1641

179
3/25/80

DR. 947

MARCH 1980

PPPL-1641

UC-20g

MASTER

A PROGRAM GENERATOR FOR THE
INCOMPLETE CHOLESKY CONJUGATE
GRADIENT (ICCG) METHOD WITH A
SYMMETRIZING PREPROCESSOR

BY

G. KUO-PETRAVIC AND M. PETRAVIC

PLASMA PHYSICS LABORATORY

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

PRINCETON UNIVERSITY
PRINCETON, NEW JERSEY

This work was supported by the U. S. Department of Energy
Contract No. EY-76-C-02-3073. Reproduction, translation,
publication, use and disposal, in whole or in part, by or
for the United States Government is permitted.

G. Kuo-Petravic and M. Petravic

Plasma Physics Laboratory, Princeton University
Princeton, New Jersey 08544 USA

DISCLAIMERS

[illegible]

1962-1963

ABSTRACT

This paper is an extension of our previous paper "A Program Generator for the Incomplete LU-Decomposition-Conjugate Gradient (ILUCG) Method" which appeared in Computer Physics Communications [1]. In that paper we presented a generator program which produced a code package to solve the system of equations $Ax = b$, where A is an arbitrary nonsingular matrix, by the ILUCG method. In the present paper we offer an alternative generator program which produces a code package applicable to the case where A is symmetric and positive definite. The numerical algorithm used is the Incomplete Cholesky Conjugate Gradient (ICCG) method of Meijerink and Van der Vorst [2] which executes approximately twice as fast per iteration as the ILUCG method. In addition, we provide an optional preprocessor to treat the case of a not diagonally dominant nonsymmetric and nonsingular matrix A by solving the equation $A^T A x = A^T b$.

PROGRAM SUMMARY

Title of Program: GENIC

Catalogue number: ABSE

Program obtainable from: CPC Program Library, Queen's University
of Belfast, N. Ireland, U.K.

Computer: PDP-10/CYBER 172

Installation Computer Center, Plasma Physics Laboratory,
Princeton University. James Forrestal
Campus, P. O. Box 451, Princeton, NJ 08544 USA

Operating system: TOPS-10, version 6.03/NOS 1.3

Programming Language used: FORTRAN

High-speed storage required: None

Number of bits in word: 36 bits/60 bits

Overlay structure: None

Number of magnetic tapes required: None

Other peripherals used: None

Number of cards in combined program and test deck:
approximately 3,000 cards

Card-punching code: EBCDIC

Keywords: Linear sparse system, Incomplete Cholesky,
Conjugate Gradient, Code Generator

Nature of Physical Problem

The program generator GENIC and the resultant Incomplete Cholesky Conjugate Gradient (ICCG) solver package SOLIC are applicable to a wide range of physical problems which require the solution of a large sparse system of linear equations:

$$Ax = b$$

where A is a symmetric and positive definite square matrix of rank N . It is also desirable that A be reasonably diagonally dominant; that is, $A_{ii} \geq \sum_{i \neq j}^N |A_{ij}|$, for otherwise the incomplete decomposition used here may be a poor approximation of the complete decomposition, and in the worst case the decomposition itself may become unstable. Since this is an iterative method, it is particularly suitable for large systems. The package is most efficient when the nonzero elements of A appear clustered in diagonal bands. Such patterns are often found in implicit finite difference formulations of partial differential equations resulting from modeling of multidimensional physical problems.

Method of Solution

For symmetric, positive definite matrices, the Incomplete Cholesky Conjugate Gradient (ICCG) algorithm of Meijerink and Van der Vorst [2] is used. For nonsymmetric matrices which are not diagonally dominant, a symmetrizing preprocessor is generated which transforms $Ax = b$ into $A^T Ax = A^T b$, where A^T is the transpose of A ; after which the standard ICCG algorithm applies.

Restrictions on the complexity of the problem

The generator assumes that the nonzero elements of the sparse matrix occur in diagonal bands, as is often the case with finite difference equations. If this pattern does not pertain, then some unnecessary work corresponding to computation on zeros will be produced.

Typical running time

Under 5 seconds.

References:

- [1] G. Kuo-Petravic and M. Petravic, "A Program Generator for the Incomplete LU Decomposition Conjugate Gradient (ILUCG) Method," Comput. Phys. Commun. 1979 (in press).
- [2] J. A. Meijerink and H. A. Van der Vorst, "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix," Math. of Comput., 31, 137 (1977).

LONG WRITE UP

1. Introduction

The Incomplete Cholesky Conjugate Gradient (ICCG) method is a much improved version of the conjugate gradient method developed by Hestennes and Stiefel [3], the improvement resulting from replacing the iterations with the original problem matrix A in $Ax = b$ by iterations with an approximate inverse of A . In the ICCG method, the approximate inverse is obtained by incomplete Cholesky UTU decomposition where U is an upper triangular matrix and U^T its transpose as proposed by Meijerink and Van der Vorst [2]. The usual Cholesky version of Gaussian elimination is used except that a preselected sparsity pattern, usually that of the original matrix A , is forced upon U .

The advantages of the code generator approach over fixed program package in the implementation of a particular numerical algorithm are two fold: That of increased program efficiency through the generation of highly optimized code, coupled with the ease of producing a code, which results in the ability to try different variations of the algorithm on a particular user problem. Naturally, efficiency is seldom bought without the loss of generality and in our case we have made use of the fact that finite difference formulation of partial differential equations, in general, result in sparse matrices where the nonzero elements cluster in diagonal bands. The flexibility of the code generator approach has enabled us to build the following features into the GENIC package:

- (i) The pattern of the Incomplete Cholesky decomposition

may be varied by specifying that the sparsity pattern of the decomposed matrix be different from that of the original matrix, subject to the condition that the decomposed matrix has also a banded structure. Thus, ICCG (m), where $m = 0, 1, 2$ denoting the number of extra diagonals, may all be tried with great ease.

(ii) If A is not positive definite or diagonally dominant, problems of stability and accuracy may arise during the decomposition phase. This can be alleviated by either adopting the ILUCG algorithm of [1], or, the use of a symmetrizing preprocessor which produces

$$\underline{A}^T \underline{A} \underline{x} = \underline{A}^T \underline{b} \quad \text{from} \quad \underline{A} \underline{x} = \underline{b} ,$$

where A^T is the transpose of A. In this case, of course, A can be asymmetric. The conjugate gradient algorithm is then applied to the symmetric matrix $A^T A$. The chief drawback for this method, apart from the increased storage requirements, is that the condition number is now $\lambda_{\max}^2 / \lambda_{\min}^2$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the original maximum and minimum eigenvalues of A respectively, hence more iterations may be required for convergence. However, each iteration is faster in execution time than that of the ILUCG algorithm. So precisely which method is more suitable for a particular problem has to be found by trial and error.

2. Numerical algorithm

The basic ICCG algorithm of Meijerink and Van der Vorst [2] has been implemented. First, the symmetric and positive definite matrix A in the equation

$$\underline{A} \underline{x} = \underline{b} \tag{1}$$

where $\underline{x}$ is the solution vector, is decomposed by symmetric incomplete Cholesky decomposition into:

$$A \approx U^T U$$

where U is an upper triangular matrix and U^T its transpose. The recursive procedure for solving the system then involves the use of two subsidiary vectors $\underline{r}_i$ and $\underline{p}_i$. We start with

$$\underline{r}_0 = \underline{b} - A\underline{x}_0 \quad (3.1)$$

where $\underline{x}_0$ is an arbitrary initial trial vector and

$$\underline{p}_0 = (U^T U)^{-1} \underline{r}_0 \quad (3.2)$$

The iteration consists of:

$$\alpha_i = (\underline{r}_i, (U^T U)^{-1} \underline{r}_i) / (\underline{p}_i, A \underline{p}_i) \quad (3.3)$$

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \underline{p}_i \quad (3.4)$$

$$\underline{r}_{i+1} = \underline{r}_i - \alpha_i A \underline{p}_i \quad (3.5)$$

$$\underline{s}_{i+1} = (U^T U)^{-1} \underline{r}_{i+1} \quad (3.6)$$

$$\beta_i = (\underline{r}_{i+1}, \underline{s}_{i+1}) / (\underline{r}_i, (U^T U)^{-1} \underline{r}_i) \quad (3.7)$$

$$\underline{p}_{i+1} = \underline{s}_{i+1} + \beta_i \underline{p}_i \quad (3.8)$$

until a predetermined convergence criterion such as:

$$\|x_i - x_{i-1}\|_2 / \|x_i\|_2 \leq \epsilon$$

where ϵ is a user-specified parameter, has been satisfied.

3. The generator program GENIC

The structure of the generator program GENIC is analogous to that of GENLU as described in [1]. The banded structure of the matrix A lends itself naturally to packing into a two-dimensional array ADU(N, KP), where N is the rank of A and KP the total number of diagonals in the upper triangular portion of A. The main diagonal is stored in ADU(n, 1), $n = 1, 2, \dots, N$. Any other diagonal is stored analogously in ADU(n, kp), $n = 1, 2, \dots, N - k + 1$, where k is the ordinal number of the diagonal, counting the main diagonal as having $k = 1$; that is, the immediate superdiagonal has $k = 2$, that above it, has $k = 3$, and so on. Note that $k \geq kp$ as kp is the packed value of k because it ignores the presence of diagonals containing only zeros.

The program functions start with the reading of input data provided by the user which define the structure and symmetry of A. From this information, the generator then generates a suite of subroutines which are to be attached to the user problem program. It is quite possible to provide GENIC with the ordinal numbers of all the nonzero diagonals but since these diagonals cluster in bands it is sufficient to provide only the starting and finishing ordinal numbers of the diagonal bands.

