1 SUMMARY

This routine finds an approximate inverse **M** of an $n \times n$ sparse unsymmetric matrix **A** by attempting to minimize the difference between **AM** and the identity matrix in the Frobenius norm. The process may be improved by first performing a block triangularization of **A** and then finding approximate inverses to the resulting diagonal blocks.

A second entry allows the user to form the matrix-vector products

$$\mathbf{y} = \mathbf{M}\mathbf{z} \quad \text{and} \quad \mathbf{y} = \mathbf{M}^{\mathrm{T}}\mathbf{z}.$$

The principal use of such an approximate inverse is likely to be in preconditioning iterative methods for solving the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

ATTRIBUTES — **Version:** 1.1.0. (22 February 2005) **Types:** Real (single, double). **Calls:** `FD15`, `MC25`, `_NRM2`, `_AXPY`, `_COPY`, `_SCAL`, `_GEMV`, and `_TRSV`. **Language:** Fortran 77. **Original date:** April 1995. **Origin:** N.I.M. Gould and J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequence

There are three entries:

(a) `MI12I/ID` sets default values for control parameters. It should normally be called once prior to calls to `MI12A/AD`.

(b) `MI12A/AD` determines an approximate inverse preconditioner, **M**, one column at a time. Each column is chosen to be a sparse approximation to the actual column of the inverse. If required, the matrix is first reduced to block upper-triangular form, and the algorithm simply applied to each diagonal block.

(c) `MI12B/BD` performs the preconditioning operation $\mathbf{y} = \mathbf{M}\mathbf{z}$ or $\mathbf{y} = \mathbf{M}^{\mathrm{T}}\mathbf{z}$.

2.1.1 To set default values for the control parameters

The single precision version

```
      CALL MI12I( ICNTL, CNTL )
```

The double precision version

```
      CALL MI12ID( ICNTL, CNTL )
```

`ICNTL` is an `INTEGER` array of length `6` that need not be set by the user. On return it contains default values (see Section 2.2.1 for details).

`CNTL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `1` that need not be set by the user. On return it contains a default value (see Section 2.2.1 for details).

2.1.2 To form the approximate inverse preconditioner

The single precision version

```
      CALL MI12A ( N , A , IRNA  , LA    , IPCA  , IWORK , LIWORK,
     +             WORK  , LWORK , ICNTL , CNTL  , INFO )
```

The double precision version

```
      CALL MI12AD( N , A , IRNA  , LA    , IPCA  , IWORK , LIWORK,
     +             WORK  , LWORK , ICNTL , CNTL  , INFO )
```

N is an INTEGER variable that must be set by the user to *n*, the order of the matrix **A**. N is unaltered on exit. This argument is not altered by the routine. **Restriction**: N ≥ 1.

A is a REAL (DOUBLE PRECISION in the D version) array of length LA that must be set by the user to the values of the entries of the matrix **A** corresponding to the column indices set in IRNA. If there is more than one entry for a particular position, the values are accumulated. Any entry whose row index is out of range will be ignored; the value is reset to zero so that the calculation may continue. Otherwise, A is unaltered on exit.

IRNA is an INTEGER array of length LA. The first IPCA(N+1)-1 entries must be set by the user to hold the column indices of the entries of the matrix **A**. The entries must be ordered by rows, with the entries in each row contiguous and those of row I preceding those of row I+1 (I = 1, 2,..., N). The ordering within each row is unimportant. Any entry whose row index is out of range will be ignored; the row index is reset to 1 so that the calculation may continue. Otherwise, IRNA is unaltered on exit.

LA is an INTEGER variable that must be set by the user to the length of the arrays A, and IRNA. LA must be at least as large as the number of entries in the matrix **A**. LA is unaltered on exit. This argument is not altered by the routine. **Restriction**: LA ≥ IPCA(N+1)-1.

IPCA is an INTEGER array of length N+1 that must be set by the user so that IPCA(I) points to the position in the arrays IRNA and A of the first entry in row I (I = 1, 2,..., N). IPCA(N+1)–1 must be set to the number of entries in the matrix **A**. IPCA is unaltered on exit.

IWORK is an INTEGER array of length LIWORK that is used by the routine as workspace.

