1 SUMMARY

This routine uses **the BiCG (BiConjugate Gradient) method to solve the $n\times n$ unsymmetric linear system $\mathbf{Ax}=\mathbf{b}$, optionally using preconditioning.** If $\mathbf{P}_L$, $\mathbf{P}_R$ are the preconditioning matrices, the routine actually solves the preconditioned system

$$\overline{\mathbf{A}}\overline{\mathbf{x}}=\overline{\mathbf{b}},$$

with $\overline{\mathbf{A}}=\mathbf{P}_L\mathbf{A}\mathbf{P}_R$ and $\overline{\mathbf{b}}=\mathbf{P}_L\mathbf{b}$ and recovers the solution $\mathbf{x}=\mathbf{P}_R\overline{\mathbf{x}}$. If $\mathbf{P}_L=\mathbf{I}$, preconditioning is said to be from the right, if $\mathbf{P}_R=\mathbf{I}$, it is said to be from the left, and otherwise it is from both sides. Reverse communication is used for preconditioning operations $\mathbf{Pz}$ and $\mathbf{P}^T\mathbf{z}$, where $\mathbf{P}=\mathbf{P}_L\mathbf{P}_R$, and for matrix-vector products of the form $\mathbf{Az}$ and $\mathbf{A}^T\mathbf{z}$.

ATTRIBUTES — **Version:** 1.1.0. (22 February 2005) **Types:** Real (single, double). **Calls:** FD15, _COPY, _DOT, _NRM2, _AXPY. **Language:** Fortran 77. **Original date:** March 2001. **Origin:** N.I.M. Gould and J.A. Scott, Rutherford Appleton Laboratory. **Remark:** This is a threadsafe version of MI05A and supersedes it.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequence

There are two entries:

(a) MI25I/ID sets default values for control parameters. It should normally be called once prior to calls to MI25A/AD.

(b) MI25A/AD uses the BiConjugate Gradient method to solve $\mathbf{Ax}=\mathbf{b}$, optionally using preconditioning. MI25A/AD uses reverse communication for preconditioning operations and matrix-vector products.

2.1.1 To set default values for the control parameters

The single precision version

```
CALL MI25I(ICNTL,CNTL,ISAVE,RSAVE)
```

The double precision version

```
CALL MI25ID(ICNTL,CNTL,ISAVE,RSAVE)
```

ICNTL is an INTEGER array of length 8 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 5 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

ISAVE is an INTEGER array of length 14 used by MI25 as private workspace and must not be altered by the user.

RSAVE is a REAL (DOUBLE PRECISION in the D version) array of length 8 used by MI25 as private workspace and must not be altered by the user.

2.1.2 To solve $\mathbf{Ax}=\mathbf{b}$

The single precision version

```
CALL MI25A(IACT,N,W,LDW,LOCY,LOCZ,RESID,ICNTL,CNTL,INFO,ISAVE,RSAVE)
```

The double precision version

```
CALL MI25AD(IACT,N,W,LDW,LOCY,LOCZ,RESID,ICNTL,CNTL,INFO,ISAVE,RSAVE)
```

IACT is an INTEGER variable. Prior to the first call to MI25A/AD, IACT must be set by the user to 0. On each exit, IACT indicates the action required by the user. Possible values of IACT and the action required are as follows:

–1 An error has occurred and the user must terminate the computation (see INFO(1)).

1 If ICNTL(4)=0 (the default), convergence has been achieved and the user should terminate the computation. If ICNTL(4) is nonzero, the user may test for convergence. If the user does not wish to test for convergence (we do not recommend the user tests for convergence each time IACT=1 is returned) or if convergence has not been achieved, the user must recall MI25A/AD without changing any of the arguments.

2 The user must perform the matrix-vector products

$$\mathbf{y}_1 := \mathbf{A}\mathbf{z}_1 \tag{2.2a}$$

and

$$\mathbf{y}_2 := \mathbf{A}^{\mathrm{T}}\mathbf{z}_2 \tag{2.2b}$$

and recall MI25A/AD. The vectors $\mathbf{y}_i$ and $\mathbf{z}_i$ ($i=1,2$) are held in columns LOCY(i) and LOCZ(i) of array W, respectively. The vectors $\mathbf{z}_i$ must be unchanged by the user.