Data for the GENIC program is input through two namelists:

1. Namelist/DIMS/NOUTPP, N, MANEXP
2. Namelist/ABANDS/IDFAUT, AUS, AUF, US, UF, USATA, UFATA, ALS, ALF

which the program reads from the file GENIN.

All of the output of GENIC is placed into the file GENOUT.

The meaning of variables read via DIMS is:

- a. NOUTPP = output unit number for SOLIC to output diagnostic printout.
- b. N = rank of matrix A.
- c. MANEXP = number of binary digits in the mantissa of the floating point number representation of the computer on which SOLIC is to be run. This parameter is used in the optimization of the treatment for bad pivots [4]. The default of MANEXP = 48 has been set for CDC 60-bit word computers.

The meaning of variables read via namelist ABANDS is:

- a. IDFAUT, a switch which chooses the basic options provided by GENIC:

IDFAUT = 1, means A is symmetric and the decomposed matrix which is to be stored in the array UD has the same sparsity pattern as A, which is stored in the array ADU. Therefore, the user need only input parameters for ADU, i.e., AUS, AUF.

IDFAUT = 2, means A is symmetric but the decomposed matrix which is to be stored in the array UD has more nonzero diagonals than the matrix A, i.e., ICCG(m) algorithm, where m is an integer greater than zero, is generated. Input parameters for ADU, UD only, i.e., AUS, AUF, US, UF.

IDFAUT = 3, means transform $Ax = b$ into $A^T Ax = A^T b$ where A^T is the transpose of A. This transformation is only of value where A is asymmetric, $A^T A$ on the other hand is symmetric and the ICCG algorithm can be used. Input parameters for ADU, ADL, i.e., AUS, AUF, ALS, ALF only. GENIC will automatically output AUSATA, AUFATA giving the starting and finishing ordinal numbers of diagonals in the matrix $A^T A$.

IDFAUT = 4, means the same as IDFAUT = 3 except that the decomposed matrix which is to be stored in the array UATA, is to have more diagonals than the matrix $A^T A$ which is to be stored in the array ATA. Input all parameters listed: AUS, AUF, ALS, ALF, USATA, UFATA. In order to find out which are the nonzero ordinal bands of the product matrix $A^T A$ the user should first run GENIC with IDFAUT = 3 as AUSATA and AUFATA

will be printed out. After knowing AUSATA and AUFATA the user can then experiment with USATA and UFATA.

b. AUS(I)

AUS(1) is the ordinal number of the first nonzero diagonal of the first band of the upper triangular part of matrix A. AUS(2) is the ordinal number of the first nonzero diagonal of the second band in the upper triangular part of matrix A, and so on.

c. AUF(I)

AUF(1) is the ordinal number of the last nonzero diagonal of the first band in the upper triangular part of matrix A. AUF(2) is the ordinal number of the last nonzero diagonal of the second band in the upper triangular part of matrix A, and so on.

d. US(I)

The same as AUS(I) except that it applies to the decomposed matrix which is stored in UD.

e. UF(I)

The same as AUF(I) except that it applies to the decomposed matrix which is stored in UD.

f. ALS(I)

This is only necessary when the preprocessor is to be used. ALS has the same meaning as AUS(I) except that it applies to the lower triangular part of matrix A which is initially stored in the array ATA.

g. ALF(I)

This is only necessary when the preprocessor is to be used. ALF has the same meaning as AUF(I) except that it applies to the lower triangular part of matrix A which is initially stored in the array ATA.

h. USATA(I)

Same connotation as US(I) except that it applies to the decomposed form of $A^T A$ which is to be read in the array UATA. USATA(I) has to be read in only if one wants to have different band widths for the decomposed matrix as compared with $A^T A$. IDFAUT = 3 should first be used and the band widths of $A^T A$ will be output by GENIC in the form of comments on AUSATA(I) and AUFATA(I). Once the user has gained experience on the SOLIC package thus produced, further optimization may be obtained by experimenting with USATA(I) and UFATA(I) using IDFAUT = 4.

i. UFATA(I)

Same connotation as UF except that it applies to the decomposed $A^T A$.

4. SOLIC, the solver package produced by GENIC

(a) A is symmetric and IDFAUT = 1 or 2.

The SOLIC package consists of the following five subroutines: SOLIC, SETU, UDCOMP, ADOTV, and UTUINV. Subroutine SOLIC contains the main conjugate gradient algorithm and also provides the interface with the user program. All other subroutines are called from SOLIC.

Having received the problem matrix A in the compacted form ADU, the right hand side b and the trial vector x_0 for $Ax = b$, SOLIC first copies the contents of ADU into UD by calling SETU. It then calls UDCOMP which performs the incomplete Cholesky decomposition on UD after which the algorithm (3) is applied. Several specialized tasks within the algorithm are performed by specific subroutines: Operations by A on any vector y are performed by calling ADOTV. This is used in sections (3.3) and (3.5) of the algorithm. Similarly, the triangular system $(UTU)^{-1}r_i$ is solved by the subroutine UTUINV.

(b) A is asymmetric and IDFAUT = 3 or 4

The SOLIC package consists of the following subroutines: SOLIC, MULTDI, STUATA, UATACP, ATADV, ATDOTV, and UATATU. Subroutine MULTDI will perform the operation of multiplying diagonals to form the matrix A^TA which will be found in the array ATA(N, M) where M is the number of diagonals in the resulting matrix A^TA . Subroutines STUATA will copy the contents of ATA into the array UATA. SOLIC then calls subroutine UATACP to perform incomplete Cholesky decomposition. Subroutine ATDOTV will form the new vector A^Tb from b. Subroutine ATADV will perform the operation

of $A^T A \cdot y$ where y is any vector. Lastly, subroutine UATATU will perform the operation $(U^T U)^{-1} y$.

5. User Interface with the SOLIC package

(a) A is symmetric and IDFAUT = 1 or 2.

This interface is entirely handled by the three parameters of the SUBROUTINE SOLIC (ISTOP, CONVEG, XNORMS) and one labeled common block.

COMMON/IC/ADU(D1,D2),XICCG(D1),RHS(D1)

The meaning of the three calling parameters of subroutine SOLIC are:

1. Integer ISTOP = the maximum number of iterations to be performed before passing control from SOLIC to the calling program.
2. Real CONVEG = specification of the convergence criterion. SOLIC returns control to the calling subprogram when $\|x_i - x_{i-1}\|_2 / \|x_i\|_2$ becomes less or equal to CONVEG.

3. Real array XNORMS (ISTOP) = the value of $\|x_i - x_{i-1}\|_2$ at each iteration step from $i = 1$ to $i = \text{ISTOP}$. The values are returned to the calling program so that the rate of convergence may be monitored by the user every time SOLIC is called.

The labeled common block COMMON/IC/ should be included in the calling program and all elements of the three arrays in it are to be set. RHS must equal the right hand side vector b of (1) and XICCG the trial vector x_0 of (3:1). ADU must contain the problem matrix A of (1) in a compacted form described earlier. It is desirable that A be normalized so that all main diagonal elements are equal to 1, if at all possible. The symbolic dimension D1 has the meaning of row and D2 the diagonal of the A matrix. D1 and D2 will appear as absolute constants in SOLIC. The user must set $\text{ADU}(I, KP) = A(I, KP)$ where ADU contains the upper triangular half of A ; here I is the row index of A and KP is the packed diagonal index K of A . KP is determined from K simply by counting only the nonzero subdiagonals sequentially starting from the main diagonal which is counted as 1. Note that the matrix $A(I, KP)$ does not actually exist in the SOLIC program; it is used here only to explain the contents of ADU.

- (b) A is asymmetric and IDFAUT = 3 or 4

Again subroutine SOLIC has the same three parameters, but two labeled common blocks are now required in the calling program:

COMMON/IC/XICCG(D1), RHS(D1)

COMMON/CA/ATA(D1, D2), UATA(D1, D2)

The value of D2 will depend on the number of diagonals in

the product symmetric matrix $A^T A$ and this number can be obtained once SOLIC has been generated. In fact one can simply copy the common blocks from SOLIC. The upper and lower portions of the matrix A are to be stored in the arrays UATA and ATA respectively, again the main diagonal should be stored in $UATA(n, 1)$ $n = 1, 2, \dots, N$ and the nonzero diagonal immediately above the main diagonal in $UATA(n, 2)$, and so on.

It is clear from the foregoing paragraph that each time the user wishes to change the number of grid points a new SOLIC package has to be generated. However, the generation process is quite transparent and not time consuming. We feel that the disadvantages of this extra generation are more than compensated by the gain in execution speed of the SOLIC package in its existing form where the indices of arrays appear as absolute constants rather than combinations of variables.

6. Output of the solver program SOLIC:

SOLIC outputs only a small amount of diagnostic messages on an output channel defined by NOUTPP, a parameter read in through the namelist DIMS. The diagnostic output contains the following information:

1. ITSTOP = the actual number of iterations performed in the present call to SOLIC.
2. XNORM = $\| \underline{x}_i \|_2$
3. DXNORM = $\| \underline{x}_i - \underline{x}_{i-1} \|_2$
4. RATIO = $\| \underline{x}_i - \underline{x}_{i-1} \|_2 / \| \underline{x}_i \|_2$
5. SIRI = $(\underline{s}_i, \underline{r}_{i+1})$. This is a measure of orthogonality of $\underline{s}_i$ and $\underline{r}_{i+1}$. Though neither vector is normalized, this is still a good indication of the proper functioning of the code. SIRI should be normally around 10^{-10} after reasonable convergence has been achieved.

The solution of the system (1) is returned in XICCG and the vector $\underline{b}$ is overwritten by the residue $\underline{r}_i$.