LIWORK is an INTEGER variable that must be set by the user to the length of the array IWORK. LIWORK may need to be as large as LINV + 10 * N + 1 + IPCA(N+1) + MAX(3 * N, IPCA(N+1) + ICNTL(5) + 1) where LINV is the space required to hold the entries of the approximate inverse. LINV may need to be as large as ICNTL(3) * N + N - ICNTL(3).

WORK is a REAL (DOUBLE PRECISION in the D version) array of length LWORK that is used by the routine as workspace.

LWORK is an INTEGER variable that must be set by the user to the length of the array WORK. LWORK may need to be as large as LINV + 4 * N + IPCA(N+1) - 1 + ICNTL(3) * (N+ICNTL(3)) where LINV is the space required to hold the entries of the approximate inverse. LINV may need to be as large as ICNTL(3) * N + N - ICNTL(3).

ICNTL is an INTEGER array of length 6 that contains control parameters. Default values for the components may be set by a call to MI12I/ID. Details of the control parameters are given in Section 2.2.1 This argument is not altered by the routine.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 1 that contains a control parameter and must be set by the user. A Default value for its component may be set by a call to MI12I/ID. Details of the control parameter are given in Section 2.2.1 This argument is not altered by the routine.

INFO is an INTEGER array of length 9 that need not be set by the user. It is used to hold information about the execution of the subroutine. On exit from MI12A/AD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.1. For information output in the other components, see Section 2.2.2.

2.2.2 To perform preconditioning operations

The single precision version

```
      CALL MI12B ( TRANS , N     , IWORK , LIWORK, WORK  , LWORK ,
     +             ICNTL , INFO  , Z , Y  )
```

The double precision version

```
      CALL MI12BD( TRANS , N     , IWORK , LIWORK, WORK  , LWORK ,
  +                ICNTL , INFO  , Z , Y  )
```

TRANS is a LOGICAL variable that must be set by the user. If TRANS=.TRUE., the preconditioning operation $\mathbf{y} = \mathbf{M}^{\mathrm{T}}\mathbf{z}$ is performed and if TRANS=.FALSE., the preconditioning operation $\mathbf{y} = \mathbf{M}\mathbf{z}$ is performed, where $\mathbf{M}$ is the approximate-inverse preconditioner. This argument is not altered by the routine.

N must all be unchanged since the call to MI12A/AD. This argument is not altered by the routine.

IWORK is an INTEGER array of length LIWORK that is used by the routine to hold the preconditioner. The first INFO(5) components of IWORK must not be altered between calls to MI12A/AD and MI12B/BD, nor between consecutive calls to MI12B/BD.

LIWORK is an INTEGER variable that must be set by the user to the length of the array IWORK. LIWORK should be at least INFO(5)

WORK is a REAL (DOUBLE PRECISION in the D version) array of length LWORK that is used by the routine to hold the preconditioner. The first INFO(6) components of WORK must not be altered between calls to MI12A/AD and MI12B/BD, nor between consecutive calls to MI12B/BD.

LWORK is an INTEGER variable that must be set by the user to the length of the array WORK. LWORK should be at least INFO(6).

Z is a REAL (DOUBLE PRECISION in the D version) array of length N that must be set by the user to hold the vector $\mathbf{z}$ which is to be preconditioned. This argument is not altered by the routine.

Y is a REAL (DOUBLE PRECISION in the D version) array of length N that need not be set by the user. On exit, Y holds the preconditioned vector $\mathbf{y}$

ICNTL is an INTEGER array of length 6 that contains control parameters. Default values for the components may be set by a call to MI12I/ID. Details of the control parameters are given in Section 2.2.1 This argument is not altered by the routine.

INFO is an INTEGER array of length 9 that need not be set by the user. On exit from MI12B/BD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.2. The other components are not accessed by the routine.

2.2 Arrays for control and information

2.2.1 Control parameters

The elements of the arrays ICNTL and CNTL control the action of MI12A/AD and MI12B/BD. Default values may be set by calling MI12I/ID.

ICNTL(1) is the stream number for error messages and has the default value 6. Printing of error messages is suppressed if ICNTL(1) ≤ 0.

ICNTL(2) is the stream number for warning messages and has the default value 6. Printing of warning messages is suppressed if ICNTL(2) ≤ 0.