3 The user must perform the preconditioning operations

$$\mathbf{y}_1 := \mathbf{P}_{\mathrm{L}}\mathbf{P}_{\mathrm{R}}\mathbf{z}_1, \tag{2.1a}$$

and

$$\mathbf{y}_2 := \mathbf{P}_{\mathrm{R}}^{\mathrm{T}}\mathbf{P}_{\mathrm{L}}^{\mathrm{T}}\mathbf{z}_2, \tag{2.1b}$$

where $\mathbf{P}_{\mathrm{L}}$, $\mathbf{P}_{\mathrm{R}}$ are the preconditioning matrices, and recall MI25A/AD. The vectors $\mathbf{y}_i$ and $\mathbf{z}_i$ ($i=1,2$) are held in columns LOCY(i) and LOCZ(i) of array W, respectively. The vectors $\mathbf{z}_i$ must be unchanged by the user.

4 The user must perform the matrix-vector product

$$\mathbf{y}_1 := \mathbf{A}\mathbf{z}_1 \tag{2.3}$$

and recall MI25A/AD. The vectors $\mathbf{y}_1$ and $\mathbf{z}_1$ are held in columns LOCY(1) and LOCZ(1) of array W, respectively. The vectors $\mathbf{z}_1$ must be unchanged by the user. Note that IACT=4 can only be returned if ICNTL(5) is nonzero (see Section 2.2).

N is an INTEGER variable that must be set by the user to n, the order of the matrix $\mathbf{A}$. This variable must be preserved by the user between calls to MI25A/AD. This argument is not altered by the routine. **Restriction**: N≥1.

W is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LDW, 7). Prior to the first call, column 1 must hold the right-hand side vector $\mathbf{b}$ and, if ICNTL(5) is nonzero, column 2 must hold the initial estimate of the solution vector $\mathbf{x}$. On exit with IACT=1, the holds the current residual vector $\mathbf{r}=\mathbf{b}-\mathbf{Ax}$, and the current estimate of the solution $\mathbf{x}$ is held in column 2. On exit with IACT>1, the user is required to calculate $\mathbf{y}_i$ in column LOCY(i) ($i=1,2$) of W (see argument IACT). The remaining columns of W must not be altered by the user between calls to MI25A/AD.

LDW is an INTEGER variable that must be set by the user to the first dimension of the array W. This argument is not altered by the routine. **Restriction:** LDW≥N.

LOCY,LOCZ are INTEGER arrays of length 2 that need not be set by the user. On exit with IACT > 1, they indicate which columns of W contain the vectors $\mathbf{y}_i$ and $\mathbf{z}_i$ (see argument IACT). These arguments must not be altered by the user between calls to MI25A/AD.

RESID is a REAL (DOUBLE PRECISION in the D version) variable that need not be set by the user. On exit with IACT = 1, RESID holds the 2-norm of the current residual vector $\|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2$, where $\mathbf{x}$ is the current estimate of the solution.

ICNTL is an INTEGER array of length 8 that contains control parameters. Default values for the components may be set by a call to MI25I/ID. Details of the control parameters are given in Section 2.2. This argument is not altered by the routine.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 5 that contains control parameters and must be set by the user. Default values for the components may be set by a call to MI25I/ID. Details of the control parameters are given in Section 2.2. This argument is not altered by the routine.

INFO is an INTEGER array of length 4 that need not be set by the user. It is used to hold information about the execution of the subroutine. On exit from MI25A/AD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3. INFO(2) holds the number of iterations performed. INFO(3) and INFO(4) are not currently used.

ISAVE is an INTEGER array of length 14 used by MI25 as private workspace and must not be altered by the user.

RSAVE is a REAL (DOUBLE PRECISION in the D version) array of length 8 used by MI25 as private workspace and must not be altered by the user.