7. Test Case Input and Output

We show in Table 1(a) a test program which contains a SOLIC package generated from GENIC using the following input parameters:

```
%DIMS
%NOUTPP=6, N=25, %ANF%XP=48 *
%$PARAMS
IDFAINT=1,AUS(1)=1,AUF(1)=2,AUS(2)=5,AUF(2)=7 *
```

A is symmetric and the decomposed matrix is assumed to have the same sparsity pattern as A. In the test program, the RHS vector $\underline{b}$ in the equation $A\underline{x} = \underline{b}$ was set assuming the solution

vector x to be of the form $x(n) = n$, $n = 1, 2, \dots, N$. It may be verified from the output shown in Table 1(b) that this is indeed the case.

In Table 2(a) we show a test program using the SOLIC program which includes the symmetrizing preprocessor so that the equation solved was $A^T A x = A^T b$. The input to GENIC was:

```
$DIMS
NOUTPP=6, N=24, MANEXP=49
$PARAMS
IDFAUT=3,AUS(1)=1,AUF(1)=2,AUS(2)=7,AUF(2)=7,
ALS(1)=1,ALF(1)=2,ALS(2)=7,ALF(2)=7
```

The matrix A was set to have $a_{ij} = 0$ for $i \neq j$ and $a_{ij} = 1$ for $|i - j| = 1$, apart from small nonzero terms at $|i - j| = 6$. It may be seen from the output of this SOLIC package, in Table 2(b), that the preprocessor was able to handle this case well, resulting in a matrix $A^T A$ which is reasonably diagonally dominant. This same matrix was also used to test the ILUCG algorithm of Ref. [1]. It was found that while $a_{ij} = 0$, the ILUCG algorithm was unable to handle this case, but with $a_{ij} \approx 0.01$ the ILUCG method was already able to reach convergence as fast as the $A^T A$ method.

A precise timing test performed on a SOLIC package with a 600×600 matrix, against a SOLLU package [1] using the same parameters showed that SOLIC runs in approximately 58% of the time (per iteration) of the SOLLU package.

This work was supported by United States Department of Energy Contract No. EY-76-C-02-3073.

- [1] G. Kuo Petravic and M. Petravic, Computer Phys. Commun., 1979, in press.

- [2] J. A. Meijerink and H. A. Van Der Vorst, "An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-Matrix," Math. of Comp. 31, 137 (1977), 148-162.

- [3] M. R. Hestenes and E. Stiefel, J. Res. Of National Bureau of Standards, 49 (1952) 409.

- [4] D. S. Kershaw, "On the Problem of Unstable Pivots in the Incomplete LU-Conjugate Gradient Method," Proceedings of 8th Conference on Numerical Simulation of Plasma. Monterey, California, June 1978.

```

PROGRAM TSYN(TSYMD,TAPE6=TSYMD)
COMMON/IC/ADU(25,5),XICCG(25),RHS(25)
COMMON/UD/UD(25,5)
DIMENSION KDIAG(5)
DIMENSION XNORMS(600)
N=25
DO 1 M=1,N
  ADU(M,2)=-0.25
  ADU(M,4)=-0.25
  ADU(M,3)=-0.0001
  ADU(M,5)=-0.0001
1 CONTINUE

C
C
DO 3 M=1,N
3  ADU(M,1)=1.0
  DO 5 M=1,N
    RHS(M)=0.
5  XICCG(M)=1.0
    KDIAG(1)=1
    KDIAG(2)=2
    KDIAG(3)=5
    KDIAG(4)=6
    KDIAG(5)=7
    DO 7 M=1,N
      RM=M
      RHS(M)=RHS(M)+ADU(M,1)*RM
      DO 8 MM=2,5
        RM=M+KDIAG(MM)-1
        IF (RM.GT.N) GO TO 18
        RHS(M)=RHS(M)+ADU(M,MM)*RM
        RMR=M
        K=KDIAG(MM)
        RHS(M+K-1)=RHS(M+K-1)+ADU(M,MM)*RMR
      GO TO 8
18  CONTINUE
      ADU(M,MM)=1.0E+10
8  CONTINUE

C
7  CONTINUE
  ISTOP=100
  CONVEG=1.0E-10
  CALL APRINT(N,ADU,XICCG,RHS)

  CALL SOLIC (ISTOP,CONVEG,XNORMS)
  CALL DPRINT(N,UD,XICCG,RHS)
  STOP
END
SUBROUTINE APRINT(N,ADU,XICCG,RHS)
  DIMENSION ADU(N,5),
  * XICCG(N),RHS(N)
  DIMENSION PARRAY(12)
  WRITE(6,100)
100 FORMAT(' ADU(I,1) ADU(I,2) ADU(I,3) ADU(I,4) ADU(I,5)',
  * XICCG(I) RHS(I)')
  DO 10 I=1,N
    PARRAY(1)=ADU(I,1)
    PARRAY(2)=ADU(I,2)
    PARRAY(3)=ADU(I,3)
    PARRAY(4)=ADU(I,4)
    PARRAY(5)=ADU(I,5)
    PARRAY(6)=XICCG(I)
    PARRAY(7)=RHS(I)
    WRITE(6,101)(PARRAY(IP),IP=1,7)
  10 CONTINUE

```

```
101 FORMAT(7E10.3)
10 CONTINUE
RETURN
END
```

C

```
SUBROUTINE DPRINT(N,UD,XICCG,RHS)
DIMENSION UD(N,5),XICCG(N),RHS(N)
DIMENSION PARRAY(12)
WRITE(6,100)
100 FORMAT(" UD(I,1) UD(I,2) UD(I,3) UD(I,4) UD(I,5)",
" XICCG(I) RHS(I)" )
DO 10 I=1,N
PARRAY(1)=UD(I,1)
PARRAY(2)=UD(I,2)
PARRAY(3)=UD(I,3)
PARRAY(4)=UD(I,4)
PARRAY(5)=UD(I,5)
PARRAY(6)=XICCG(I)
PARRAY(7)=RHS(I)
WRITE(6,101)PARRAY(IP),IP=1,7)
101 FORMAT(7E10.3)
10 CONTINUE
RETURN
END
```

C THIS PROGRAM PACKAGE SOLIC USES THE FOLLOWING
C PARAMETERS WHICH HAVE BEEN INPUT:

C NOUT= 3 NOUTPP= 6 N= 25 MANEXP= 48

C IDFAUT= 1

AUS(I)=										
1	5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

AUF(I)=										
2	7	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

US(I)=										
1	5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

UF(I)=										
2	7	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

```
SUBROUTINE SOLIC(ISTOP,CONVEG,XNORMS)
DIMENSION XNORMS(ISTOP)
COMMON/IC/ADU( 25, 5),XICCG( 25),RHS( 25)
COMMON/UU/UD( 25, 5)
DIMENSION XIM1( 25)
```

C THIS SUBROUTINE IS CALLED BY THE PROBLEM PROGRAM. IT

```

C CONTAINS THE MAIN CONJUGATE GRADIENT LOOP
  DIMENSION RI( 25),VTEMP( 25),SI( 25),PI( 25)
  EQUIVALENCE (RI(1),RHS(1))
  DO 1 M=1, 25
    1 XIM1(M)=0.0
    CALL SETU
    ITSTEP=1
    CALL UDCOMP
C FORM RESIDUAL VECTOR AND STORE IN RI
    CALL ADDTV(VTEMP,XICCG)
    DO 10 M=1, 25
      10 RI(M)=RHS(M)-VTEMP(M)
C
C
C FORM GRADIENT VECTOR
C SI=(UT.U)**-1.RI
    CALL UTUINV4SI,RI,VTEMP,
    DO 11 M=1, 25
      11 PI(M)=SI(M)
C
C
C CALCULATE ALFAI
    ANUM=0
    DO 12 M=1, 25
      12 ANUM=ANUM+K1(M)*PI(M)
    100 CONTINUE
C VTEMP=A.PI
    CALL ADDTV(VTEMP,PI)
    ADENOM=0
    DO 13 M=1, 25
      13 ADENOM=ADENOM+PI(M)*VTEMP(M)
    ALFAI=ANUM/ADENOM
C
C
    DO 14 M=1, 25
      14 XICCG(M)=XICCG(M)+ALFAI*PI(M)
C FIND NEW RESIDUE
    DO 15 M=1, 25
      15 RI(M)=RI(M)-ALFAI*VTEMP(M)
C
C
C CHECK FOR CONVERGENCE
    DXNORM=0.0
    XNORM=0.0
    DO 16 M=1, 25
      DXNORM=DXNORM+(XIM1(M)-XICCG(M))**2
      XNORM=XNORM+XICCG(M)*(XICCG(M)
      XIM1(M)=XICCG(M)
    16 CONTINUE
    XNORM=SQRT(XNORM)
    DXNORM=SQRT(DXNORM)
    RATIO=DXNORM/XNORM
    XNORMS(ITSTEP)=DXNORM
C*****
C THIS SECTION DEMARCATED BY *****SERVE FOR DIAGNOSTIC
C PURPOSES ONLY AND CAN BE DELETED FOR PRODUCTION RUNS
    SIRI=0.0
    DO 17 M=1, 25
      17 SIRI=SIRI+RI(M)*SI(M)
    WRITE( 6,101) ITSTEP,XNORM,DXNORM,RATIO,SIRI
C*****
    IF (RATIO.LE.CONVEG) GO TO 1000
    IF (ITSTEP.EQ.ISTOP) GO TO 1000
    ITSTEP=ITSTEP+1

