1 SUMMARY

This routine uses the **SYMMBK method to solve the $n \times n$ symmetric but possibly indefinite linear system $\mathbf{Ax} = \mathbf{b}$, optionally using preconditioning.** If $\mathbf{PP}^T$ is the preconditioning matrix, the routine actually solves the preconditioned system

$$\overline{\mathbf{Ax}} = \overline{\mathbf{b}},$$

with $\overline{\mathbf{A}} = \mathbf{PAP}^T$ and $\overline{\mathbf{b}} = \mathbf{Pb}$ and recovers the solution $\mathbf{x} = \mathbf{P}^T \overline{\mathbf{x}}$. Reverse communication is used for preconditioning operations and matrix-vector products of the form **Az**.

ATTRIBUTES — Version: 1.0.0. **Types:** Real (single, double). **Calls:** `_LAEV2`. **Original date:** August 1996. **Origin:** N. I. M. Gould, Rutherford Appleton Laboratory. **Language:** Fortran 90.

2 HOW TO USE THE PACKAGE

Access to the package requires a `USE` statement such as

Single precision version

```
USE HSL_MI02_SINGLE
```

Double precision version

```
USE HSL_MI02_DOUBLE
```

If it is required to use both modules at the same time, the derived types `MI02_CONTROL_TYPE`, `MI02_INFO_TYPE`, `MI02_DATA_TYPE`, (Section 2.1), and the subroutines `MI02_INITIALIZE`, `MI02_SYMMBK` and `MI02_WIND_UP` (Section 2.2) must be renamed on one of the `USE` statements.

2.1 The derived data types

Three derived data types are accessible from the package.

2.1.1 The derived data type for holding control parameters

The derived data type `MI02_CONTROL_TYPE` is used to hold controlling data. The components of `MI02_CONTROL_TYPE` are:

`out` is a scalar variable of type default `INTEGER` which holds the stream number for informational messages. Printing of informational messages in `MI02_SYMMBK` is suppressed if `out` < `0`.

`error` is a scalar variable of type default `INTEGER` which holds the stream number for error messages. Printing of error messages in `MI02_SYMMBK` and `MI02_WIND_UP` is suppressed if `error` ≤ `0`.

`itmax` is a scalar variable of type default `INTEGER` which holds the maximum number of iterations which will be allowed in `MI02_SYMMBK`. If `itmax` is set to a negative number, it will be reset by `MI02_SYMMBK` to $n+1$.

`precondition` is a scalar variable of type default `LOGICAL` which is set `.TRUE.` if the user intends to provide a preconditioner and `.FALSE.` otherwise.

`own_stopping_rule` is a scalar variable of type default `LOGICAL` which is set `.TRUE.` if the user intends to provide the stopping rule and `.FALSE.` otherwise.

`stop_relative` and `stop_absolute` are scalar variables of type default `REAL` (double precision `REAL` in `HSL_MI02_DOUBLE`) which holds the relative and absolute convergence tolerances (see Section 4). If `own_stopping_rule` is `.TRUE.`, `stop_relative` and `stop_absolute` are not accessed by `MI02`. Otherwise, the computed solution **x** is accepted by `MI02_SYMMBK` if $\|\mathbf{Ax} - \mathbf{b}\|_2$ is less than or equal to max($\|\mathbf{Ax}_0 - \mathbf{b}\|_2$*stop_relative, stop_absolute), where $\mathbf{x}_0$ is the initial estimate of the solution.

norm_est is a scalar variable of type default REAL (double precision REAL in HSL_MI02_DOUBLE) which holds an upper bound for the two-norm of **A**. A negative value is reset by MI02_SYMMBK to $\sqrt{n}$. Such a value is sufficient to cope with any matrix whose elements are smaller than one in absolute value, but a better estimate may improve the performance of the algorithm.

2.1.2 The derived data type for informational parameters

The derived data type MI02_INFO_TYPE is used to hold parameters which give information about the progress and needs of the algorithm. The components of MI02_INFO_TYPE are:

rnorm is a scalar variable of type default REAL (double precision REAL in HSL_MI02_DOUBLE) which holds the two norm of the residual, $\|\mathbf{Ax}-\mathbf{b}\|_2$.

iter is a scalar variable of type default INTEGER which holds the current iteration count.

allocation_status is a scalar variable of type default INTEGER which gives the status of the most recent array allocation or deallocation.

status is a scalar variable of type default INTEGER which gives the current status of the algorithm. See Sections 2.3 and 2.4 for details.