2.2 Control parameters

The elements of the arrays ICNTL and CNTL control the action of MI25A/AD. Default values may be set by calling MI25I/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) controls whether the user wishes to use preconditioning. It has default value 0 and in this case no preconditioning is used. If ICNTL(3) is nonzero, the user will be expected to perform preconditioning.

ICNTL(4) controls whether the convergence test offered by MI25A/AD is to be used. It has default value 0 and in this case the computed solution $\mathbf{x}$ is accepted if $\|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2$ is less than or equal to max($\|\mathbf{b}-\mathbf{A}\mathbf{x}^{(0)}\|_2$ *CNTL(1), CNTL(2)), where $\mathbf{x}^{(0)}$ is the initial estimate of the solution. If the user does not want to use this test for convergence, ICNTL(4) should be nonzero. In this case, the user may test for convergence when IACT = 1 is returned.

ICNTL(5) controls whether the user wishes to supply an initial estimate of the solution vector $\mathbf{x}$. It has default value 0 and in this case $\mathbf{x} = (0, 0, ..., 0)^{\mathrm{T}}$ is used as the initial estimate. If the user wishes to supply an initial estimate, ICNTL(5) should be nonzero and the initial estimate placed in the first N entries of column 2 of the array W prior to the first call to MI25A/AD.

ICNTL(6) determines the maximum number of iterations allowed. It has default value –1 and in this case the maximum number of iterations allowed is n. If the user does not want the maximum to be n, ICNTL(6) should be set to the maximum number of iterations the user does wish to allow. Values of ICNTL(6) which are less than or equal to zero are treated as if they were the default –1.

ICNTL(7) and ICNTL(8) are set to zero by MI25I/ID but not currently used by MI25A/AD.

CNTL(1) and CNTL(2) are the convergence tolerances (see Section 4). CNTL(1) has default value $\sqrt{u}$, where u is the relative machine precision (that is, the smallest machine number such that $1+u>1$), while CNTL(2) has default

value zero. If ICNTL(4) is nonzero, CNTL(1) and CNTL(2) are not accessed by MI25A/AD.

CNTL(3) is the breakdown tolerance (see Section 4). It has default u, where u is the relative machine precision.

CNTL(4) and CNTL(5) are set to zero by MI25I/ID but not currently used by MI25A/AD.

2.3 Error diagnostics

If MI25A/AD returns with a negative value of INFO(1), an error has occurred; if MI25A/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 N out of range. N < 1. Immediate return with input parameters unchanged.

–2 Value of LDW out of range. LDW < N. Immediate return with input parameters unchanged.

–3 Algorithm has broken down (see Section 4).

–4 The maximum number of iterations determined by the control parameter ICNTL(6) has been exceeded.

+1 The user-supplied convergence tolerance CNTL(1) lies outside the interval (u,1.0), where u is the machine precision. CNTL(1) is reset to the default convergence tolerance $\sqrt{u}$.

2.4 Underflows

The nature of the calculations performed in this subroutine means that underflows are likely to occur. It is quite safe to set numbers that underflow to zero, and action by the user may be required to ensure that this is done efficiently by the computing system in use.

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: MI25A/AD calls HSL routine FD15A/AD, and the BLAS kernels SNRM2/DNRM2, SCOPY/DCOPY, SAXPY/DAXPY, SDOT/DDOT.

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

Restrictions:

N ≥ 1,
LDW ≥ N.

4 METHOD

The BiConjugate Gradient method is due to Fletcher (1976) The algorithm used by `MI25A/AD` proceeds as follows:

