \)

Set \( DELSRM := DELNRM, C1 := RHO_{N+1} \cdot \Gamma_{N+1}, C2 := RHO_{N+1}, C3 := 1 - RHO_{N+1}, C4 := RHO_{N+1} (1 - \Gamma_{N+1}) \)
RS-CG (continued)

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 \( N, UNRM^{1/2}, STEST, CME, RHO \), \(\gamma_{N+1}, \Gamma_{N+1} \)

Set \( N = N+1 \)

Go to \( \text{START} \)

**EXIT**

Compute \( u_{R}^{(N)} = F_{B}^{(N)} + C_{R} \)

\[ UNRM := u^{T}(N) D u^{(N)} \]

Print \( N, UNRM^{1/2} \)

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

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

(1) Adaptive Parameters

\[ \text{SPECR} = S(\omega), \quad \gamma = 2/(2-\text{SPECR}), \quad \sigma_E = \text{SPECR}/(2-\text{SPECR}) \]

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

(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

\[ v = \omega u^{(n)} + k^{(F)} \]

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

\[ \delta^{(n)} = q v + k^{(B)} - u^{(n)} \]

(4) Iteration Vector

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

(5) Stopping Test

\[ d^{(n)} = \Delta^{(n)} T \Delta^{(n)} \]

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

If \( \text{STEST} < \xi \), then exit.
(6) Changing Parameter Test

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

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

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

(7) Computing new \( S' \) and \( M_E \)

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

\[ X = Z^{1/(n-s)} \]

\[ \sigma = (X + r/X) / (1+r) \]

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

\[ \tilde{u}(n+1) = u(n) + \delta(n) \]

\[ \Delta(n+1) = Z \tilde{u}(n+1) + k(F) - \tilde{u}(n+1) \]

\[ S_2 = (\Delta(n)^T \ast \Delta(n+1)) / d(n) \]

\[ S' = \max \{SPECR, S_1, S_2\} \]

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

\[ v = B*\delta(n), \quad d(n) = \delta(n)^T \ast D \ast \delta(n), \]

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

\[ M_E = \max \{M_E, M_1, M_2\} \]
SSOR-SI (continued)

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

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

(9) Computation of $\omega$, $\text{SPECR} = S(\omega)$

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

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

$$\text{SPECR} = \frac{2 - 2\omega + \alpha M_E}{(2 - \omega M_E)}$$

else

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

$$\text{SPECR} = \omega - 1$$

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

(10) Computing $\omega^*$

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

(11) Selecting $\omega^*$ Test

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

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

(12) Computation of $\omega$, $\text{SPECR} = S(\omega), M_E$

$$\omega = \omega^*, \quad \text{SPECR} = \omega - 1, \quad M_E = 2\sqrt{\beta}$$
Flow Chart 5a: SSOR-SI Method
Flow Chart 5b: SSOR-SI Method
SSOR-SI: Symmetric Successive Overrelaxation Semi-iterative Algorithm

Input \( u(0), CME, ZETA, F, \text{CASE, ITMAX, LEVEL, EPSI} \)

Set \( N:=0, S:=0, \text{OMEGCHG}:=\text{.TRUE.} \)

Compute \( \text{BETA}:=\max \{ W_{ij} \left[ E_{i-1,j} + N_{i-1,j} \right] + S_{ij} \left[ E_{i,j-1} + N_{i,j-1} \right] \} \)

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

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

\[
\text{SPECR}:=\frac{(2-2 \cdot \text{OMEGA} + \text{CME} \cdot \text{OMEGA})}{(1-\text{CME} \cdot \text{OMEGA})}
\]

else compute

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

\[
\text{SPECR}:=\frac{\text{OMEGA}}{\text{OMEGA} - 1}
\]

\[
\text{CME}:=2 \cdot \text{BETA}^{1/2}
\]

\[
\text{OMEGCHG}:=\text{.FALSE.}
\]

Print \( N, CME, \text{OMEGA, SPECR} \)

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

If \( \text{CNRM} < \text{EPSI} \), then go to \( \text{EXIT} \)

\( \text{START} \) If \( N > \text{ITMAX} \), then go to \( \text{EXIT} \)

Copy \( u(N) \) into \( v \).

Compute \( v = z^*_\omega \ast v + k_\omega(F) \)

\[
\Delta(N) = v - u(N)
\]

\[
v = q^*_\omega \ast v + k_\omega(B)
\]

\[
\delta(N) = v - u(N)
\]

(Test for stopping)

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

If \( \text{UNRM} < \text{CNRM} \), then set \( \text{UNRM}:=\text{CNRM} \)

Compute \( \text{DELRNM}:=\Delta(N)^T D^* \Delta(N) \)

\[
\text{STEST}:=\left[\frac{(2 \cdot \text{OMEGA})}{\text{OMEGA}} \frac{\text{DELRNM} \text{CNRM}}{(1-\text{CME})^{1/2} \text{SPECR}}\right]
\]

If \( \text{STEST} < \text{ZETA} \), then go to \( \text{EXIT} \)

If \( N=0 \), then go to \( \text{CHANGE} \)

If \( \text{OMEGCHG}=.\text{FALSE.} \), then go to \( \text{THREE} \)

IF \( \text{BETA} \geq 1/4 \), then go to \( \text{ONE} \)
SSOR-SI (continued)

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

$\text{TEMP1} := \log(\phi(\text{OMEGAS}-1))$

$\text{TEMP2} := \log(\phi(\text{SPECR}))$

where $\phi(x) = (1-(1-x)^{1/2})/((1+(1-x)^{1/2})$.