```



```

C
C
C   CALCULATE SI
      CALL UTUINV(SI,RI,VTEMP)
      BNUM=0
      DO 18 M=1, 25
18   BNUM=BNUM+RI(M)*SI(M)
      BETA1=BNUM/ANUM
      ANUM=BNUM

C
C
C   UPDATE PI
      DO 19 M=1, 25
19   PI(M)=SI(M)+BETA1*PI(M)
      GO TO 100
101  FORMAT(1H ,8H ITSTEP=,I3,8H XNORM=,E8.2,
+9H DXNORM=,E8.2,3H RATIO=,E8.2,7H SIRI=,E9.2)
1000 RETURN
      END
C-----
C-----
      SUBROUTINE SETU
      COMMON/IC/ADU( 25, 5),XICCG( 25),RHS( 25)
      COMMON/UU/UD( 25, 5)
C   THIS SUBROUTINE SETS UD TO THE VALUES GIVEN
C   IN ADU AND ADL BY THE USER
      DO 10 J=1, 5
      DO 10 M=1, 25
      UD(M,J)=0.0
10   CONTINUE

C
C
      DO 20 M=1, 25
      UD(M,1)=ADU(M,1)
      UD(M,2)=ADU(M,2)
20   CONTINUE

C
C
      DO 30 J= 3, 5
      DO 30 M=1, 25
      UD(M,J+ 0)=ADU(M,J)
30   CONTINUE
      RETURN
      END
C-----
C-----
      SUBROUTINE UDCOMP
      COMMON/IC/ADU( 25, 5),XICCG( 25),RHS( 25)
      COMMON/UU/UD( 25, 5)
      DIMENSION KDIAG( 50)
      RNDOFF=SQRT(EXP(- 48*ALOG(2.0)))
      DO 100 J=1, 19
C   TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
      UDMAX=0
      DO 501 KP=2, 5
      UDABS=ABS(UD(J,KP))
      IF(UDABS.GT.UDMAX) UDMAX=UDABS
501  CONTINUE
      ALIMIT=RNDOFF*UDMAX
      IF(UD(J,1).LT.ALIMIT)
      UD(J,1)= ALIMIT
      UD(J,1)=SORT(UD(J,1))
      DO 1 KP=2, 5
1   UD(J,KP)=UD(J,KP)/UD(J,1)

```

```

R=UD(J, 2)
UD(J+ 1, 1)=UD(J+ 1, 1)-R*UD(J, 2)
UD(J+ 1, 3)=UD(J+ 1, 3)-R*UD(J, 4)
UD(J+ 1, 4)=UD(J+ 1, 4)-R*UD(J, 5)
R=UD(J, 3)
UD(J+ 4, 1)=UD(J+ 4, 1)-R*UD(J, 3)
UD(J+ 4, 2)=UD(J+ 4, 2)-R*UD(J, 4)
R=UD(J, 4)
UD(J+ 5, 1)=UD(J+ 5, 1)-R*UD(J, 4)
UD(J+ 5, 2)=UD(J+ 5, 2)-R*UD(J, 5)
R=UD(J, 5)
UD(J+ 6, 1)=UD(J+ 6, 1)-R*UD(J, 5)
100 CONTINUE
J= 20
C TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
UDMAX=0
DO 502 KP=2, 4
UDABS=ABS(UD(J,KP))
IF(UDABS.GT.UDMAX) UDMAX=UDABS
502 CONTINUE
ALIMIT=RNDOFF*UDMAX
IF(UD(J,1).LT.ALIMIT)
UD(J,1)= ALIMIT
UD(J,1)=SQRT(UD(J,1))
DO 2 KP=2, 4
2 UD(J,KP)=UD(J,KP)/UD(J,1)
R=UD(J, 2)
UD(J+ 1, 1)=UD(J+ 1, 1)-R*UD(J, 2)
UD(J+ 1, 3)=UD(J+ 1, 3)-R*UD(J, 4)
R=UD(J, 3)
UD(J+ 4, 1)=UD(J+ 4, 1)-R*UD(J, 3)
UD(J+ 4, 2)=UD(J+ 4, 2)-R*UD(J, 4)
R=UD(J, 4)
UD(J+ 5, 1)=UD(J+ 5, 1)-R*UD(J, 4)
J= 21
C TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
UDMAX=0
DO 503 KP=2, 3
UDABS=ABS(UD(J,KP))
IF(UDABS.GT.UDMAX) UDMAX=UDABS
503 CONTINUE
ALIMIT=RNDOFF*UDMAX
IF(UD(J,1).LT.ALIMIT)
UD(J,1)= ALIMIT
UD(J,1)=SQRT(UD(J,1))
DO 3 KP=2, 3
3 UD(J,KP)=UD(J,KP)/UD(J,1)
R=UD(J, 2)
UD(J+ 1, 1)=UD(J+ 1, 1)-R*UD(J, 2)
R=UD(J, 3)
UD(J+ 4, 1)=UD(J+ 4, 1)-R*UD(J, 3)
DO 101 J= 22, 24
C TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
UDMAX=0
DO 504 KP=2, 2
UDABS=ABS(UD(J,KP))
IF(UDABS.GT.UDMAX) UDMAX=UDABS
504 CONTINUE
ALIMIT=RNDOFF*UDMAX
IF(UD(J,1).LT.ALIMIT)
UD(J,1)= ALIMIT
UD(J,1)=SQRT(UD(J,1))
DO 4 KP=2, 2
4 UD(J,KP)=UD(J,KP)/UD(J,1)

```

```

      R=UD(J, 2)
      UD(J+ 1, 1)=UD(J+ 1, 1)-R*UD(J, 2)
101 CONTINUE
      J= 25
      ALIMIT=RNDOFF
      IF (UD(J,1).LT.ALIMIT)
      . UD(J,1)=ALIMIT
      UD(J,1)=SQRT(UD(J,1))
      RETURN
      END
C-----
C-----
      SUBROUTINE ADDTV(RESULT,COLVEC)
      COMMON/IC/ADU( 25, 5),XICCG( 25),RHS( 25)
      COMMON/UU/UD( 25, 5)
      DIMENSION RESULT(1),COLVEC(1)
C THIS SUBROUTINE PERFORMS A DOT PRODUCT OF MATRIX A
C WITH A COLUMN VECTOR
C COMPUTE RESULT=A.COLVEC
      DO 9 M=1, 25
9 RESULT(M)=ADU(M,1)*COLVEC(M)
C
C
      K= 2
      IULIM= 26-K
      KP=K
      DO 10 M=1,IULIM
      RESULT(M)=RESULT(M)+ADU(M,KP)*COLVEC(K-1+M)
10 CONTINUE
C
C
      DO 20 K= 5, 7
      IULIM= 26-K
      KP=KP+1
      DO 20 M=1-IULIM
      RESULT(M)=RESULT(M)+ADU(M,KP)*COLVEC(K-1+M)
20 CONTINUE
C
C
      K= 2
      ILLIM=K
      KP=K
      DO 30 M=ILLIM, 25
      RESULT(M)=RESULT(M)+ADU(M-K+1,KP)*COLVEC(M-K+1)
30 CONTINUE
C
C
      DO 40 K= 5, 7
      ILLIM=K
      KP=KP+1
      DO 40 M=ILLIM, 25
      RESULT(M)=RESULT(M)+ADU(M-K+1,KP)*COLVEC(M-K+1)
40 CONTINUE
      RETURN
      END
C-----
C-----
      SUBROUTINE UTUINV(RESULT,COLVEC,AUX)
      COMMON/IC/ADU( 25, 5),XICCG( 25),RHS( 25)
      COMMON/UU/UD( 25, 5)
      DIMENSION RESULT(1),COLVEC(1),AUX(1)
C THIS SUBROUTINE PERFORMS THE OPERATION:
C (UT*TRANSPOSE)*U**=-1* A COLUMN VECTOR
C COMPUTE RESULT=(UT.U)**-1*COLVEC

```

```

C FIRST AUX=UT**~1*COLVEC
C THEN RESULT=U**~1*AUX
C SECTION A
  AUX( 1)=COLVEC( 1)
  .
  . /UD( 1,1)
C SECTION B
  DO 10 M= 2, 4
10  AUX(M)=(COLVEC(M)
  .      -UD(M- 1, 2)*AUX(M- 1)
  .      )/UD(M,1)
C SECTION C: BAND NUMBER IBAND= 2
  AUX( 5)=(COLVEC( 5)
  .      -UD( 4, 2)*AUX( 4)
  .      -UD( 1, 3)*AUX( 1)
  .      )/UD( 5,1)
  AUX( 6)=(COLVEC( 6)
  .      -UD( 5, 2)*AUX( 5)
  .      -UD( 2, 3)*AUX( 2)
  .      -UD( 1, 4)*AUX( 1)
  .      )/UD( 6,1)
C SECTION D: BAND NUMBER IBAND= 2
  DO 20 M= 7, 25
20  AUX(M)=(COLVEC(M)
  .      -UD(M- 1, 2)*AUX(M- 1)
  .      -UD(M- 4, 3)*AUX(M- 4)
  .      -UD(M- 5, 4)*AUX(M- 5)
  .      -UD(M- 6, 5)*AUX(M- 6)
  .      )/UD(M,1)
C
C
C NOW GENERATE BACKWARD SWEEP
C SECTION E
  RESULT( 25)=AUX( 25)/UD( 25,1)
C SECTION F
  DO 40 M= 2, 4
  MR= 26-M
40  RESULT(MR)=(AUX(MR)
  .      -UD(MR, 2)*RESULT(MR+ 1)
  .      )/UD(MR,1)
C SECTION G: BAND NUMBER IBAND= 2
  RESULT( 21)=(AUX( 21)
  .      -UD( 21, 2)*RESULT( 22)
  .      -UD( 21, 3)*RESULT( 25)
  .      )/UD( 21,1)
  RESULT( 20)=(AUX( 20)
  .      -UD( 20, 2)*RESULT( 21)
  .      -UD( 20, 3)*RESULT( 24)
  .      -UD( 20, 4)*RESULT( 25)
  .      )/UD( 20,1)
C SECTION H: BAND NUMBER IBAND= 2
  DO 50 M= 7, 25
  MR= 26-M
50  RESULT(MR)=(AUX(MR)
  .      -UD(MR, 2)*RESULT(MR+ 1)
  .      -UD(MR, 3)*RESULT(MR+ 4)
  .      -UD(MR, 4)*RESULT(MR+ 5)
  .      -UD(MR, 5)*RESULT(MR+ 6)
  .      )/UD(MR,1)
  RETURN
  END