2.1.3 The derived data type for holding problem data

The derived data type MI02_DATA_TYPE is used to hold all the data for a particular system between calls of MI02 procedures. All components of MI02_DATA_TYPE are private.

2.2 Argument lists and calling sequences

There are three procedures for user calls:

1. The subroutine MI02_INITIALIZE is used to set default values and initialize private data.
2. The subroutine MI02_SYMMBK is called repeatedly to solve the system. On each exit, the user is expected to provide additional information and, if necessary, re-enter the subroutine.
3. The subroutine MI02_WIND_UP is provided to allow the user to automatically deallocate array components of the private data, allocated by MI02_SYMMBK, at the end of the solution process. It is important to do this if the data object is re-used for another problem since MI02_INITIALIZE cannot test for this situation, and any existing associated targets will subsequently become unreachable.

2.2.1 The initialization subroutine

Default values are provided as follows:

```
CALL MI02_INITIALIZE( data, control )
```

data is a scalar INTENT(OUT) argument of type MI02_DATA_TYPE. It is used to hold data about the system being solved. A call to MI02_INITIALIZE will ensure that all internal pointer arrays are disassociated.

control is a scalar INTENT(OUT) argument of type MI02_CONTROL_TYPE which need not be set on input. On exit, control contains default values for the components out = -1, error = 6, itmax = -1, precondition = .TRUE., own_stopping_rule = .FALSE., stop_relative = SQRT(u), stop_absolute = 0.0 and norm_est = -1.0, where u is EPSILON(1.0) (EPSILON(1.0D0) in HSL_MI02_DOUBLE). These values should only be changed after calling MI02_INITIALIZE.

2.2.2 The linear system solution subroutine

The linear system solution algorithm is called as follows:

```
CALL MI02_SYMMBK( n, X, V_in, V_out, data, control, info )
```

n is a scalar INTENT(IN) argument of type default INTEGER which must be set to the number of unknowns, n. **Restriction:** $n > 0$.

X is an array INTENT(INOUT) argument of dimension n and type default REAL (double precision REAL in HSL_MI02_DOUBLE) which holds an estimate of the solution **x** of the linear system. On initial entry, X must contain an estimate of the solution. On exit, X contains the current best estimate of the solution.

V_in is an array INTENT(INOUT) argument of dimension n and type default REAL (double precision REAL in HSL_MI02_DOUBLE) which is used to pass information to MI02_SYMMBK. The required content of the array is under the control of the parameter info%status (see Section 2.3). On initial entry, V_in must contain the residual **Ax**–**b**.

V_out is a one-dimensional POINTER array of type default REAL (double precision REAL in HSL_MI02_DOUBLE) which is used to pass information from MI02_SYMMBK. The actual content of the array depends on the value of the parameter info%status (see Section 2.3). Its allocation status and value must not be altered by the user.

data is a scalar INTENT(INOUT) argument of type MI02_DATA_TYPE. It is used to hold data about the system being solved.

control is a scalar INTENT(IN) argument of type MI02_CONTROL_TYPE. Default values may be assigned by calling MI02_INITIALIZE prior to the first call to MI02_SYMMBK.

info is a scalar INTENT(INOUT) argument of type MI02_INFO_type. On initial entry, the component status must be set to 1. The remaining components need not be set. A successful call to MI02_SYMMBK is indicated when the component status has the value 0. For other return values of status, see Sections 2.3 and 2.4.

2.2.3 The termination subroutine

Pointer arrays holding private data are deallocated as follows:

```
CALL MI02_WIND_UP( data, control, info )
```

data is a scalar INTENT(INOUT) argument of type MI02_DATA_TYPE exactly as for MI02_SYMMBK. On exit, its pointer array components will have been deallocated.

control is a scalar INTENT(IN) argument of type MI02_CONTROL_TYPE exactly as for MI02_SYMMBK.

info is a scalar INTENT(INOUT) argument of type MI02_INFO_type exactly as for MI02_SYMMBK. Only the component status will be set on exit, and a successful call to MI02_WIND_UP is indicated when this component has the value 0. For other return values of status, see Section 2.4.