If $\text{TEMP1}/\text{TEMP2} < F$, then go to ONE

Set $\text{OMEGA} := \text{OMEGAS}$

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

$\text{OMECHG} := \text{FALSE}$.

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

$S := N$

Print $N, \text{CME}, \text{SPECR}, \text{OMEGA}$

Go to TWO

**ONE** (Test for changing parameters)

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

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

If $\text{QA} \geq \text{QT}^F$, then go to CHANGE

else go to **THREE**

**CHANGE** (Change parameters)

If $N=0$, then set $\text{SIG1} := \text{SPECR}$

Compute $Z := (1+R^P)(\text{QA}+(\text{QA}^2-\text{QT}^2)^{1/2})/2$

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

$\text{SIGEL} := (X+R/X)/(1+R)$

$\text{SIG1} := (\text{SPECR}+\text{SIGEL}(2-\text{SPECR}))/2$

Compute $\text{SIG2} := \Delta(N)^T \ast D \ast \Delta(N+1)/\text{DELNRM}$

Set $\text{SME} = \max\{\text{SIG1}, \text{SIG2}, \text{SPECR}\}$

(Determine new CME, OMEGA, SPECR)

Compute $\text{ZM1} := ((1-\text{SME})(1+\text{BETA} \cdot \text{OMEGA}^2)-\text{OMEGA}(2-\text{OMEGA}))/\text{(OMEGA}(\text{OMEGA}-1-\text{SME})$

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

If CASE = TRUE., then compute $\text{ZM2} := (\delta^{(n)}^T \ast D \ast v)/(\delta^{(n)}^T \ast D \ast \delta^{(n)})$

else compute $\text{ZM2} := [(v^T \ast D \ast v)/(\delta^{(n)}^T \ast D \ast \delta^{(n)})]^{1/2}$
SSOR-SI (continued)

Compute \( CME := \max \{ CME, ZM1, ZM2 \} \)
Set \( S := N \)
If \( CME \leq 4BETA \), then compute
\[
OMEGA := 2 / \left(1 + \sqrt{1 - 2CME + 4BETA} \right)
\]
\[
SPECR := \frac{2 - 2 \cdot OMEGA + CME \cdot OMEGA}{2 - CME + OMEGA}
\]
else compute
\[
OMEGA := 2 / \left(1 + \sqrt{1 - 4BETA} \right)
\]
\[
SPECR := \frac{OMEGA - 1}{2 - CME + OMEGA}
\]
\[
CME := 2 \cdot BETA \sqrt{ } \}
\]
\[
OMEGACHG := .FALSE.
\]
Print \( N, CME, OMEGA, SPECR \)

**TWO**

Compute \( R := (1 - (1 - SPECR) \sqrt{ })/(1 + (1 - SPECR) \sqrt{ }) \)
\[
SIGE := SPECR / (2 - SPECR)
\]
\[
GAMMA := 2 / (2 - SPECR)
\]
\[
RHO := 1
\]

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

Copy \( u(n) \) into \( v \)
Compute \( v = \omega v + k(F) \)
\[
\Delta(n) = v - u(n)
\]
\[
DELSRM := \Delta(n) \cdot D \cdot \Delta(n)
\]
\[
v = q \omega v + k(B)
\]
\[
\delta(n) = v - u(n)
\]
\[
u(n+1) = GAMMA \cdot \delta(n) + u(n)
\]
Go to **ENDIT**

**THREE**

(OMEGA has not been changed)
If \( N = S + 1 \), then compute \( RHO := 1 / (1 - SIGE^2 / 2) \)
else compute \( RHO := 1 / (1 - RHO \cdot SIGE^2 / 4) \)
Set \( C1 := GAMMA \cdot RHO, C2 := RHO, C3 := 1 - RHO \)
Compute \( u(N+1) = C1 \cdot \delta(N) + C2 \cdot u(N) + C3 \cdot u(N-1) \)
SSOR-SI (continued)

**END**

Print $N, UNRM^{\frac{1}{2}}, TSTEST, QA, QT_F, CME, RHO, GAMMA$

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

\[
v = \sum_{i}^{F} u^{(n)} + k^{(F)}
\]

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

\[
\delta^{(n)} = q \cdot v + k^{(B)} - u^{(n)}
\]

(2) Acceleration Parameters

\[
v = \delta^{(n)} - \sum_{i}^{F} \delta^{(n)}
\]

\[
d(n) = \Delta^{(n)} T \Delta^{(n)}
\]

\[
\gamma_{n+1} = d(n)/(\Delta^{(n)} T \Delta^{(n)})
\]

\[
\rho^{n+1} = \begin{cases} 
1 & , \quad n = s+1 \\
1/ \left[ 1 - \left( \frac{\gamma_{n+1}}{\gamma_{n} \rho_{n}} \right) \frac{d(n)}{d(n-1)} \right] & , \quad n > s+1 
\end{cases}
\]