```

ADU(I,1)	ADU(I,2)	ADU(I,3)	ADU(I,4)	ADU(I,5)	XICCG(I)	RHS(I)
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.100E+01
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.751E+00
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.502E+00
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.252E+00
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.210E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.240E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.280E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.320E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.360E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.400E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.440E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.480E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.520E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.560E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.600E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.640E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.680E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.720E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	-.760E-02
.100E+01	-.250E+00	-.100E-03	-.250E+00	.100E+11	.100E+01	-.540E-02
.100E+01	-.250E+00	-.100E-03	.100E+11	.100E+11	.100E+01	.649E+01
.100E+01	-.250E+00	.100E+11	.100E+11	.100E+11	.100E+01	.675E+01
.100E+01	-.250E+00	.100E+11	.100E+11	.100E+11	.100E+01	.700E+01
.100E+01	-.250E+00	.100E+11	.100E+11	.100E+11	.100E+01	.725E+01
.100E+01	.100E+11	.100E+11	.100E+11	.100E+11	.100E+01	.140E+02
ITSTEP= 1	XNORM= .62E+02	DXNORM= .62E+02	RATIO= .10E+01	SIRI= .66E-11		
ITSTEP= 2	XNORM= .74E+02	DXNORM= .17E+02	RATIO= .23E+00	SIRI= .27E-12		
ITSTEP= 3	XNORM= .74E+02	DXNORM= .12E+01	RATIO= .17E-01	SIRI= .27E-14		
ITSTEP= 4	XNORM= .74E+02	DXNORM= .40E-01	RATIO= .54E-03	SIRI= .11E-16		
ITSTEP= 5	XNORM= .74E+02	DXNORM= .20E-02	RATIO= .27E-04	SIRI= .20E-19		
ITSTEP= 6	XNORM= .74E+02	DXNORM= .85E-04	RATIO= .11E-05	SIRI= .22E-22		
ITSTEP= 7	XNORM= .74E+02	DXNORM= .23E-05	RATIO= .31E-07	SIRI= .21E-25		
ITSTEP= 8	XNORM= .74E+02	DXNORM= .33E-07	RATIO= .44E-09	SIRI= .13E-28		
ITSTEP= 9	XNORM= .74E+02	DXNORM= .60E-09	RATIO= .81E-11	SIRI= .45E-35		
UD(I,1)	UD(I,2)	UD(I,3)	UD(I,4)	UD(I,5)	XICCG(I)	RHS(I)
.100E+01	-.250E+00	-.100E-03	-.250E+00	-.100E-03	.100E+01	.102E-11
.968E+00	-.258E+00	-.647E-01	-.258E+00	-.103E-03	.200E+01	.676E-12
.966E+00	-.259E+00	-.691E-01	-.259E+00	-.104E-03	.300E+01	-.460E-11
.966E+00	-.259E+00	-.694E-01	-.259E+00	-.104E-03	.400E+01	-.603E-11
.966E+00	-.259E+00	-.695E-01	-.259E+00	-.104E-03	.500E+01	.285E-11
.931E+00	-.287E+00	-.721E-01	-.269E+00	-.107E-03	.600E+01	.382E-11
.920E+00	-.291E+00	-.838E-01	-.272E+00	-.109E-03	.700E+01	.169E-11
.918E+00	-.292E+00	-.863E-01	-.272E+00	-.109E-03	.800E+01	-.152E-11
.918E+00	-.292E+00	-.866E-01	-.272E+00	-.109E-03	.900E+01	-.515E-11
.918E+00	-.293E+00	-.867E-01	-.272E+00	-.109E-03	.100E+02	.323E-11
.914E+00	-.299E+00	-.876E-01	-.274E+00	-.109E-03	.110E+02	.449E-11
.911E+00	-.300E+00	-.898E-01	-.275E+00	-.110E-03	.120E+02	.470E-12
.910E+00	-.301E+00	-.907E-01	-.275E+00	-.110E-03	.130E+02	.393E-12
.910E+00	-.301E+00	-.909E-01	-.275E+00	-.110E-03	.140E+02	-.397E-12
.910E+00	-.301E+00	-.909E-01	-.275E+00	-.110E-03	.150E+02	-.645E-11
.909E+00	-.302E+00	-.912E-01	-.275E+00	-.110E-03	.160E+02	.243E-11
.908E+00	-.303E+00	-.916E-01	-.275E+00	-.110E-03	.170E+02	.406E-11
.908E+00	-.303E+00	-.919E-01	-.275E+00	-.110E-03	.180E+02	-.729E-11
.908E+00	-.303E+00	-.919E-01	-.275E+00	-.110E-03	.190E+02	.507E-11
.908E+00	-.303E+00	-.920E-01	-.275E+00	.100E+11	.200E+02	.242E-11
.908E+00	-.303E+00	-.920E-01	.100E+11	.100E+11	.210E+02	-.173E-11
.908E+00	-.303E+00	.100E+11	.100E+11	.100E+11	.220E+02	.135E-13
.908E+00	-.303E+00	.100E+11	.100E+11	.100E+11	.230E+02	-.140E-11
.908E+00	-.303E+00	.100E+11	.100E+11	.100E+11	.240E+02	-.319E-11
.908E+00	.100E+11	.100E+11	.100E+11	.100E+11	.250E+02	.305E-11

```

PROGRAM TATA(TATA0,TAPE6=TATA0)
COMMON/IC/ XICCG(36),RHS(36)
COMMON/CA/ATA(36,7),UATA(36,7)
DIMENSION KDIAG(5)
DIMENSION XNORMS(200)
N=36
DO 1 M=1,N
  UATA(M,2)=1.0
  UATA(M,3)=-0.1
1 CONTINUE
C
DO 2 M=1,N
  ATA(M,2)=1.0
  ATA(M,3)=-0.1
2 CONTINUE
C
DO 3 M=1,N
  UATA(M,7)=0.0
3 UATA(M,7)=0.0
DO 5 M=1,N
  RHS(M)=0.
  XICCG(M)=1.0
  KDIAG(1)=1
  KDIAG(2)=2
  KDIAG(3)=7
DO 7 M=1,N
  RM=M
  RHS(M)=RHS(M)+UATA(M,1)*RM
DO 8 MM=2,3
  RP=M+KDIAG(MM)-1
  IF (RM.GT.N) GO TO 19
  RHS(M)=RHS(M)+UATA(M,MM)*RM
GO TO 8
19 CONTINUE
  UATA(M,MM)=1.0E+10
8 CONTINUE
C
DO 9 MM=2,3
  RP=M-KDIAG(MM)+1
  IF (RM.LT.1) GO TO 19
  RHS(M)=RHS(M)+ATA(M,MM)*RM
GO TO 9
19 CONTINUE
  ATA(M,MM)=1.0E+10
9 CONTINUE
6 CONTINUE
7 CONTINUE
ISTOP=100
CONVEG=1.0E-10
CALL APRINT(N,ATA,UATA,XICCG,RHS)
TIME=SECOND(TT)
WRITE(6,100) TIME
100 FORMAT(" TIME=",F20.5)

CALL SOLIC (ISTOP,CONVEG,XNORMS)
TIME=SECOND(TT)
WRITE(6,100) TIME
CALL ATAPRT(N,ATA,XICCG)

```

```

STOP
END
SUBROUTINE APRINT(N,ADL,ADU,XICCG,RHS)
  DIMENSION ADU(N,7),ADL(N,7),
    . XICCG(N),RHS(N)
  DIMENSION PARRAY(12)
  WRITE(6,100)
100  FORMAT(" ADL(I,3) ADL(I,2) ADL(I,1) ADU(I,1) ADU(I,2)",
    . " ADU(I,3) XICCG(I) RHS(I)" )
  DO 10 I=1,N
    PARRAY(1)=ADL(I,3)
    PARRAY(2)=ADL(I,2)
    PARRAY(3)=ADL(I,1)
    PARRAY(4)=ADU(I,1)
    PARRAY(5)=ADU(I,2)
    PARRAY(6)=ADU(I,3)
    PARRAY(7)=XICCG(I)
    PARRAY(8)=RHS(I)
    WRITE(6,101)(PARRAY(IP),IP=1,8)
101  FORMAT(PE10.3)
  10  CONTINUE
  RETURN
END

```

C

```

SUBROUTINE ATAPRT(N,A,XICCG)
  DIMENSION A(N,7),XICCG(N)
  DIMENSION PARRAY(13)
  WRITE(6,100)
100  FORMAT(" ATA(I,1) ATA(I,2) ATA(I,3) ATA(I,4) ATA(I,5)",
    . " ATA(I,6) ATA(I,7) XICCG(I)" )
  DO 10 I=1,N
    PARRAY(1)=A(I,1)
    PARRAY(2)=A(I,2)
    PARRAY(3)=A(I,3)
    PARRAY(4)=A(I,4)
    PARRAY(5)=A(I,5)
    PARRAY(6)=A(I,6)
    PARRAY(7)=A(I,7)
    PARRAY(8)=XICCG(I)
    WRITE(6,101)(PARRAY(IP),IP=1,8)
  10  CONTINUE
101  FORMAT(PE10.3)
  RETURN
END
SUBROUTINE CPRINT(N,UD,LD,XICCG,RHS)
  DIMENSION UD(N,5),LD(N,5),XICCG(N),RHS(N)
  REAL LD
  DIMENSION PARRAY(12)
  WRITE(6,100)
100  FORMAT(" LD(N,5) LD(N,4) LD(N,3) LD(N,2) LD(N,1)",
    . " UD(N,1) UD(N,2) UD(N,3) UD(N,4) UD(N,5) XICCG(N)",
    . "RHS(N)" )
  DO 10 I=1,N
    PARRAY(1)=LD(I,5)
    PARRAY(2)=LD(I,4)
    PARRAY(3)=LD(I,3)
    PARRAY(4)=LD(I,2)
    PARRAY(5)=LD(I,1)