ICNTL(3) specifies the maximum number of entries which are permitted in a column of the approximate inverse of any of the block diagonal blocks. It has default value 10.

ICNTL(4) specifies whether the improvement which may be made by introducing an extra nonzero into a column of the approximate inverse should be calculated exactly (ICNTL(4) = 1) or approximated (ICNTL(4) = 0). Approximating this value may lead to a faster execution at the expense of a poorer approximate inverse. ICNTL(4) has default value 1.

ICNTL(5) specifies the maximum number of new entries in a column of the approximate inverse which may be

simultaneously selected. The selection is made on the basis of improvement to the approximate inverse through introducing a new entry; at most ICNTL(5) entries may be selected at once, these being the ICNTL(5) entries which promise the best improvement. Note, however, that selecting ICNTL(5) >1 does not guarantee that the globally best entries will be selected, but just the locally best ones. A value of ICNTL(5) >1 may lead to a faster calculation. ICNTL(5) has default value 1.

ICNTL(6) specifies whether the matrix should be reduced to block form, and an approximate inverse of each diagonal block found (ICNTL(6) = 1), or whether the whole matrix should be treated as a single block (ICNTL(6) = 0). ICNTL(6) has default value 1.

CNTL(1) is the convergence tolerance. The calculation of each column $\mathbf{m}_{jj}^{i}$, of the approximate inverse $\mathbf{M}_{jj}$ of the block diagonal $\mathbf{A}_{jj}$, is terminated whenever $\|\mathbf{A}_{jj}\mathbf{m}_{jj}^{i}-\mathbf{e}_i\|_2 \leq$ CNTL(1), where $\mathbf{e}_i$ is the i–th column of the identity matrix. CNTL(1) has default value 0.1.

2.2.2 Information returned to the user

The array INFO is used to hold information which is returned to the user.

INFO(1) has the value zero if the call was successful, a positive value if a warning was issued, and a negative value in the event of a fatal error (see Section 2.3).

INFO(2) gives the number of columns of the approximate inverse for which the accuracy required by CNTL(1) was not attained.

INFO(3) gives the number of blocks used.

INFO(4) gives the size of the largest block.

INFO(5) gives the size of the leading part of IWORK which must be preserved between calls of MI12A/AD and MI12B/BD

INFO(6) gives the size of the leading part of WORK which must be preserved between calls of MI12A/AD and MI12B/BD

INFO(7) holds the number of out-of-range indices. These indices, and the values of their corresponding entries, are altered by the routine.

INFO(8) holds the total number nonzeros which represent the approximate inverse. This number is the sum of the number of entries in **A** which occur in off-diagonal blocks and the number of entries which occur in the approximate-inverses of the diagonal blocks.

INFO(9) holds the maximum number of entries in any column of the approximate inverse of a diagonal block.

2.3 Error diagnostics

2.3.1 Error returns from MI12A/AD

If MI12A/AD returns with a negative value of INFO(1), an error has occurred; if MI12A/AD returns with a positive value of INFO(1), a warning has been issued. Error messages are output on unit ICNTL(1) and warnings on unit ICNTL(2). Possible non-zero values of INFO(1) are given below.

–1 Value of LIWORK is insufficient. Increase LIWORK and retry.

–2 Value of LWORK is insufficient. Increase LWORK and retry.

–3 There is insufficient workspace to hold the approximate inverse preconditioner. Increase LIWORK and LWORK and retry.

–4 The matrix **A** is likely singular.

–5 Value of N out-of-range. N <1.

–6 Value of LA out-of-range. LA < IPRA(N+1)-1.

+1 The accuracy demanded by CNTL(1) was not always attained. See INFO(2).

+2 Out-of-range indices in IRNA. These indices, and the values of their corresponding entries, are altered by the routine. See INFO(7).

+3 Combination of warnings +1 and +2.

2.3.2 Error returns from MI12B/BD

If MI12B/BD returns with a negative value of INFO(1), an error has occurred. Error messages are output on unit ICNTL(1). There is only two possible error returns from MI12B/BD.

–1 Value of LIWORK is insufficient. Increase LIWORK and retry.

–2 Value of LWORK is insufficient. Increase LWORK and retry.