(3) Iteration Vector

\[
u^{(n+1)} = \begin{cases} 
\gamma_{n+1} \delta^{(n)} + u^{(n)} & , \quad n = s+1 \\
\rho_{n+1} \left( \gamma_{n+1} \delta^{(n)} + u^{(n)} \right) + (1 - \rho_{n+1}) u^{(n-1)} & , \quad n > s+1 
\end{cases}
\]
(4) Difference Vector and Residual Vector (recursive computation)

\[
\Delta^{(n+1)} = \begin{cases} 
\Delta^{(n)} - \gamma^{(n+1)}v, & n = s+1 \\
\rho^{(n+1)}(\Delta^{(n)} - \gamma^{(n+1)}v) + (1-\rho^{(n+1)})\Delta^{(n-1)}, & n > s+1 
\end{cases}
\]

\[
v = \delta^{(n)} - v = \omega^{(n)}
\]

\[
v = \delta^{(n)} - v = \omega^{(n)}
\]

\[
\delta^{(n+1)} = \begin{cases} 
\gamma^{(n+1)}v + (1-\gamma^{(n+1)})\delta^{(n)}, & n = s+1 \\
\rho^{(n+1)}(\gamma^{(n+1)}v + (1-\gamma^{(n+1)})\delta^{(n)}) + (1-\rho^{(n+1)})\delta^{(n-1)}, & n > s+1 
\end{cases}
\]

(5) Stopping Test

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

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

If STEST < \(\zeta\), 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 \( n \times n \) tridiagonal matrix \((s+1 \leq i \leq n)\)

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

(7b) Compute new $M_E$

\[
M_1 = \frac{(1-S')(1+S^2 - \omega(2-\omega))}{[\omega(\omega-1-S')]} \quad v = B_5 S(n), \quad d(n) = S(n)^T D \ast S(n)
\]

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

\[
M_E = \max\{M_E, M_1, M_2\}
\]

(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** \( N:=0, S:=0, \text{OMEGCHG:=.TRUE.} \)

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

**If** CME \( \leq 4 \cdot \text{BETA} \), then compute

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

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

else compute

\[
\text{OMEGA}:=2/(1+(1-\text{BETA})^{1/2})
\]

\[
\text{SPECR}:=\text{OMEGA}-1
\]

\[
\text{CME}:=2 \cdot \text{BETA}^{1/2}
\]

\[
\text{OMEGCHG}:=.\text{FALSE.}
\]

Print \( N, \text{CME, OMEGA, SPECR} \)

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

If \( \text{CNRM} < \text{EPSI} \), then go to \( \text{EXIT} \)

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

Compute \( v = \omega * v + k \omega \)

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

\[
v = \omega * v + k \omega
\]

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

**START**

If \( N > \text{ITMAX} \), then go to \( \text{EXIT} \)

(Test for stopping)

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

If \( \text{UNRM} < \text{CNRM} \), then set \( \text{UNRM}:=\text{CNRM} \)

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

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

\[
\left\{ \left[ \frac{\rho_{i+1}-1}{\gamma_{i+1} \rho_{i+1} \gamma_{i-1}} \right]^\frac{1}{2}, \left[ 1 - \frac{1}{\gamma_{i}} \right] \right\}, \text{ for } s+1 \leq i \leq N
\]

If BETA \geq 1/4, then go to ONE
Compute OMEGAS:=2/(1+(1-4 \cdot BETA)^{1/2})
TEMP1:=log(\phi(OMEGAS-1))
TEMP2:=log(\phi(SPECR))
\phi(X) = \frac{1-(1-X)^{1/2}}{1+(1-X)^{1/2}}
If TEMP1/TEMP2 \leq F, then go to ONE
Set OMEGA:=OMEGAS
SPECR:=OMEGAS-1
OMEGCHG:=.FALSE.
CME:=2 \cdot BETA^{1/2}
S:=N
Print N,CME,SPECR,OMEGA
Go to TWO

(Test for changing parameters)
If SPECR > SME, then go to THREE
Compute
\[\lambda_1 = -\log \left( \frac{\phi(SPECR)}{\phi(SME)} \right)\]
\[\lambda_2 = -\log(\phi(SME))\]
If \(\lambda_1/\lambda_2 \geq F\), then go to THREE
(Determine new CME, OMEGA, SPECR)

Compute $ZM1 := \frac{(1-SME)(1+BETA \cdot OMEGA^2) - OMEGA \cdot (2-OMEGA)}{OMEGA(OMEGA-1-SME)}$

Compute $v = B \cdot \delta(n)$

If CASE = .TRUE., then compute $ZM2 = \frac{(\delta(n)^T \cdot D \cdot v)/(\delta(n)^T \cdot D \cdot \delta(n))}{(v^T \cdot D \cdot v)/(\delta(n)^T \cdot D \cdot \delta(n))}^{1/2}$

else compute $ZM2 = \frac{v^T \cdot D \cdot v}{(\delta(n)^T \cdot D \cdot \delta(n))}$

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

Set $S := N$

If CME < 4 * BETA, then compute

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

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

else compute

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

$SPECR := OMEGA-1$

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

$CMEGCHG := .FALSE.$

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 $v = zomega_v + komega$($F$)

$\Delta(n) = v - u(n)$

$v = qomega_v + komega$($B$)

$\delta(n) = v - u(n)$

$DELNRM = \Delta(n)^T \cdot D \cdot \Delta(n)$

(THREE)