2.3 Reverse communication

A positive value of info%status on exit from MI02_SYMMBK indicates that the user needs to take appropriate action before re-entering the subroutine. Possible values are:

2. The user must perform the preconditioning operation

$$\mathbf{y} := \mathbf{P}\mathbf{P}^T\mathbf{z},$$

where $\mathbf{P}\mathbf{P}^T$ is the preconditioning matrix, and recall MI02_SYMMBK. The vector **z** is available as the first n components of the array V_out, and **y** must be placed in V_in. No argument except V_in should be altered before recalling MI02_SYMMBK.

3. The user must perform the matrix-vector product

$$\mathbf{y} := \mathbf{A}\mathbf{z}$$

and recall MI02_SYMMBK. The vector **z** is available as the first n components of the array V_out, and **y** must be placed in V_in. No argument except V_in should be altered before recalling MI02_SYMMBK.

4. The user should test for convergence. This value will only occur when the user has opted to test convergence by

setting `control%own_stopping_rule` to `.TRUE.`. If the user does not wish to test for convergence (we do not recommend the user tests for convergence each time `info%status` = 4 is returned) or if convergence has not been achieved, the user must recall `MI02_SYMMBK` without changing any of the arguments.

2.4 Warning and error messages

A negative value of `info%status` on exit from `MI02_SYMMBK` or `MI02_WIND_UP` indicates that an error has occurred. No further calls should be made until the problem has been resolved. Possible values are:

– 1. (`MI02_SYMMBK` only) The input parameter `n` is not positive.

– 2. (`MI02_SYMMBK` only) More than `control%itmax` iterations have been performed without obtaining convergence.

– 3. (`MI02_SYMMBK` only) The matrix **A** appears to be singular and the system inconsistent.

– 4. An array allocation (`MI02_SYMMBK`) or deallocation (`MI02_WIND_UP`) has failed. A message indicating the offending array is written on unit `control%error` and the returned allocation status is given by `info%allocation_status`.

2.5 Information printed

If `control%out` is positive, information about the progress of the algorithm will be printed on unit `control%out`. A one-line summary of each iteration will be given containing the iteration number, the norm of the residual, the latest diagonal and off-diagonal elements in the Lanczos tridiagonal matrix (see Section 4) and a flag indicating the pivot type used when factorizing this matrix.

3 GENERAL INFORMATION

Use of common: None.

Other modules used directly: `MI02_SYMMBK` calls the LAPACK subroutine `_LAEV2`.

Input/output: Output is under control of the arguments `control%error` and `control%out`.

Restrictions: `n` > 0.

4 METHOD

The method is iterative. Starting with the vector $(\mathbf{Ax}_0 - \mathbf{b})/\|\mathbf{P}^T(\mathbf{Ax}_0 - \mathbf{b})\|_2$, a matrix of Lanczos vectors is built one column at a time so that the *k*–th column is generated during iteration *k*. The resulting *n* by *k* matrix $\mathbf{Q}_k$ has the property that $\mathbf{Q}_k^T\mathbf{AQ}_k = \mathbf{T}_k$, where $\mathbf{T}_k$ is tridiagonal. An approximation to the required solution may then be expressed formally as

$$\mathbf{x}_{k+1} = \mathbf{x}_0 - \mathbf{Q}_k\mathbf{y}_k,$$

where $\mathbf{y}_k$ solves the tridiagonal system

$$\mathbf{T}_k\mathbf{y}_k = \|\mathbf{P}^T(\mathbf{Ax}_0 - \mathbf{b})\|_2\,\mathbf{e}_1 \qquad (4.1)$$

and $\mathbf{e}_1$ is the first unit vector.

SYMMBK forms a symmetric Bunch-Kaufman factorization of the Lanczos tridiagonal matrix $\mathbf{T}_k$. The matrix is decomposed into the product $\mathbf{T}_k = \mathbf{M}_k\mathbf{D}_k\mathbf{M}_k^T$, where $\mathbf{M}_k^T$ is unit lower bidiagonal amd $\mathbf{D}_k$ is block diagonal with blocks of dimension at most two. The factors are obtained as $\mathbf{T}_k$ is formed. At each stage stability issues dictate whether the current diagonal entry will form a 1 by 1 block or whether it will be combined with its neighbour as part of a 2 by 2 block. The form of (4.1) enables the efficient calculation of the solution $\mathbf{y}_k$. The particular form of this solution enables us to obtain $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$ without storing the complete matrix $\mathbf{Q}_k$ but merely its last two columns.