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: MI12A/AD and MI12B/BD call the internal routines MI12C/CD, MI12D/DD, MI12E/ED, and MI12F/FD. In addition, HSL routines FD15A/AD and MC25A/AD are called, along with the BLAS SNRM2/DNRM2, SAXPY/DAXPY, SCOPY/DCOPY, SSCAL/DSCAL, SGEMV/DGEMV, and STRSV/DTRSV.

Input/output: Error messages are printed on unit ICNTL(1) and warnings on unit ICNTL(2); see Section 2.3.

Restrictions:

N ≥ 1,

LAZ ≥ IPRA(N+1)-1.

4 METHOD

If required (ICNTL(6) = 1), the matrix is permuted to block upper-triangular form

$$\mathbf{PAQ} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdot & \mathbf{A}_{1b} \\ & A_{22} & \cdot & A_{2b} \\ \cdot & \cdot & \cdot & \cdot \\ & & & A_{bb} \end{pmatrix},$$

using permutations **P** and **Q**; otherwise, the entire matrix is treated as the single block $\mathbf{A}_{11} = \mathbf{A}$. An approximate inverse of each diagonal block $\mathbf{A}_{jj}$ is determined, one column at a time.

Starting with the approximation $\mathbf{m}_{jj}^{i} = 0$, the *i*–th column of the approximate inverse $\mathbf{M}_{jj}$ of $\mathbf{A}_{jj}$ is built up by introducing nonzeros in specified positions. Given a set of specified positions, $\mathbf{m}_{jj_i}$, is chosen to solve the least-squares problem of minimizing $\|\mathbf{A}_{jj}\mathbf{m}_{jj}^{i} - \mathbf{e}_i\|_2$, where $\mathbf{e}_i$ is the *i*–th column of the identity matrix. A new specified position is determined by considering how much the sum of squares will decrease if an extra nonzero is allowed in a previously unfilled position. This potential decrease may either be estimated (ICNTL(4) = 0) or calculated accurately (ICNTL(4) = 1). The unfilled position which predicts the largest potential decrease is added to the list of specified positions. To save computation, more than one candidate may be added at once; in this case the *k* candidates which predict the largest potential decrease may be added. This process is stopped when either $\|\mathbf{A}_{jj}\mathbf{m}_{jj}^{i} - \mathbf{e}_i\| \leq$ CNTL(1) or when $\mathbf{m}_{jj}^{i}$ has ICNTL(3) specified positions. The number of candidates *k* which may be added at one pass is bounded by ICNTL(5), but may be smaller if adding this many would exceed the maximum allowed in a column, ICNTL(3).

The product $\mathbf{y} = \mathbf{Mz}$ between the preconditioner and a vector is formed, in the obvious back-substitution fashion, by solving the triangular linear system

$$\begin{pmatrix} \mathbf{P}_{11}^{-1} & \mathbf{A}_{12} & \cdot & \mathbf{A}_{1b} \\ & P_{22}^{-1} & \cdot & \mathbf{A}_{2b} \\ & & \cdot & \cdot \\ & & & \mathbf{P}_{bb}^{-1} \end{pmatrix} \mathbf{w} = \mathbf{P}\mathbf{z}$$

and recovering $\mathbf{y} = \mathbf{Q}w$. Similarly, the product $\mathbf{y} = \mathbf{M}^T\mathbf{z}$ between the transpose of the preconditioner and a vector is formed by forward substitution in the triangular linear system

$$\begin{pmatrix} \mathbf{P}_{11}^{-T} & & & \\ \mathbf{A}_{12}^{T} & P_{22}^{-T} & & \\ & & \cdot & \\ \mathbf{A}_{1b}^{T} & \mathbf{A}_{2b}^{T} & \cdot & \mathbf{P}_{bb}^{-T} \end{pmatrix} \mathbf{w} = \mathbf{Q}^T\mathbf{z}$$

and then by recovering $\mathbf{y} = \mathbf{P}^T\mathbf{w}$.

References

J. D. F. Cosgrove and J. C. Diaz and A. Griewank (1992). Approximate Inverse Preconditionings for Sparse Linear Systems. International Journal on Computer Mathematics, **44**, 91-110.

N. I. M. Gould and J. A. Scott (1995) On approximate inverse preconditioners. Rutherford Appleton Laboratory, Chilton, England, in preparation.