Copy $\delta(n)$ into $v$

Compute $v = zomega_v$

$v = \delta \cdot v$

$\gamma_{N+1} = DELNRM/(\delta(n)^T \cdot D \cdot v)$

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

else compute

$\rho_{N+1} = 1/(1-\gamma_{N+1} \cdot DELNRM/(\gamma_N \cdot DELSNM))$
SSOR-CG (continued)

Set DELSNM:=DELNRM,

\[ C_1 := \rho_{N+1} \gamma_{N+1}, \quad C_2 := \rho_{N+1}, \quad C_3 := 1 - \rho_{N+1}, \quad C_4 := \rho_{N+1} (1 - \gamma_{N+1}) \]

Compute \( u^{(n+1)} = C_1 \delta^{(n)} + C_2 u^{(n)} + C_3 u^{(n-1)} \)

\[ \Delta^{(n+1)} = -C_1 \Delta^{(n)} + C_2 \Delta^{(n-1)} \]

\[ v = \delta^{(n)} - \Delta^{(n)} \]

\[ v = \delta^{(n)} - \Delta^{(n)} \]

\[ \delta^{(n+1)} = C_1 \delta^{(n)} + C_4 \delta^{(n)} * C_3 \delta^{(n-1)} \]

Print \( N, UNRM^2, \text{TEST, CME, OMEGA, SPECR, SME,} \rho_{N+1} \gamma_{N+1} \)

Set \( N := N+1 \)

Go to \( \text{START} \)

EXIT

Compute \( UNRM = u^{(n)} \cdot d \cdot u^{(n)} \)

Print \( n, UNRM^2 \)

If \( \text{LEVEL} > 2 \), 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.
PROGRAM CGTST(INP,OUTP)

REAL GRIDX(21),GRIDD(21),COEF(441,4),WKR=(1523),UNKWN(441)

INTEGER GTYPE(21,21),NMXQ(441),IMVNX(441)

DATA IN/6,6/OUTPUT,/*INP*/5,5/INPUT/

COMMON WORKSP

C *** BEGIN: COMMON DECK - ITPACK

COMMON / ITPACK / NGRPTS, NROPTS, NPKPTS, NROPP1,
A CVUF, SMUF, ZETA, EPSI, F, GAMMA, RHO, SIGF,
B HALT, CASE, CHANGE,
C RR, DETNM, DELSNM, UDNM, TEST1, QA, QT,
D IN1, IS1, IV5X,
E ITR, IHR,
F OMEGA, SPECTRA, RFTBAR, OMEGCHG

LOGICAL HALT, CASE, CHANGE, OMEGCHG

C *** END : COMMON DECK - ITPACK

C *** BEGIN: COMMON DECK - FLLPACK

COMMON / RANDY / IPICEF, NROUND, NNDPT

COMMON / CONSTS / IPACK1, IPACK2, IPACKH, INSIDE, HORIZ, VERT, ROTH,
A CORNER, INTER

COMMON / CONTBL / DE-HUG-LEVEL

COMMON / CPDFL / CUXX, CUXY, CUYX, CUY, CUX, CU

COMMON / EQFORM / NUM-F4, NUMCOE

COMMON / EQNDX / NROW, NCOL

COMMON / PRB / DIM2, DIM3, POISON, LAPLAC, CONST, SELFA, CROSST,
A DIRICH, NFUMAN, MIXED, AX, BX, AY, AY, AZ, AY

COMMON / PROP / NGRIDX, NGRIDY, NGRIDZ, UNIFRM, HX, HY, HZ,
C ELL77, RECTAN

INTEGER HORIZ, VERT, ROTH, CORNER,
A PIECE, HPTYPE, RNFIG, BGRD

LOGICAL DIM2, DIM3, POISON, LAPLAC, CONST, SELFA, CROSST, DIRICH,
A NFUMAN, MIXED, UNIFRM, DEBUG, ELLP77, RECTAN,
B PT

REAL AX, AY, AY, AY, AZ, AY, RZ, RZ, RX, HY, HZ, CUXX, CUXY, CUYX, CUX, CUY, CU

C XROUND, YROUND, BPARAM

C *** END : COMMON DECK - FLLPACK

C READ(INP, 70) ICASES

DO 60 IJKLM = 1, ICASES

C

C

************************************************************

C

C

INTERFACE 1: INITIAL SITUATION

C

C
DEFINE PROBLEM ON OUTPUT FILE

WRITE(IPT,90) CMUE, ZETA, FPSI

*****************************************************************************
** INTERFACE 2: DOMAIN PROCESSING ********************************************
*****************************************************************************

CALL REGION(GTYPE,GRIDX,NGRX,GRIDY,NGRY)

*****************************************************************************
** INTERFACE 3: EQUATION GENERATION ********************************************
*****************************************************************************

CALL FIVEPT(GTYPE,GRIDX,NGRX,GRIDY,NGRY,COEF,MXNCOE, MNEQ)

*****************************************************************************
** INTERFACE 4: EQUATION INDEXING **********************************************
*****************************************************************************

CALL RBORD(NDXEQ,MNEQ,INVNDX)

*****************************************************************************
** INTERFACE 5: EQUATION SOLUTION ***********************************************
*****************************************************************************

CALL INTUNK(GTYPE,GRIDX,NGRX,GRIDY,NGRY,MXNFG, A INKWN)

CALL CGC(GTYPE,GRIDX,NGRX,GRIDY,NGRY,COEF,MXNCOE,MXNFG, A NDXFG, INKWN)