```

```

PARRAY(6)=UD(I,1)
PARRAY(7)=UD(I,2)
PARRAY(8)=UD(I,3)
PARRAY(9)=UD(I,4)
PARRAY(10)=UD(I,5)
PARRAY(11)=XICCG(I)
PARRAY(12)=RHS(I)
WRITE(6,101)(PARRAY(IP),IP=1,12)
101 FORMAT(12F10.3)
10 CONTINUE
RETURN
END
C THIS PROGRAM PACKAGE SOLIC USES THE FOLLOWING
C PARAMETERS WHICH HAVE BEEN INPUT:
C NCUT= 3 NCUTPP= 6 N= 36 MANEXP= 48
C IDFAUT= 3
C AUS(I)=
C 1 7 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C AUF(I)=
C 2 7 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C ALS(I)=
C 1 7 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C ALF(I)=
C 2 7 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C 0 0 0 0 0 0 0 0 0 0
C
SUBROUTINE MULTDI
COMMON/CA/ATA( 36, 7),UATA( 36, 7)
DO 10 ND=1, 3
DO 10 M=1, 36
10 UATA(M, 8-ND)=ATA(M,ND)
DO 20 M=1, 36
20 UATA(M, 7)=UATA(M,1)
DO 30 ND=1, 7
DO 30 M=1, 36
30 ATA(M,ND)=0.
DO 40 M=1, 36
40 ATA(M+ 0, 1)=ATA(M+ 0, 1)
. +UATA(M+ 0, 1)*UATA(M+ 0, 1)
DO 50 M=1, 35

```



```
C      AUSATA(I)=  
C    1     6   13     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
  
C      AUFA TA(I)=  
C    3     8   13     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
C    0     0     0     0     0     0     0     0     0  
  
C      USATA(I)=
```

```

C      1      6      13      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      UFATA(I)=
C      3      6      13      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0
C      0      0      0      0      0      0      0      0      0      0

```

```

C-----
C-----

```

```

      SUBROUTINE SOLIC(ISTOP,CONVEG,XNORMS)
      DIMENSION XNORMS(ISTOP)
      COMMON/IC/XICCG( 36),RHS( 36)
      COMMON/CA/ATA( 36, 7),UATA( 36, 7)
      DIMENSION XIM1( 36)
C THIS SUBROUTINE IS CALLED BY THE PROBLEM PROGRAM. IT
C CONTAINS THE MAIN CONJUGATE GRADIENT LOOP
      DIMENSION RI( 36),VTEMP( 36),SI( 36),PI( 36)
      EQUIVALENCE (RI(1),RHS(1))
      CALL ATDOTV(VTEMP,RHS)
      DO 1 M=1, 36
        RHS(M)=VTEMP(M)
      1 XIM1(M)=0.0
      CALL MULTDI
C ATA NOW CONTAINS THE MATRIX AT.A
      CALL STUATA
      ITSTEP=1
      CALL UATACP
C FORM RESIDUAL VECTOR AND STORE IN RI
      CALL ATADV(VTEMP,XICCG)
      DO 10 M=1, 36
      10 RI(M)=RHS(M)-VTEMP(M)
C
C
C FORM GRADIENT VECTOR
C SI=(UATA(I),UATA)*-1,RI
      CALL UATATU(SI,RI,VTEMP)
      DO 11 M=1, 36
      11 PI(M)=SI(M)
C
C
C CALCULATE ALFAI
      ANUM=0
      DO 12 M=1, 36
      12 ANUM=ANUM+RI(M)*PI(M)
      100 CONTINUE
C VTEMP=A.PI
      CALL ATADV(VTEMP,PI)
      ADENOM=0
      DO 13 M=1, 36
      13 ADENOM=ADENOM+PI(M)*VTEMP(M)
      ALFAI=ANUM/ADENOM
C

```

```

C
  DO 14 M=1, 36
14  XICCG(M)=XICCG(M)+ALFAT*PI(M)
C  FIND NEW RESIDUE
  DO 15 M=1, 36
15  RI(M)=RI(M)-ALFAT*VTEMP(M)
C
C
C  CHECK FOR CONVERGENCE
  DXNORM=0.0
  XNORM=0.0
  DO 16 M=1, 36
  DXNORM=DXNORM+(XIMI(M)-XICCG(M))**2
  XNORM=XNORM+XICCG(M)*XICCG(M)
  XIMI(M)=XICCG(M)
16  CONTINUE
  XNORM=SQRT(XNORM)
  DXNORM=SQRT(DXNORM)
  RATIO=DXNORM/XNORM
  XNORMS(ITSTEP)=DXNORM
C*****
C  THIS SECTION DEMARCATED BY *****SERVE FOR DIAGNOSTIC
C  PURPOSES ONLY AND CAN BE DELETED FOR PRODUCTION RUNS
  SIRI=0.0
  DO 17 M=1, 36
17  SIRI=SIRI+RI(M)*SI(M)
  WRITE( 6,101) ITSTEP,XNORM,DXNORM,RATIO,SIRI
C*****
  IF (RATIO.LE.CONVEG) GO TO 1000
  IF (ITSTEP.EQ.ISTOP) GO TO 1000
  ITSTEP=ITSTEP+1
C
C
C  CALCULATE SI
  CALL UATATU(SI,RI,VTEMP)
  BNUM=0
  DO 18 M=1, 36
18  BNUM=BNUM+RI(M)*SI(M)
  BETAI=BNUM/ANUM
  ANUM=BNUM
C
C
C  UPDATE PI
  DO 19 M=1, 36
19  PI(M)=SI(M)+BETAI*PI(M)
  GO TO 100
101  FORMAT(1H ,8H ITSTEP=,I3,8H XNORM=,E8.2,
.9H DXNORM=,E8.2,8H RATIO=,E8.2,7H SIRI=,F9.2)
1000 RETURN
  END
C-----
C-----
C  SUBROUTINE STUATA
  COMMON/IC/XICCG( 36),RHS( 36)
  COMMON/CA/ATA( 36, 7),UATA( 36, 7)
C  THIS SUBROUTINE SETS UATA TO THE VALUES GIVEN
C  BY ATA OBTAINED BY MULTIPLYING A(TRANPOSED).A
  DO 10 J=1, 7

```

```

      DO 10 M=1, 36
      UATA(M,J)=0.0
10    CONTINUE
C
C
      DO 20 M=1, 36
      UATA(M,1)=ATA(M,1)
      UATA(M, 2)=ATA(M, 2)
      UATA(M, 3)=ATA(M, 3)
20    CONTINUE
C
C
      DO 30 J= 4, 6
      DO 30 M=1, 36
      UATA(M,J+ 0)=ATA(M,J)
30    CONTINUE
      DO 40 J= 7, 7
      DO 40 M=1, 36
      UATA(M,J+ 0)=ATA(M,J)
40    CONTINUE
      RETURN
      END
C-----
C-----
      SUBROUTINE UATACP
      COMMON/IC/XICCG( 36),RMS( 36)
      COMMON/CA/ATA( 36, 7),UATA( 36, 7)
      DIMENSION KDIAG( 50)
      RNDOFF=SQRT(EXP(- 4**ALOG(2.0)))
      DO 100 J=1, 24
C  TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
      UDMAX=0
      DO 501 KP=2, 7
      UDARS=APS(UATA(J,KP))
      IF(UDARS.GT.UDMAX) UDMAX=UDARS
501  CONTINUE
      ALIMIT=RNDOFF*UDMAX
      IF(UATA(J,1).LT.ALIMIT)
      .UATA(J,1)= ALIMIT
      UATA(J,1)=SQRT(UATA(J,1))
      DO 1 KP=2, 7
      1  UATA(J,KP)=UATA(J,KP)/UATA(J,1)
      R=UATA(J, 2)
      UATA(J+ 1, 1)=UATA(J+ 1, 1)-R*UATA(J, 2)
      UATA(J+ 1, 2)=UATA(J+ 1, 2)-R*UATA(J, 3)
      UATA(J+ 1, 4)=UATA(J+ 1, 4)-R*UATA(J, 5)
      UATA(J+ 1, 5)=UATA(J+ 1, 5)-R*UATA(J, 6)
      R=UATA(J, 3)
      UATA(J+ 2, 1)=UATA(J+ 2, 1)-R*UATA(J, 3)
      UATA(J+ 2, 4)=UATA(J+ 2, 4)-R*UATA(J, 6)
      R=UATA(J, 4)
      UATA(J+ 5, 1)=UATA(J+ 5, 1)-R*UATA(J, 4)
      UATA(J+ 5, 2)=UATA(J+ 5, 2)-R*UATA(J, 5)
      UATA(J+ 5, 3)=UATA(J+ 5, 3)-R*UATA(J, 6)
      UATA(J+ 5, 6)=UATA(J+ 5, 6)-R*UATA(J, 7)
      R=UATA(J, 5)
      UATA(J+ 6, 1)=UATA(J+ 6, 1)-R*UATA(J, 5)
      UATA(J+ 6, 2)=UATA(J+ 6, 2)-R*UATA(J, 6)