T. Huckle and M. Grote (1994). A new approach to parallel preconditioning with sparse approximate inverses. Technical report SCCM-94-03, School of Engineering, Stanford University, California, USA.

5 EXAMPLE OF USE

We wish to solve the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ where

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 2 & 0 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 3 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 1 & 3 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 6 \\ 4 \\ 9 \\ 3 \\ 11 \\ 8 \\ 9 \\ 3 \\ 12 \end{pmatrix},$$

by forming an approximate inverse preconditioner for **A** and using this within the preconditioned GMRES(m) method. The following program, which illustrates the calling sequence for MI12, may be used.

```
      PROGRAM MAIN
C
C  Solve the linear system A x = b,
C  using the conjugate-gradients squared (CGS) method and an
C  approximate inverse preconditioner with at most 2 nonzeros in
C  each column.
C
      INTEGER N , LAEND, LA, MMAX, LINV, LIWORK, LWORK
      PARAMETER ( N = 9, LA = 52, LAEND = LA + N, MMAX = N )
      PARAMETER ( LINV   = MMAX * N + N - MMAX )
      PARAMETER ( LIWORK = LINV + 10 * N + 2 + LA + LA + MMAX + 2 )
      PARAMETER ( LWORK  = LINV + 4 * N + LA + MMAX * ( N + MMAX ) )
      INTEGER IRNA( LA ), IPCA( N + 1 ), IWORK( LIWORK )
      INTEGER INFO( 9 ), ICNTL( 6 )
      DOUBLE PRECISION A( LA ), WORK( LWORK ), CNTL( 1 )
C
```

```
C  variables for MI24
C
      INTEGER M, LDW, LDH
      PARAMETER ( M = 5, LDW = N, LDH = M + 1 )
      DOUBLE PRECISION RESID
      INTEGER INFO24( 4 ), ICNT24( 8 ), ISAV24( 17 )
      DOUBLE PRECISION W( LDW, M + 7 ), H( LDH, M + 2 ), CNTL24( 4 ),
     +  RSAV24( 9 )
      LOGICAL LSAV24( 4 )
      EXTERNAL MI24AD, MI24ID, MI12AD, MI12BD, MI12ID
      INTEGER I, IACT, K, LOCY, LOCZ
C
C  Set up A
C
      DATA ( A   ( I ), I =  1,  6 ) / 1D0, 1D0, 1D0, 2D0, 2D0, 1D0 /
      DATA ( IRNA( I ), I =  1,  6 ) / 1, 3, 5, 6, 7, 9 /
      DATA ( A   ( I ), I =  7,  9 ) / 1D0, 2D0, 3D0 /
      DATA ( IRNA( I ), I =  7,  9 ) / 3, 5, 9 /
      DATA ( A   ( I ), I = 10, 12 ) / 1D0, 1D0, 1D0 /
      DATA ( IRNA( I ), I = 10, 12 ) / 3, 5, 9 /
      DATA ( A   ( I ), I = 13, 18 ) / 1D0, 1D0, 1D0, 2D0, 3D0, 1D0 /
      DATA ( IRNA( I ), I = 13, 18 ) / 1, 3, 5, 6, 7, 9 /
      DATA ( A   ( I ), I = 19, 21 ) / 1D0, 2D0, 2D0 /
      DATA ( IRNA( I ), I = 19, 21 ) / 3, 5, 9 /
      DATA ( A   ( I ), I = 22, 27 ) / 1D0, 1D0, 1D0, 1D0, 1D0, 1D0 /
      DATA ( IRNA( I ), I = 22, 27 ) / 1, 3, 5, 6, 7, 9 /
      DATA ( A   ( I ), I = 28, 35 ) / 1D0, 1D0, 1D0, 1D0, 1D0, 1D0,
     *                                 1D0, 1D0 /
      DATA ( IRNA( I ), I = 28, 35 ) / 1, 2, 3, 4, 5, 6, 7, 9 /
      DATA ( A   ( I ), I = 36, 44 ) / 1D0, 1D0, 1D0, 1D0, 1D0, 1D0,
     *                                 1D0, 3D0, 1D0 /
      DATA ( IRNA( I ), I = 36, 44 ) / 1, 2, 3, 4, 5, 6, 7, 8, 9 /