*****************************************************************************
** INTERFACE 6: OUTPUT *********************************************************
*****************************************************************************

DO 50 IJ = 1,NGRPTS
   IX = MOD(IJ-1,NGRIDX) + 1
   JY = (IJ-IX)/GRIDX + 1
   GO TO (10,20,30) GTYPE(IX,JY)
50 CONTINUE

WORKSP(IJ) = TRUE(GRIDX(IX),GRIDY(JY))
20 CONTINUE
CALL BCOND(DUMMY,GRIDX(I),GRIDY(J),BVALUS)
WORKSP(TJ) = BVALUS(4)/BVALUS(1)
GO TO 40
30 CONTINUE
WORKSP(TJ) = 0.0
40 CONTINUE
WORKSP(TJ+NGRPTS) = WORKSP(TJ) - UNKNWN(TJ)
50 CONTINUE
TRUNORM = UTNDV(GTYPE,NGRDXD,NGRDXD,COFF,MXNOE,MXNOE,NDXEQ)
A = WORKSP(1+NGRPTS)*WORKSP(1+NGRPTS)
ERRNORM = UTNDV(GTYPE,NGRDXD,NGRDXD,COFF,MXNOE,MXNOE,NDXEQ)
A = WORKSP(1+NGRPTS)*WORKSP(1+NGRPTS)
TRUNORM = SQRT(TRUNORM)
ERRNORM = SQRT(ERRNORM)
WRITE(11,100) TRUNORM, ERRNORM
CALL VOUT(GRIDX,GRIDY,GRIDX,GRIDY,WORKSP(1),NGRPTS)
60 CONTINUE
70 FORMAT(15)
80 FORMAT(1H1,130X,*THE METHOD BEING USED: CJCG/30X, A *THE ORDERING BEING USED: RED-BLACK/30X, B /30X,*CASE II OF THE ADAPTIVE PROCEDE/30X, C *BOUNDARY VALUES ARE SET TO: 0.0/30X, D *INITIAL SOLUTION IS: 0.0/30X, E *TEST PROBLEM NO. 20/30X, F /30X,*STARTING ITERATIVE PARAMETERS ARE: 0/35X, A *F = 0.3*F15.*X/35X,*CMUE = 0, B F15.*X/35X,*ZETA = 0*F15.*X/35X,*EPSI = 0*F15.*X)
100 FORMAT(1H1,130X,*0 TO 1/2 NORM OF TRUE SOLUTION = 0*E15.*X/30X, A 00 TO 1/2 NORM OF THE ERROR = 0*E15.*X)
END

SUBROUTINE PDF(X,Y,CVALUS)
REAL CVALUS(7)
DATA PI/3.14159265358979/
C
C
TWO DIMENSIONS
C
VALUES OF EQUATION COEFFICIENTS AT (X,Y) IN ORDER:
UXX,UXY,UYY,UXY,UYY,RIGHT SIDE
C
EXXY = EX(X,Y)
CVALUS(1) = EXXY
CVALUS(2) = 0.0
CVALUS(3) = 1.0/EXXY
CVALUS(4) = 0.0
CVALUS(5) = 0.0
CVALUS(6) = -1./(1. + X + Y)
TSX = S1*(PI*X)
TSY = S1*(PI*Y)
TCX = COS(PI*X)
TCY = COS(PI*Y)
F2XY = EXXY*EXXY
TEMP = PI*(X*TSX*TCY + 3.*Y*E2XY*TCX*TSY)
TEMP1 = TSX*TSY*((2.*Y*PI*PI*E2XY - PI*PI*E2XY/(1.+Y))
CVALUS(7) = TEMP*TEMP1
C
RETURN
END

SUBROUTINE BCONDTXY,BVALUS)
REAL BVALUS(4)
C
VALUES OF BOUNDARY CONDITIONS COEFFICIENTS AT (X,Y)
IN THE ORDER:
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 PXUNK(X,Y)
C
INITIAL APPROXIMATION TO UNKNOWN VALUES
C
APXUNK = 0.0
C
RETURN
END