Check the input data for errors. Set `INFO(1)` and return with `IACT` $=-1$ if a fatal error is encountered.
if `ICNTL(5)` is nonzero
  let $\mathbf{x}^{(0)}$ be the initial guess supplied by the user
  Return with `IACT` $=4$ for the user to compute $\mathbf{A}\mathbf{x}^{(0)}$.
else
  set $\mathbf{x}^{(0)} = (0, 0, ..., 0)^{\mathrm{T}}$
end if
Compute the initial residual $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$.
Set $\tilde{\mathbf{r}}^{(0)} = \mathbf{r}^{(0)}$
if `ICNTL(6)` is greater then zero
  `ITMAX` $=$ `ICNTL(6)`
else
  `ITMAX` $=$ `N`
end if
do $i = 1, 2, ...,$ `ITMAX`
  if `ICNTL(3)` is nonzero
    Return with `IACT` $=3$ for the user to compute $\mathbf{z}^{(i-1)} = \mathbf{P}_{\mathrm{L}}\mathbf{P}_{\mathrm{R}}\mathbf{r}^{(i-1)}$ and $\tilde{\mathbf{z}}^{(i-1)} = \mathbf{P}_{\mathrm{R}}^{\mathrm{T}}\mathbf{P}_{\mathrm{L}}^{\mathrm{T}}\tilde{\mathbf{r}}^{(i-1)}$.
  else
    $\mathbf{z}^{(i-1)} = \mathbf{r}^{(i-1)}$ and $\tilde{\mathbf{z}}^{(i-1)} = \tilde{\mathbf{r}}^{(i-1)}$
  end if
  $\rho_{i-1} = (\mathbf{z}^{(i-1)})^{\mathrm{T}}\tilde{\mathbf{r}}^{(i-1)}$
  if $|\rho_{i-1}| <$ `CNTL(3)*N`
    if $|\rho_{i-1}| <$ `CNTL(3)*`$\|\mathbf{z}^{(i-1)}\|_2 *\|\tilde{\mathbf{r}}^{(i-1)}\|_2$ the method has broken down.
    Set `INFO(1)` and return with `IACT` $=-1$.
  end if
  if $i = 1$
    $\mathbf{p}^{(1)} = \mathbf{z}^{(0)}$
    $\tilde{\mathbf{p}}^{(1)} = \tilde{\mathbf{z}}^{(0)}$
  else
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $\mathbf{p}^{(i)} = \mathbf{z}^{(i-1)} + \beta_{i-1}\mathbf{p}^{(i-1)}$
    $\tilde{\mathbf{p}}^{(i)} = \tilde{\mathbf{z}}^{(i-1)} + \beta_{i-1}\tilde{\mathbf{p}}^{(i-1)}$
  end if
  Return with `IACT` $=2$ for the user to compute $\mathbf{q} = \mathbf{A}\mathbf{p}^{(i)}$ and $\tilde{\mathbf{q}} = \mathbf{A}^{\mathrm{T}}\tilde{\mathbf{p}}^{(i)}$.
  $\alpha_i = \rho_{i-1}/(\tilde{\mathbf{p}}^{(i)})^{\mathrm{T}}\mathbf{q}$
  $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i\mathbf{p}^{(i)}$
  $\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} - \alpha_i\mathbf{q}$
  $\tilde{\mathbf{r}}^{(i)} = \tilde{\mathbf{r}}^{(i-1)} - \alpha_i\tilde{\mathbf{q}}$
  if `ICNTL(4)` is zero,
    if $\|\mathbf{r}^{(k)}\|_2 \leq \max(\|\mathbf{r}^{(0)}\|_2 *$`CNTL(1),CNTL(2))` convergence has been achieved.
    Return with `IACT` $=1$.
  else
    Return with `IACT` $=1$ to allow the user to check for convergence.
  end if
end do

Reference

R. Fletcher (1976). Conjugate gradient methods for indefinite systems. *Lecture Notes in Mathematics*, volume 506, Springer-Verlag, Berlin.