```

```

      UATA(J+ 6, 5)=UATA(J+ 6, 5)-R*UATA(J, 7)
      R=UATA(J, 6)
      UATA(J+ 7, 1)=UATA(J+ 7, 1)-R*UATA(J, 6)
      UATA(J+ 7, 4)=UATA(J+ 7, 4)-R*UATA(J, 7)
      R=UATA(J, 7)
      UATA(J+ 12, 1)=UATA(J+ 12, 1)-R*UATA(J, 7)
100 CONTINUE
      DO 101 J= 25, 29
C   TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
      UDMAX=0
      DO 502 KP=2, 6
      UDABS=ABS(UATA(J,KP))
      IF(UDABS.GT.UDMAX) UDMAX=UDABS
502 CONTINUE
      ALIMIT=PNOFF*UDMAX
      IF(UATA(J,1).LT.ALIMIT)
      .UATA(J,1)= ALIMIT
      UATA(J,1)=SORT(UATA(J,1))
      DO 2 KP=2, 6
2   UATA(J,KP)=UATA(J,KP)/UATA(J,1)
      R=UATA(J, 2)
      UATA(J+ 1, 1)=UATA(J+ 1, 1)-R*UATA(J, 2)
      UATA(J+ 1, 2)=UATA(J+ 1, 2)-R*UATA(J, 3)
      UATA(J+ 1, 4)=UATA(J+ 1, 4)-R*UATA(J, 5)
      UATA(J+ 1, 5)=UATA(J+ 1, 5)-R*UATA(J, 6)
      R=UATA(J, 3)
      UATA(J+ 2, 1)=UATA(J+ 2, 1)-R*UATA(J, 3)
      UATA(J+ 2, 4)=UATA(J+ 2, 4)-R*UATA(J, 6)
      R=UATA(J, 4)
      UATA(J+ 5, 1)=UATA(J+ 5, 1)-R*UATA(J, 4)
      UATA(J+ 5, 2)=UATA(J+ 5, 2)-R*UATA(J, 5)
      UATA(J+ 5, 3)=UATA(J+ 5, 3)-R*UATA(J, 6)
      R=UATA(J, 5)
      UATA(J+ 6, 1)=UATA(J+ 6, 1)-R*UATA(J, 5)
      UATA(J+ 6, 2)=UATA(J+ 6, 2)-R*UATA(J, 6)
      R=UATA(J, 6)
      UATA(J+ 7, 1)=UATA(J+ 7, 1)-R*UATA(J, 6)
101 CONTINUE
      J= 30
C   TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED
      UDMAX=0
      DO 503 KP=2, 5
      UDABS=ABS(UATA(J,KP))
      IF(UDABS.GT.UDMAX) UDMAX=UDABS
503 CONTINUE
      ALIMIT=PNOFF*UDMAX
      IF(UATA(J,1).LT.ALIMIT)
      .UATA(J,1)= ALIMIT
      UATA(J,1)=SORT(UATA(J,1))
      DO 3 KP=2, 5
3   UATA(J,KP)=UATA(J,KP)/UATA(J,1)
      R=UATA(J, 2)
      UATA(J+ 1, 1)=UATA(J+ 1, 1)-R*UATA(J, 2)
      UATA(J+ 1, 2)=UATA(J+ 1, 2)-R*UATA(J, 3)
      UATA(J+ 1, 4)=UATA(J+ 1, 4)-R*UATA(J, 5)
      R=UATA(J, 3)
      UATA(J+ 2, 1)=UATA(J+ 2, 1)-R*UATA(J, 3)
      R=UATA(J, 4)

```

```

UATA(J+ 5, 1)=UATA(J+ 5, 1)-R*UATA(J, 4)
UATA(J+ 5, 2)=UATA(J+ 5, 2)-R*UATA(J, 5)
R=UATA(J, 5)
UATA(J+ 6, 1)=UATA(J+ 6, 1)-R*UATA(J, 5)
J= 31

```

C TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED

```

UDMAX=0
DO 504 KP=2, 4
  UDABS=ABS(UATA(J,KP))
  IF(UDABS.GT.UDMAX) UDMAX=UDABS

```

504 CONTINUE

```

ALIMIT=RNDOFF*UDMAX
IF(UATA(J,1).LT.ALIMIT)
  UATA(J,1)=ALIMIT
UATA(J,1)=SORT(UATA(J,1))
DO 4 KP=2, 4
  UATA(J,KP)=UATA(J,KP)/UATA(J,1)
  R=UATA(J, 2)
  UATA(J+ 1, 1)=UATA(J+ 1, 1)-R*UATA(J, 2)
  UATA(J+ 1, 2)=UATA(J+ 1, 2)-R*UATA(J, 3)
  R=UATA(J, 3)
  UATA(J+ 2, 1)=UATA(J+ 2, 1)-R*UATA(J, 3)
  R=UATA(J, 4)
  UATA(J+ 5, 1)=UATA(J+ 5, 1)-R*UATA(J, 4)
DO 102 J= 32, 34

```

C TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED

```

UDMAX=0
DO 505 KP=2, 3
  UDABS=ABS(UATA(J,KP))
  IF(UDABS.GT.UDMAX) UDMAX=UDABS

```

505 CONTINUE

```

ALIMIT=RNDOFF*UDMAX
IF(UATA(J,1).LT.ALIMIT)
  UATA(J,1)=ALIMIT
UATA(J,1)=SORT(UATA(J,1))
DO 5 KP=2, 3
  UATA(J,KP)=UATA(J,KP)/UATA(J,1)
  R=UATA(J, 2)
  UATA(J+ 1, 1)=UATA(J+ 1, 1)-R*UATA(J, 2)
  UATA(J+ 1, 2)=UATA(J+ 1, 2)-R*UATA(J, 3)
  R=UATA(J, 3)
  UATA(J+ 2, 1)=UATA(J+ 2, 1)-R*UATA(J, 3)

```

102 CONTINUE

J= 35

C TEST FOR CRITICAL PIVOT SIZE AND CORRECT IF REQUIRED

```

UDMAX=0
DO 506 KP=2, 2
  UDABS=ABS(UATA(J,KP))
  IF(UDABS.GT.UDMAX) UDMAX=UDABS

```

506 CONTINUE

```

ALIMIT=RNDOFF*UDMAX
IF(UATA(J,1).LT.ALIMIT)
  UATA(J,1)=ALIMIT
UATA(J,1)=SORT(UATA(J,1))
DO 6 KP=2, 2
  UATA(J,KP)=UATA(J,KP)/UATA(J,1)
  R=UATA(J, 2)
  UATA(J+ 1, 1)=UATA(J+ 1, 1)-R*UATA(J, 2)

```

```

J= 36
ALIMIT=PNDOFF
IF (UATA(J,1).LT.ALIMIT)
  UATA(J,1)=ALIMIT
UATA(J,1)=SORT(UATA(J,1))
RETURN
END

```

```

C-----
C-----
      SUBROUTINE ATADV(PRESULT,COLVEC)
      COMMON/IC/XICCG( 36),RHS( 36)
      COMMON/CA/ATA( 36, 7),UATA( 36, 7)
      DIMENSION RESULT(1),COLVEC(1)
C THIS SUBROUTINE PERFORMS A DOT PRODUCT OF MATRIX ATA
C WITH A COLUMN VECTOR
C COMPUTE RESULT=ATA.COLVEC
      DO 9 M=1, 36
        9 RESULT(M)=ATA(M,1)*COLVEC(M)
C
C
      DO 10 K= 2, 3
        IULIM= 37-K
        KP=K
        DO 10 M=1,IULIM
          RESULT(M)=RESULT(M)+ATA(M,KP)*COLVEC(K-1+M)
10 CONTINUE
C
C
      DO 20 K= 6, 8
        IULIM= 37-K
        KP=KP+1
        DO 20 M=1,IULIM
          RESULT(M)=RESULT(M)+ATA(M,KP)*COLVEC(K-1+M)
20 CONTINUE
C
C
      DO 30 K= 13, 13
        IULIM= 37-K
        KP=KP+1
        DO 30 M=1,IULIM
          RESULT(M)=RESULT(M)+ATA(M,KP)*COLVEC(K-1+M)
30 CONTINUE
C
C
      DO 40 K= 2, 3
        ILLIM=K
        KP=K
        DO 40 M=ILLIM, 36
          RESULT(M)=RESULT(M)+ATA(M-K+1,KP)*COLVEC(M-K+1)
40 CONTINUE
C
C
      DO 50 K= 6, 8
        ILLIM=K
        KP=KP+1
        DO 50 M=ILLIM, 36
          RESULT(M)=RESULT(M)+ATA(M-K+1,KP)*COLVEC(M-K+1)
50 CONTINUE