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

BEST AVAILABLE COPY
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 $45^\circ$ 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 $45^\circ$ 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 1615 and consists of the number of contours, the number of vertices for a contour followed by 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
0
  10  0  10  10  10  0  10  5  10  5  10  5  10  10  10
0
  10  10  10
2
  10  2  10  2  10  4  10  4  10  4  10  4  10  2  10
```

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

<table>
<thead>
<tr>
<th>Region</th>
<th>Integer Array GTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22222222223333333333</td>
</tr>
<tr>
<td>2</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>3</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>4</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>5</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>6</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>7</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>8</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>9</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>10</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>11</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>12</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>13</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>14</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>15</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>16</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>17</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>18</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>19</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>20</td>
<td>2111111111112333333333</td>
</tr>
<tr>
<td>21</td>
<td>2111111111112333333333</td>
</tr>
</tbody>
</table>
For completeness, we now give the listing of subroutine REGION.
SUBROUTINE REGION (GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD)

FUNCTION: SUPERIMPOSES A MESH OF SIZE $H_x=H_y=H_1/H_2$ ON A REGION DEFINED BY CLOSED CONTOURS AND CONSTRUCTS AN INTEGER ARRAY WHICH DESCRIBES EACH MESH POINT ON THE SMALLEST RECTANGLE CIRCUMSCRIBING THE POSSIBLY IRREGULAR REGION AS AN INTERIOR POINT (+1), AN BOUNDARY POINT (+2), OR AN EXTERIOR POINT (+3).

USAGE: CALL REGION (GTYPE,GRIDX,NGRDXD,GRIDY,NGRDYD).

PARAMETERS:

GTYPE - GTYPE IS AN NGRDXD BY NGRDYD INTEGER ARRAY USED TO INDICATE THE TYPE OF POINT ON THE GRID. THE NUMBERS 1, 2, OR 3 INDICATE RESPECTIVELY INTERIOR, BOUNDARY, OR EXTERIOR POINTS OF THE GRID.

NGRDXD - NGRDXD IS THE ROW DIMENSION OF THE ARRAY GTYPE AS SPECIFIED IN THE CALLING PROGRAM.

NGRDYD - NGRDYD IS THE COLUMN DIMENSION OF THE ARRAY GTYPE SPECIFIED IN THE CALLING PROGRAM.

GRIDX - GRIDX IS AN ARRAY OF LENGTH NGRDXD DIMENSIONED IN THE CALLING PROGRAM. UPON LEAVING REGION IT CONTAINS THE $X$ COORDINATES OF THE MESH LINES STARTING IN THE LOWER LEFT HAND CORNER.

GRIDY - GRIDY IS AN ARRAY OF LENGTH NGRDYD DIMENSIONED IN THE CALLING PROGRAM. UPON LEAVING REGION IT CONTAINS THE $Y$ COORDINATES OF THE MESH LINES STARTING IN THE LOWER LEFT HAND CORNER.

OTHER PARAMETERS PASSED IN LABELED COMMON ARE:

LEVEL = 0 NO PRINTING FROM REGION
       = 1 THE INPUT DATA ONLY IS PRINTED.
       = 2 THE GRAPH OF THE REGION ONLY IS PRINTED.
       = 3 PRINT BOTH INPUT DATA AND GRAPH OF REGION

DEBUG IS A LOGICAL DEBUGGING PARAMETER. IF TRUE THEN LEVEL IS RESET TO 3. IF FALSE NO ACTION IS TAKEN.
NGRIDX IS THE NUMBER OF MESH POINTS IN THE X-DIRECTION OF THE CIRCUMSCRIBED RECTANGLE. THIS IS COMPUTED IN REGION.

NGRIDY IS THE NUMBER OF MESH POINTS IN THE Y-DIRECTION OF THE CIRCUMSCRIBED RECTANGLE. THIS IS COMPUTED IN REGION.

NGRPTS IS THE NUMBER OF TOTAL MESH POINTS OF THE CIRCUMSCRIBED RECTANGLE. THIS IS COMPUTED IN REGION.

HX*HY ARE THE MESH SIZE FOR THE GRID. REGION READS H1 AND H2 FROM DATA AND COMPUTES HX=HY=H1/H2. HX AND HY ARE THEN RETURNED TO THE CALLING PROGRAM.

AX,BX ARE THE MINIMUM AND MAXIMUM VALUES OF THE X COORDINATE. REGION COMPUTES THESE AND RETURNS THEM IN LABELED COMMON.

AY,BY ARE THE MINIMUM AND MAXIMUM VALUES OF THE Y COORDINATE. REGION COMPUTES THESE AND RETURNS THEM IN LABELED COMMON.

PARAMETERS USING BLANK COMMON ARE:

L IS DESCRIBED BELOW

OTHER PARAMETERS:

H1,H2 ARE PARAMETERS INDICATING THE UNIFORM MESH SIZE IN RATIONAL FORM. THESE AS WELL AS L ARE READ IN AS DATA AND ARE DESCRIBED FURTHER BELOW.

THE DEFINITION OF THE REGION IS AS FOLLOWS:

KN = NUMBER OF CONTOURS IN THE REGION.
L(5,K) = NUMBER OF VERTICES ON THE K-TH CONTOUR -- 3 OR MORE
L(1,I)/L(2,I) = THE X COORDINATE OF THE I-TH VERTEX.
L(3,I)/L(4,I) = THE Y COORDINATE OF THE I-TH VERTEX.