The aim of preconditioning is to accelerate the convergence of the method by clustering the eigenvalues of the preconditioned matrix $\overline{\mathbf{A}}$ around a small number of distinct values. If $\mathbf{A}$ is positive definite, this is often achieved by choosing $\mathbf{P}\mathbf{P}^T \approx \mathbf{A}^{-1}$. When $\mathbf{A}$ is indefinite, such a choice will not be possible, and the best that can be hoped for is that the eigenvalues of $\overline{\mathbf{A}}$ cluster around one positive and one negative value.

References

The basic method was proposed by

R. Chandra (1978). Conjugate gradient methods for partial differential equations. Ph. D. thesis, Yale University, New Haven, USA. Issued as technical report number 129.

The factorization strategy is essentially that of

J. R. Bunch and L. C. Kaufman (1977). Some stable methods for calculating inertia and solving symmetric linear equations. Math. Computation **31**, 163-179.

5 EXAMPLE OF USE

Suppose we wish to solve the linear system

$$\begin{pmatrix} 1 & & & & & 1 & & & & \\ & 2 & & & & & 1 & & & \\ & & 3 & & & & & 1 & & \\ & & & 4 & & & & & 1 & \\ & & & & 5 & & & & & 1 \\ 1 & & & & & & & & & \\ & 1 & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 1 & & & & & & \\ & & & & 1 & & & & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The coefficient matrix is indefinite, and we choose to precondition with the positive definite matrix

$$\begin{pmatrix} 1 & & & & & & & & & \\ & 1/2 & & & & & & & & \\ & & 1/3 & & & & & & & \\ & & & 1/4 & & & & & & \\ & & & & 1/5 & & & & & \\ & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \end{pmatrix}.$$

We start from $\mathbf{x} = \mathbf{0}$ and estimate the two-norm of $\mathbf{A}$ by its Gershgorin bound of 6. We may use the following code:

```
PROGRAM HSL_MI02_EXAMPLE
USE HSL_MI02_DOUBLE
IMPLICIT NONE
INTEGER, PARAMETER :: n = 10
INTEGER, PARAMETER :: working = KIND( 1.0D+0 )
REAL ( KIND = working ), DIMENSION( n ) :: X
REAL ( KIND = working ), DIMENSION( n ) :: V_in
REAL ( KIND = working ), POINTER, DIMENSION( : ) :: V_out
TYPE ( MI02_DATA_TYPE ) :: data
TYPE ( MI02_CONTROL_TYPE ) :: control
TYPE ( MI02_INFO_TYPE ) :: info
```

```
  INTEGER :: i
  CALL MI02_INITIALIZE( data, control )    ! Initialize control parameters
  control%norm_est = 6
  X = 0.0_working                          ! Set the initial point
  DO i = 1, 5                              ! Set the initial residual
     V_in( i ) = - i - 1
  END DO
  V_in( 6 : n ) = - 1
  info%status = 1
  DO                                       !  Solve the system
     CALL MI02_SYMMBK( n, X, V_in, V_out, data, control, info )
     SELECT CASE( info%status )
     CASE( 2 )                             !  Use the preconditioner
        DO i = 1, 5
           V_in( i ) = V_out( i ) / i
        END DO
        V_in(  6 : n ) = V_out( 6 : n )
     CASE( 3 )                             !  Form the matrix-vector product
        DO i = 1, 5
           V_in( i ) = i * V_out( i ) + V_out( i + 5 )
        END DO
        V_in( 6 : n ) = V_out( : 5 )
     CASE DEFAULT
        EXIT
     END SELECT
  END DO
  DO i = 1, 5                              !  Compute the final residual
     V_in( i ) = i * X( i ) + X( i + 5 ) - i - 1
  END DO
  V_in( 6 : n ) = X( : 5 ) - 1
  WRITE( 6, "( /, ' Output status = ', I6,                                   &
         &        ' norm of final residual = ', ES9.1 )" )                   &
         info%status, SQRT( DOT_PRODUCT( V_in, V_in ) )
  WRITE( 6, "( /, ' final x = ', //, ( 5ES12.4 ) )" ) X
  CALL MI02_WIND_UP( data, control, info )  !  Deallocate internal arrays
  END PROGRAM HSL_MI02_EXAMPLE
```

This produces the following output:

```
 Output status =      0 norm of final residual =   5.6E-15

 final x =

  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00
  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00
```