      DATA ( A   ( I ), I = 45, 52 ) / 1D0, 2D0, 1D0, 1D0, 1D0, 1D0,
     *                                 1D0, 1D0 /
      DATA ( IRNA( I ), I = 45, 52 ) / 1, 2, 3, 4, 5, 6, 7, 9 /
      DATA ( IPCA( I ), I = 1, N + 1 ) / 1, 7, 10, 13, 19, 22, 28,
     *                                   36, 45, 53 /
C
C  Set up b
C
      DATA ( W( I, 1 ), I = 1, N ) / 6D0, 4D0, 9D0, 3D0, 11D0, 8D0,
     *                               9D0, 3D0, 12D0 /
C
C  Initialize data for the preconditioner
C
      CALL MI12ID( ICNTL, CNTL )
C
C  Allow at most 2 entries in each column of the preconditioner
C  and require column accuracy of no larger than 0.5
C
      ICNTL( 3 ) = 2
      CNTL( 1 ) = 0.5
C
C  Form the preconditioner
C
      CALL MI12AD( N , A , IRNA  , LA    , IPCA  , IWORK , LIWORK,
     *             WORK  , LWORK , ICNTL , CNTL  , INFO )
      IF ( INFO( 1 ) .LT. 0 ) THEN
         WRITE( 6, 2000 ) INFO( 1 )
         GO TO 50
      ELSE
```

```
         WRITE( 6, 2010 )
      END IF
C
C  Precondition from the left
C
      CALL MI24ID( ICNT24, CNTL24, ISAV24, RSAV24, LSAV24 )
      ICNT24( 3 ) = 1
      ICNT24( 6 ) = 100
C
C  Perform an iteration of the GMRES(m) method
C
      IACT = 0
   10 CONTINUE
      CALL MI24AD( IACT, N, M, W, LDW, LOCY, LOCZ, H, LDH, RESID,
     *             ICNT24, CNTL24, INFO24, ISAV24, RSAV24, LSAV24 )
      IF ( IACT .LT. 0 ) THEN
         WRITE( 6, 2040 ) INFO24( 1 )
         GO TO 50
      END IF
C
C  Perform the matrix-vector product
C
      IF ( IACT .EQ. 2 ) THEN
         DO 20 I = 1, N
            W( I, LOCY ) = 0.0
   20    CONTINUE
         DO 40 I = 1, N
            DO 30 K = IPCA( I ),IPCA( I + 1 ) - 1
               W( IRNA( K ), LOCY ) = W( IRNA( K ), LOCY ) +
     *                                A( K ) * W( I, LOCZ )
   30       CONTINUE
   40    CONTINUE
         GO TO 10
      END IF
C
C  Perform the left preconditioning operation
C
      IF ( IACT .EQ. 3 ) THEN
         CALL MI12BD( .FALSE., N, IWORK, LIWORK, WORK, LWORK,
     *                ICNTL , INFO,  W( 1, LOCZ ), W( 1, LOCY ) )
         IF ( INFO( 1 ) .LT. 0 ) THEN
            WRITE( 6, 2000 ) INFO( 1 )
            GO TO 50
         END IF
         GO TO 10
      END IF
C
C  Solution found
C
      WRITE( 6, 2020 ) INFO24( 2 ), ( W( I, 2 ), I = 1, N )
      IF ( INFO24( 1 ) .GT. 0 ) WRITE( 6, 2030 ) INFO24( 1 )
   50 CONTINUE
      STOP
 2000 FORMAT( ' Error exit from MI12. INFO( 1 ) = ', I2 )
 2010 FORMAT( ' Preconditioner formed successfully by MI12 ', / )
 2020 FORMAT( I6,' iterations required by MI24 ', // ' Solution = ',
     *       / ( 1P, 5D10.2 ) )
 2030 FORMAT( ' Error return: INFO( 1  ) = ',I2,' on exit from MI24 ' )
 2040 FORMAT( ' Warning: INFO( 1  ) = ',I2,' on exit from MI24 ' )
      END
```

This produces the following output:

```
 Preconditioner formed successfully by MI12

     3 iterations required by MI24

 Solution =
  1.00D+00  1.00D+00  1.00D+00  1.00D+00  1.00D+00
  1.00D+00  1.00D+00  1.00D+00  1.00D+00
```