H1/H2 = THE MESH SIZE TO BE CONSIDERED.

NOTES: (1) THE ARRAY L USES BLANK COMMON FURNISHED IN THE CALLING PROGRAM.
(2) THE INPUT DATA FOR THE VERTICES MUST BE ORDERED SO THE INTERIOR OF THE REGION ALWAYS LIES TO THE LEFT.
(3) REQUIRED SUBROUTINES -- RTREG, IABS, MODS

WRITTEN OR MODIFIED BY DATE

ROGER G. GRIMES AND DAVID R. KINCAID JUNE 1977
ROGER G. GRIMES AND DAVID R. KINCAID FEBRUARY 1977
JAMES O. SULLIVAN AND DAVID R. KINCAID FEBRUARY 1976
ALKIS J. MOURADOGLOU AND JOHN H. DAUWALDER APRIL 1967

CENTER FOR NUMERICAL ANALYSIS/COMPUTATION CENTER
UNIVERSITY OF TEXAS AT AUSTIN
REFERENCES:

(1) KINCAID, DAVID R. AND ROGER G. GRIMES, **** CNA REPORT ***


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)

BEGIN: COMMON DECK - ITPACK
END: COMMON DECK - ITPACK

BEGIN: COMMON DECK - ELLPACK
END: COMMON DECK - ELLPACK

NUMBER(1)=2
NUMBER(2)=1
NUMBER(3)=3
LETTER(1)=1H
LETTER(2)=1H
LETTER(3)=1H
MP1=NGRDXD-1
MQ1=NGRKYD-1
READ (IROR,610) KN
READ (IRDY,610) KCO1
L(5,1)=KCO1
READ (IRDY,610) (L(1,J),L(2,J),L(3,J),L(4,J),J=1,KCO1)
IF (KN.LE.1) GO TO 20
DO 10 K=2,KN
   READ (IRDY,610) L(5,K)
   KCO2=KCO1+1
   KCO1=KCO1*L(5,K)
10 CONTINUE

IT IS ASSUMED THAT L(5,K) > 0 FOR K=1,2,...,KN AND THAT KN > 0.

10 READ (IRDY,610) (L(1,J),L(2,J),L(3,J),L(4,J),J=KCO2,KCO1)

THE READING OF THE COORDINATES OF THE VERTICES IS WITH 1615 FORMAT

L(1*I),L(2*I),L(3*I),L(4*I),L(1*I+1),L(2*I+1),L(3*I+1),...

20 READ (IRDY,610) H1,H2
   KJ=0
   IF (DEBUG) LEVEL=3
   IF (MOD(LEVEL,2).EQ.0) GO TO 40

PRINT INPUT DATA
WRITE (IPTR,620)
KJ2=0
DO 30 K=I,KN
   KJ1=KJ2+1
   KJ2=KJ2+L(5,K)
   WRITE (IPTR,630) K
   WRITE (IPTR,640) K
   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=I,KN
   KJ1=KJ2+2
   KJ2=L(5,I)
   IF (KJ1.GT.KJ2) STOP 1
   DO 50 KS=KJ1,KJ2
      KS1=KS-1
      IF (L(1,KS).*L(2,KS).GT.L(1,KS1).*L(2,KS1)) W
         WRITE (IPTR,670) KS,1
      CONTINUE
      KS2=KS+1
      IF (L(1,KS).LT.L(1,KS2)) W
         WRITE (IPTR,670) KS,2
      CONTINUE
50 CONTINUE
   DO 60 J=I,KN
      IF (L(1,KJ1).EQ.L(1,KJ2).OR.L(3,KJ1).EQ.L(3,KJ2).
         L(1,KJ1).LT.L(1,KJ2).OR.L(3,KJ1).GT.L(3,KJ2)) W
         WRITE (IPTR,670) KJ1,J
      CONTINUE
   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   MNY1/MNY2 = 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   IF (KCO1.LE.1) STOP 4
DO 100 I=2*KC01
IF (MAXX1*L(2,1) .GE. MAXX2*L(1,1)) GO TO 70
MAXX1=L(1,1)
MAXX2=L(2,1)
GO TO 80
70 IF (MINX1*L(2,1) .LE. MINX2*L(1,1)) GO TO 80
MINX1=L(1,1)
MINX2=L(2,1)
80 IF (MAXY1*L(4,1) .GE. MAXY2*L(3,1)) GO TO 90
MAXY1=L(3,1)
MAXY2=L(4,1)
GO TO 100
90 IF (MINY1*L(4,1) .LE. MINY2*L(3,1)) GO TO 100
MINY1=L(3,1)
MINY2=L(4,1)
100 CONTINUE
IF (H2*(MAXX1*MINX2-MAXX2*MINX1) .LE. MP1*H1*MINX2*MAXX2) GO TO 110
WRITE (I,680)
STOP 5
110 IF (H2*(MAXY1*MINY2-MAXY2*MINY1) .LE. MQ1*H1*MINY2*MAXY2) GO TO 120
WRITE (I,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 IS THE TOTAL NUMBER OF VERTICES.
C
MM= -2*(L(1,1)*MINX2-L(2,1)*MINX1)
NN= -1*MINX2*L(2,1)
KK=MM/NN
IF (KK .NE. QQ.MM) GO TO 140
130 WRITE (I,700)
STOP 7
140 L(1,1)=KK
MM= -2*(L(3,1)*MINY2-L(4,1)*MINY1)
NN= -1*MINY2*L(4,1)
KK=MM/NN
IF (KK .NE. QQ.MM) GO TO 130
L(2,1)=KK
150 CONTINUE
C
C*** DETERMINE NGRIDX AND NGRIDY.

NGRXM1=L(1,1)
NGRYM1=L(2,1)
IF (KCO1 .LT. 2) STOP 10
DO 170 I=2,KCO1
   IF (NGRXM1.GT.L(1,1)) GO TO 160
   NGRXM1=L(1,1)
160 IF (NGRYM1.GT.L(2,1)) GO TO 170
   NGRYM1=L(2,1)
170 CONTINUE
NGRIDX=NGRXM1+1
NGRXP1=NGRIDX+1
NGRIDY=NGRYM1+1
NGRYP1=NGRIDY+1

BEST AVAILABLE COPY
C SET THE ARRAY ETYPE TO ZERO.

DO 180 J=1,NGRIDY
180 GTYPE(I,J)=0

C DEFINE THE BOUNDARY POINTS WHICH ARE NOT VERTICES IN THE ARRAY GTYPE

KJ2=0
DO 480 K=1,KN
  KJ1=KJ2+1
  KJ2=KJ2*L(S,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,IPN,IQN,S)
   IF (GTYPE(IP,IQ).EQ.0) GO TO 230
   WRITE (IPTP,710) J,K
   STOP 11

230 MODS=MOD(R+4,8)
240 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
250 IF (R.NE.S) GO TO 260
   WRITE (IPTP,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(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=ABS(IPP-IP)-1
IQM1=ABS(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,IQP+N)=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 390 N=1,IQM1
390 GTYPE(IP+IQP+N)=SET
GO TO 470
390 DO 400 N=1,IPM1
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+IQP-N)=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=IPM1
IQ=IQM1
P=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
C000 CHECK CONSISTENCY OF CONTOUR ORIENTATIONS SCANNING RIGHT TO LEFT.
C
FLAG=0
SFT=3
DO 540 K=1,NGRIDX
I=NGRP1-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 520
FLAG=I
GO TO 540
530 IF (GTYPE(I,J) .EQ. SET) GO TO 540
WRITE (IPTR,730) I,J
STOP 13
540 CONTINUE