```

```

C
C      DO 40 K=    13,    13
        ILLIM=K
        KP=KP+1
        DO 60 M=ILLIM,    36
          RESULT(M)=RESULT(M)+ATA(M-K+1,KP)*COLVEC(M-K+1)
60      CONTINUE
        RETURN
      END

-----
C
C      SUBROUTINE ATDNTV(RESULT,COLVEC)
        COMMON/IC/XICCG( 36),RHS( 36)
        COMMON/C4/ATA( 36, 7),UATA( 36, 7)
        DIMENSION RESULT(1),COLVEC(1)
C      THIS SUBROUTINE PERFORMS A DOT PRODUCT OF THE TRANSPOSE OF
C      THE MATRIX A WITH A COLUMN VECTOR
C      COMPUTE RESULT=AT,COLVEC
C      UATA CONTAINS ADU AT THIS STAGE
C      ATA CONTAINS ADL AT THIS STAGE
        DO 9 M=1,    36
          9  RESULT(M)=UATA(M,1)*COLVEC(M)
          K= 2
          ILLIM=K
          KP=K
          DO 10 M=ILLIM,    36
            RESULT(M)=RESULT(M)+UATA(M-K+1,KP)*COLVEC(M-K+1)
10      CONTINUE
C
C
        DO 20 K=    7,    7
          ILLIM=K
          KP=KP+1
          DO 20 M=ILLIM,    36
            RESULT(M)=RESULT(M)+UATA(M-K+1,KP)*COLVEC(M-K+1)
20      CONTINUE
C
C
          K= 2
          IULIM= 37-K
          KP=K
          DO 30 M=1,IULIM
            RESULT(M)=RESULT(M)+ATA(M+K-1,KP)*COLVEC(M+K-1)
30      CONTINUE
C
C
        DO 40 K=    7,    7
          IULIM= 37-K
          KP=KP+1
          DO 40 M=1,IULIM
            RESULT(M)=RESULT(M)+ATA(M+K-1,KP)*COLVEC(M+K-1)
40      CONTINUE
        RETURN
      END

-----
C
C      SUBROUTINE UATATV(RESULT,COLVEC,AUX)

```



```

COMMON/IC/XICCG( 36),RHS( 36)
COMMON/CA/ATA( 36, 7),UATA( 36, 7)
DIMENSION RESULT(1),COLVEC(1),AUX(1)
C THIS SUBROUTINE PERFORMS THE OPERATION:
C (UATA(TRANPOSE)+UATA)**-1* A COLUATAMN VECTOR
C COMPUTE RESULT=(UATA(T),UATA)**-1*COLVEC
C FIRST AUX=UATA(T)**-1*COLVEC
C THEN RESULT=UATA**-1*AUX
C SECTION A
  AUX( 1)=(COLVEC( 1)
  .      1/UATA( 1,1)
  .
  .      AUX( 2)=(COLVEC( 2)
  .      -UATA( 1, 2)*AUX( 1)
  .      1/UATA( 2,1)
C SECTION B
  DO 10 M= 3, 5
10  AUX(M)=(COLVEC(M)
  .      -UATA(M- 1, 2)*AUX(M- 1)
  .      -UATA(M- 2, 3)*AUX(M- 2)
  .      1/UATA(M,1))
C SECTION C: RAND NUMBER IRAND= 2
  AUX( 6)=(COLVEC( 6)
  .      -UATA( 5, 2)*AUX( 5)
  .      -UATA( 4, 3)*AUX( 4)
  .      -UATA( 1, 4)*AUX( 1)
  .      1/UATA( 6,1)
  .      AUX( 7)=(COLVEC( 7)
  .      -UATA( 5, 2)*AUX( 6)
  .      -UATA( 5, 3)*AUX( 5)
  .      -UATA( 2, 4)*AUX( 2)
  .      -UATA( 1, 5)*AUX( 1)
  .      1/UATA( 7,1)
C SECTION D: RAND NUMBER IRAND= 2
  DO 20 M= 8, 12
20  AUX(M)=(COLVEC(M)
  .      -UATA(M- 1, 2)*AUX(M- 1)
  .      -UATA(M- 2, 3)*AUX(M- 2)
  .      -UATA(M- 5, 4)*AUX(M- 5)
  .      -UATA(M- 6, 5)*AUX(M- 6)
  .      -UATA(M- 7, 6)*AUX(M- 7)
  .      1/UATA(M,1)
C SECTION C: RAND NUMBER IRAND= 3
C SECTION D: RAND NUMBER IRAND= 3
  DO 30 M= 13, 36
30  AUX(M)=(COLVEC(M)
  .      -UATA(M- 1, 2)*AUX(M- 1)
  .      -UATA(M- 2, 3)*AUX(M- 2)
  .      -UATA(M- 5, 4)*AUX(M- 5)
  .      -UATA(M- 6, 5)*AUX(M- 6)
  .      -UATA(M- 7, 6)*AUX(M- 7)
  .      -UATA(M- 12, 7)*AUX(M- 12)
  .      1/UATA(M,1)
C
C
C NOW GENERATE BACKWARD SWEEP
C SECTION E
  RESULT( 36)=AUX( 36)/UATA( 36,1)
  RESULT( 35)=(AUX( 35)

```

```

      .      -UATA( 35, 2)*RESULT( 36)
      .      )/UATA( 35,1)
C SECTION F
  DO 40 M= 3, 5
    MR= 37-M
    40 RESULT(MR)=(AUX(MR)
      .      -UATA(MR, 2)*RESULT(MR+ 1)
      .      -UATA(MR, 3)*RESULT(MR+ 2)
      .      )/UATA(MR,1)
C SECTION G: RAND NUMBER IRAND= 2
  RESULT( 31)=(AUX( 31)
    .      -UATA( 31, 2)*RESULT( 32)
    .      -UATA( 31, 3)*RESULT( 33)
    .      -UATA( 31, 4)*RESULT( 34)
    .      )/UATA( 31,1)
  RESULT( 30)=(AUX( 30)
    .      -UATA( 30, 2)*RESULT( 31)
    .      -UATA( 30, 3)*RESULT( 32)
    .      -UATA( 30, 4)*RESULT( 35)
    .      -UATA( 30, 5)*RESULT( 36)
    .      )/UATA( 30,1)
C SECTION H: RAND NUMBER IRAND= 2
  DO 50 M= 8, 12
    MR= 37-M
    50 RESULT(MR)=(AUX(MR)
      .      -UATA(MR, 2)*RESULT(MR+ 1)
      .      -UATA(MR, 3)*RESULT(MR+ 2)
      .      -UATA(MR, 4)*RESULT(MR+ 5)
      .      -UATA(MR, 5)*RESULT(MR+ 6)
      .      -UATA(MR, 6)*RESULT(MR+ 7)
      .      )/UATA(MR,1)
C SECTION G: RAND NUMBER IRAND= 3
C SECTION H: RAND NUMBER IRAND= 3
  DO 60 M= 13, 36
    MR= 37-M
    60 RESULT(MR)=(AUX(MR)
      .      -UATA(MR, 2)*RESULT(MR+ 1)
      .      -UATA(MR, 3)*RESULT(MR+ 2)
      .      -UATA(MR, 4)*RESULT(MR+ 5)
      .      -UATA(MR, 5)*RESULT(MR+ 6)
      .      -UATA(MR, 6)*RESULT(MR+ 7)
      .      -UATA(MR, 7)*RESULT(MR+ 12)
      .      )/UATA(MR,2)
      RETURN
      END

```

ADL(I,3)	ADL(I,2)	ADL(I,1)	ADU(I,1)	ADU(I,2)	ADU(I,3)	XICCG(I)	QHS(I)
.100E+11	.100E+11	0.	0.	.100E+01	-.100E+00	.100E+01	.130E+01
.100E+11	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.320E+01
.100E+11	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.510E+01
.100E+11	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.700E+01
.100E+11	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.890E+01
.100E+11	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.108E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.126E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.144E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.162E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.180E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.198E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.216E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.234E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.252E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.270E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.288E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.306E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.324E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.342E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.360E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.378E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.396E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.414E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.432E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.450E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.468E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.486E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.504E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.522E+02
-.100E+00	.100E+01	0.	0.	.100E+01	-.100E+00	.100E+01	.540E+02
-.100E+00	.100E+01	0.	0.	.100E+01	.100E+11	.100E+01	.595E+02
-.100E+00	.100E+01	0.	0.	.100E+01	.100E+11	.100E+01	.614E+02
-.100E+00	.100E+01	0.	0.	.100E+01	.100E+11	.100E+01	.633E+02
-.100E+00	.100E+01	0.	0.	.100E+01	.100E+11	.100E+01	.652E+02
-.100E+00	.100E+01	0.	0.	.100E+01	.100E+11	.100E+01	.671E+02
-.100E+00	.100E+01	0.	0.	.100E+11	.100E+11	.100E+01	.320E+02

TIME= 361.27600

ITSTEP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
KNORM	.12E+03	.20E+03	.14E+03	.15E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03	.13E+03
DXNORM	.12E+03	.14E+03	.12E+03	.41E+02	.81E+02	.27E+01	.79E+01	.12E+01	.29E-03	.16E+00	.21E-01	.23E-01	.94E-03	.87E-06	.78E-04	.36E-05	.49E-05	.25E-06	.80E-09
RATIO	.10E+01	.71E+00	.84E+00	.27E+00	.64E+00	.21E-01	.62E-01	.98E-02	.23E-02	.12E-02	.16E-03	.18E-03	.74E-05	.68E-08	.61E-06	.29E-07	.39E-07	.20E-08	.63E-11
SIRI	0.	.99E-02	.23E-04	-.23E-04	-.16E-04	-.14E-07	-.36E-05	-.41E-07	-.14E-13	.61E-07	.66E-09	.23E-09	.11E-10	.12E-18	-.15E-13	-.19E-17	-.34E-16	-.17E-18	-.65E-25

TIME= 361.62200

ATA(I,1)	ATA(I,2)	ATA(I,3)	ATA(I,4)	ATA(I,5)	ATA(I,6)	ATA(I,7)	XICCG(I)
.101E+01	0.	.100E+01	-.100E+00	0.	-.200E+00	.100E-01	.100E+01
.201E+01	0.	.100E+01	-.200E+00	0.	-.200E+00	.100E-01	.200E+01
.201E+01	0.	.100E+01	-.200E+00	0.	-.200E+00	.100E-01	.300E+01
.201E+01	0.	.100E+01	-.200E+00	0.	-.200E+00	.100E-01	.400E+01
.201E+01	0.	.100E+01	-.200E+00	0.	-.200E+00	.100E-01	.500E+01