5 EXAMPLE OF USE

The following program illustrates the calling sequence for MI25.

```
C  Solve the linear system A x = b, where A is tridiagonal with
C  superdiagonal values 1, subdiagonals -1 and diagonals 2, and where
C  the inverse of the diagonal of A is used as a preconditioner

C     .. Parameters ..
      INTEGER N,LDW
      PARAMETER (N=10,LDW=N)
      DOUBLE PRECISION TWO,ONE,THREE
      PARAMETER (TWO=2.0D+0,ONE=1.0D+0,THREE=3.0D+0)
C     ..
C     .. Local Scalars ..
      DOUBLE PRECISION RESID
      INTEGER I,IACT
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION CNTL(5),W(LDW,7)
      INTEGER ICNTL(8),INFO(4),LOCY(2),LOCZ(2)
      INTEGER ISAVE(14)
      DOUBLE PRECISION RSAVE(8)
C     ..
C     .. External Subroutines ..
      EXTERNAL MI25AD,MI25ID
C     ..

      CALL MI25ID(ICNTL,CNTL,ISAVE,RSAVE)
C Preconditioning is required
      ICNTL(3) = 1
C Supply an initial estimate of the solution.
      ICNTL(5) = 1
      W(1,2) = ONE
      DO 10 I = 2,N - 1
         W(I,2) = ONE/2.0D0
   10 CONTINUE
      W(N,2) = ONE

C  Set right hand side, b so that solution is x=0

      W(1,1) = THREE
      DO 20 I = 2,N - 1
         W(I,1) = TWO
   20 CONTINUE
      W(N,1) = ONE

C  Perform an iteration of the BiConjugate Gradient method

      IACT = 0

   30 CONTINUE
      CALL MI25AD(IACT,N,W,LDW,LOCY,LOCZ,RESID,ICNTL,CNTL,INFO,
     +            ISAVE,RSAVE)
```

```
      IF (IACT.LT.0) THEN
         WRITE (6,FMT=9020) INFO(1)
         GO TO 70
      END IF

      IF (IACT.EQ.2) THEN
C  Perform the matrix-vector products with A and A(T)
         W(1,LOCY(1)) = TWO*W(1,LOCZ(1)) + W(2,LOCZ(1))
         W(1,LOCY(2)) = TWO*W(1,LOCZ(2)) - W(2,LOCZ(2))
         DO 40 I = 2,N - 1
            W(I,LOCY(1)) = -W(I-1,LOCZ(1)) + TWO*W(I,LOCZ(1)) +
     +                      W(I+1,LOCZ(1))
            W(I,LOCY(2)) =  W(I-1,LOCZ(2)) + TWO*W(I,LOCZ(2)) -
     +                      W(I+1,LOCZ(2))
   40    CONTINUE
         W(N,LOCY(1)) = -W(N-1,LOCZ(1)) + TWO*W(N,LOCZ(1))
         W(N,LOCY(2)) =  W(N-1,LOCZ(2)) + TWO*W(N,LOCZ(2))
         GO TO 30
      END IF

      IF (IACT.EQ.3) THEN
C  Perform the preconditioning operations (P = P(T))
         DO 50 I = 1,N
            W(I,LOCY(1)) = W(I,LOCZ(1))/TWO
            W(I,LOCY(2)) = W(I,LOCZ(2))/TWO
   50    CONTINUE
         GO TO 30
      END IF

      IF (IACT.EQ.4) THEN
C  Perform the matrix-vector product with A only
         W(1,LOCY(1)) = TWO*W(1,LOCZ(1)) + W(2,LOCZ(1))
         DO 60 I = 2,N - 1
            W(I,LOCY(1)) = -W(I-1,LOCZ(1)) + TWO*W(I,LOCZ(1)) +
     +                      W(I+1,LOCZ(1))
   60    CONTINUE
         W(N,LOCY(1)) = -W(N-1,LOCZ(1)) + TWO*W(N,LOCZ(1))
         GO TO 30
      END IF

C  Solution found

      WRITE (6,FMT=9000) INFO(2), (W(I,2),I=1,N)
      IF (INFO(1).GT.0) WRITE (6,FMT=9010) INFO(1)

   70 CONTINUE

      STOP
 9000 FORMAT (I6,' iterations required by MI25 ',//' Solution = ',
     +       / (1P,5D10.2))
 9010 FORMAT (' Warning: INFO( 1 ) = ',I2,' on exit ')
 9020 FORMAT (' Error return: INFO( 1 ) = ',I2,' on exit ')
      END
```

This produces the following output:

```
     10 iterations required by MI25

  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  1.00D+00
```