C
C000 SET THE VALUES OF THE ARRAY GTYPE.
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) HI,H2
ME<AB=NGRIDX
DO 560 K=1,NGRIDY
J=NGRP1-K
560 WRITE (IPTR,750) J,GTYPE(I,J) .EQ. 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
MX=FLOAT(H1)/FLOAT(H2)
HY=HX
AX=FLOAT(MINX1)/FLOAT(MINX2)
8X=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 (1HI//)
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/))
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
3 AN 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 */)
730 FORMAT (1H2,///,10X,56HINCONSISTENT CONTOUR ORIENTATION FOR MESH P
1INT 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//)
750 FORMAT (3X,15,2X,101A1)
C END

SUBROUTINE RTREG (P1,Q1,P2,Q2,RS)
INTEGER P1,P2,Q1,Q2,RS
C
IF (P1-P2) 10,50,90
C
P1 < P2.
C
10 IF (Q1-Q2) 20,30,40
20 RS=5
RETURN
30 RS=4
RETURN
40 RS=3
RETURN
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:
The variables which are initialized from the input file are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITMAX</td>
<td>F</td>
</tr>
<tr>
<td>ZETA</td>
<td>CME</td>
</tr>
<tr>
<td>EPSI</td>
<td>SME</td>
</tr>
<tr>
<td>ADAPT</td>
<td>OMEGA</td>
</tr>
<tr>
<td>CASEI</td>
<td>SPECR</td>
</tr>
</tbody>
</table>

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 110, 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.

<table>
<thead>
<tr>
<th>Columns</th>
<th>I</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>.000001</td>
<td>.000001</td>
<td>TRUE</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>.75</td>
<td>.95</td>
<td>1.64</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ITPACK would then set:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITMAX</td>
<td>100</td>
</tr>
<tr>
<td>ZETA</td>
<td>.000001</td>
</tr>
<tr>
<td>EPSI</td>
<td>.000001</td>
</tr>
<tr>
<td>ADAPT</td>
<td>.TRUE.</td>
</tr>
<tr>
<td>CASEI</td>
<td>.FALSE.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>.75</td>
</tr>
<tr>
<td>CME</td>
<td>.95</td>
</tr>
<tr>
<td>OMEGA</td>
<td>1.64</td>
</tr>
<tr>
<td>SPECR</td>
<td>0.0</td>
</tr>
</tbody>
</table>
**REPORT DOCUMENTATION PAGE**

<table>
<thead>
<tr>
<th>1. REPORT NUMBER</th>
<th>2. GOVT ACCESSION NO.</th>
<th>3. RECIPIENT'S CATALOG NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITR-CNA-126</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE (and Subtitle)</th>
<th>5. TYPE OF REPORT &amp; PERIOD COVERED</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITPACK Report: Numerical Studies of Several Adaptive Iterative Algorithms</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6. PERFORMING ORG. REPORT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAHC04 74 G 01993</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7. AUTHOR(s)</th>
<th>8. CONTRACT OR GRANT NUMBER(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>David R. Kincaid and Roger G. Grimes</td>
<td>DAHC04 74 G 01993</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. PERFORMING ORGANIZATION NAME AND ADDRESS</th>
<th>10. PROGRAM ELEMENT, PROJECT, TASK AREA &amp; WORK UNIT NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center for Numerical Analysis</td>
<td>P-12301-M</td>
</tr>
<tr>
<td>The University of Texas at Austin</td>
<td></td>
</tr>
<tr>
<td>Austin, Texas 78712</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11. CONTROLLING OFFICE NAME AND ADDRESS</th>
<th>12. REPORT DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The University of Texas at Austin</td>
<td>August 1977</td>
</tr>
<tr>
<td>Austin, Texas 78712</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13. NUMBER OF PAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>14. MONITORING AGENCY NAME &amp; ADDRESS(If different from Controlling Office)</th>
<th>15. SECURITY CLASS. (of this report)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>unclassified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16. DISTRIBUTION STATEMENT (of this Report)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved for public release; distribution unlimited.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, If different from Report)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>18. SUPPLEMENTARY NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>19. KEY WORDS (Continue on reverse side if necessary and identify by block number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITPACK, ELLPACK, elliptic partial differential equation, iterative algorithm, adaptive parameter determination</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20. ABSTRACT (Continue on reverse side if necessary and identify by block number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>