1 SUMMARY

To **solve a sparse symmetric system of linear equations.** Given a sparse symmetric matrix $\mathbf{A}=\{a_{ij}\}_{n\times n}$ and an n-vector $\mathbf{b}$ or a matrix $\mathbf{B}=\{b_{ij}\}_{n\times r}$, this subroutine solves the system $\mathbf{Ax}=\mathbf{b}$ or the system $\mathbf{AX}=\mathbf{B}$. The matrix $\mathbf{A}$ need not be definite. There is an option for iterative refinement.

The method used is a direct method based on a sparse variant of Gaussian elimination and is discussed further by Duff and Reid, ACM Trans. Math. Software **9** (1983), 302-325. A detailed discussion on the MA57 strategy and performance is given by Duff, ACM Trans. Math. Software **30** (2004), 118-144. Relevant work on pivoting and scaling strategies is given by Duff and Pralet, SIAM Journal Matrix Analysis and Applications **27** (2005), 313-340. More recent work on static pivoting is given by Duff and Pralet, SIAM Journal Matrix Analysis and Applications **29** (2007), 1007-1024.

The MA57 package has a range of options including several sparsity orderings, multiple right-hand sides, partial solutions, error analysis, scaling, a matrix modification facility, the efficient factorization of highly rank deficient systems, and a stop and restart facility. Although the default settings should work well in general, there are several parameters available to enable the user to tune the code for his or her problem class or computer architecture.

In the Fortran 95 version, there are added facilities for automatic restarts when storage limits are exceeded, the return of information on pivots, permutations, scaling, modifications, and the possibility to alter the pivots a posteriori. More recent additions exploit sparsity in the right-hand side, compute the Fredholm alternative, multiply vectors by the triangular factor, and return the factors in standard format.

ATTRIBUTES — Version: 5.2.0. (2 August 2013) **Types:** Real (single, double). **Remark:** HSL_MA57 is a Fortran 95 encapsulation of MA57 and offers some additional facilities to the Fortran 77 version. **Calls:** MA57, _GEMM, _TPMV, _TPSV, MC71, HSL_ZD11. **Language:** Fortran 95 + TR 15581 (allocatable components). **Original date:** Original: September 2000, revised March 2013. **Origin:** I.S. Duff, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Calling sequences

Access to the package requires a USE statement

Single precision version

```
USE HSL_MA57_SINGLE
```

Double precision version

```
USE HSL_MA57_DOUBLE
```

In HSL_MA57_SINGLE, all reals are default reals. In HSL_MA57_DOUBLE, all reals are double precision reals. In both modules, all integers are default integers.

There are five principal subroutines for user calls (see Section 2.5 for further features):

1. The subroutine MA57_INITIALIZE must be called to initialize the structure for the factors. It may also be called to set default values for the components of the control structure. If non-default values are wanted for any of the control components, the corresponding components should be altered after the call to MA57_INITIALIZE.
2. MA57_ANALYSE accepts the pattern of **A** and chooses pivots for Gaussian elimination using a selection criterion to preserve sparsity. It subsequently constructs subsidiary information for the actual factorization by MA57_FACTORIZE. The user may provide the pivot sequence, in which case only the necessary information for MA57_FACTORIZE will be generated.

3. MA57_FACTORIZE factorizes a matrix **A** using the information from a previous call to MA57_ANALYSE. The actual pivot sequence used may differ from that of MA57_ANALYSE if **A** is not definite.
4. MA57_SOLVE uses the factors generated by MA57_FACTORIZE to solve a system of equations with one (**Ax** = **b**) or several (**AX** = **B**) right-hand sides, or to improve a given solution or set of solutions by iterative refinement.
5. MA57_FINALIZE reallocates the arrays held inside the structure for the factors to have size zero. It should be called when all the systems involving its matrix have been solved unless the structure is about to be used for the factors of another matrix.

2.2 The derived data types

For each problem, the user must employ derived types defined by the module to declare structures for holding the matrix, holding its factors, controlling the factorization, and providing information.

2.2.1 Derived data type for the matrix

The derived type ZD11_TYPE is used to hold the matrix. The following components are employed

N is an INTEGER scalar which holds the order n of the matrix **A**. **Restriction:** N ≥ 1.

NE is an INTEGER scalar which holds the number of matrix entries. **Restriction:** NE ≥ 0.

VAL is a REAL allocatable array of length at least NE, the leading part of which holds the values of the entries. Each pair of off-diagonal entries $a_{ij} = a_{ji}$ is represented as a single entry. Duplicated entries are summed.

ROW is an INTEGER allocatable array of length at least NE, the leading part of which holds the row indices of the entries.

COL is an INTEGER allocatable array of length at least NE, the leading part of which holds the column indices of the entries.

The other components of the type are not used.

2.2.2 Derived data type for control of the subroutines

The module contains a derived type called MA57_CONTROL with the following components

LP is an INTEGER scalar used by the subroutines as the output stream for error messages. If it is negative, these messages will be suppressed. The default value is 6.

WP is an INTEGER scalar used by the subroutines as the output stream for warning messages. If it is negative, these messages will be suppressed. The default value is 6.

MP is an INTEGER scalar used by the subroutines as the output stream for diagnostic printing. If it is negative, these messages will be suppressed. The default value is 6.

LDIAG is an INTEGER scalar used by the subroutines to control diagnostic printing. If LDIAG is less than 1, no messages will output. If the value is 1, only error messages will be printed. If the value is 2, then error and warning messages will be printed. If the value is 3, scalar data and a few entries of array data on entry and exit from each subroutine will be printed. If the value is greater than 3, all data will be printed on entry and exit. The default value is 2.

LA is an INTEGER scalar used by MA57_FACTORIZE. If LA ≥ NRLNEC (see Section 2.2.3), the real array that holds data for the factors is reallocated to have size LA. Otherwise, the array is not reallocated unless its size is less than NRLNEC, in which case it is reallocated with size NRLTOT (see Section 2.2.3). The default value is 0.

LIW is an INTEGER scalar used by MA57_FACTORIZE. If LIW ≥ NIRNEC (see Section 2.2.3), the integer array that holds data for the factors is reallocated to have size LIW. Otherwise, the array is not reallocated unless its size is less than NIRNEC, in which case it is reallocated with size NIRTOT (see Section 2.2.3). The default value is 0.

MAXLA is an INTEGER scalar used by MA57_FACTORIZE. An error return occurs if the real array that holds data for the

factors is too small and reallocating it to have size changed by the factor MULTIPLIER would make its size greater than MAXLA. The default value is HUGE(0).

MAXLIW is an INTEGER scalar used by MA57_FACTORIZE. An error return occurs if the integer array that holds data for the factors is too small and reallocating it to have size changed by the factor MULTIPLIER would make its size greater than MAXLIW. The default value is HUGE(0).

MULTIPLIER is a REAL scalar used by MA57_FACTORIZE when a real or integer array that holds data for the factors is too small. The array is reallocated with its size changed by the factor MULTIPLIER. The default value is 2.0.

REDUCE is a REAL scalar that reduces the size of previously allocated internal workspace arrays if they are larger than currently required by a factor of REDUCE or more. The default value is 2.0.

NEMIN is an INTEGER scalar used by MA57_ANALYSE for the minimum number of eliminations in a step that is automatically accepted. If two adjacent steps can be combined and each has fewer eliminations, they are combined. The default value is 16.

THRESH is an INTEGER scalar used by MA57_ANALYSE to identify dense rows during pivot selection when the minimum degree algorithm from MA27 is being used (ORDERING = 3). It is the percentage density for a row to be regarded as dense. The default value is 100.

PIVOTING is a INTEGER scalar that is used to control numerical pivoting by MA57_FACTORIZE. It must have one of the following values:

1 Numerical pivoting will be performed, with relative pivot tolerance given by the component U.

2 No pivoting will be performed and an error exit will occur immediately a sign change is detected among the pivots. This is suitable for cases when **A** is thought to be definite and is likely to decrease the factorization time while still providing a stable decomposition.

3 No pivoting will be performed and an error exit will occur if a zero pivot is detected. This is likely to decrease the factorization time, but may be unstable if there is a sign change among the pivots.

4 No pivoting will be performed but the matrix will be altered if a non-positive pivot is encountered.

The default value is 1.

U is a REAL scalar that is used by MA57_FACTORIZE when the component PIVOTING has the value 1 to hold the relative pivot tolerance. The default value is 0.01. For problems requiring greater than average numerical care a higher value than the default would be advisable. Values greater than 0.5 are treated as 0.5 and less than 0.0 as 0.0.

TOLERANCE is a REAL scalar that is used by MA57_FACTORIZE. Any entry of modulus less than or equal to TOLERANCE is treated as zero. The default value is 10^{-20}. If RANK_DEFICIENT = 1, then blocks of entries less than TOLERANCE can be discarded during the factorization and the corresponding pivots are placed at the end of the ordering. In this case or when the Fredholm alternative entry is to be used, a normal value for TOLERANCE could be 10^{-12}.

CONVERGENCE is a REAL scalar that is used by MA57_SOLVE. Iterative refinement will stop if the ratio of the norm of the scaled residual in successive iterations is greater than CONVERGENCE. The default value is 0.5.

FACTORBLOCKING is an INTEGER scalar used by MA57_FACTORIZE to determine the block size used for the Level 3 BLAS. The default value is 16.

SOLVEBLOCKING is an INTEGER scalar used by MA57_SOLVE to determine when to use Level 2 and Level 3 BLAS. The default value is 10.

ORDERING is an INTEGER scalar used by MA57_ANALYSE to determine the sparsity ordering that is used. The default value is 5. Possible values are:

0 AMD ordering using MC47 (without the detection of dense rows).

2 AMD ordering using MC47.

3 Minimum degree ordering as generated by the MA27 code.

4 MeTiS ordering is used. Note that the user needs to supply the MeTiS library

(http://www-users.cs.umn.edu/~karypis/metis/metis/download.html).

If it is not supplied and this option is requested, the routine will return immediately with AINFO%FLAG set to -10.

5 If MeTiS is available, the routine will make an automatic selection of the ordering choosing either MeTiS or MC47. If MeTiS is not available, then action is taken as if ORDERING were set to 2.

≥ 6 Currently is equivalent to setting PIVOTING = 5, but may be used for alternative orderings in later releases of HSL_MA57.

SCALING is an INTEGER scalar used by MA57_FACTORIZE to control the scaling. This has default value 1 and indicates that the matrix will be scaled using a symmetrized version of the HSL code MC64. Setting SCALING to any other value will suppress this option although this parameter may be used to call other scalings in future releases.

STATIC_TOLERANCE is a REAL scalar that is used by MA57_FACTORIZE to control the static pivoting (see Section 4). It has default value 0.0. If the value is positive, we will only accept delayed pivots to a level defined by STATIC_LEVEL and small pivots will be increased so that the factorization could be inaccurate. When static pivots are chosen, HSL_MA57_SOLVE will automatically use iterative refinement.

STATIC_LEVEL is a REAL scalar that is used by MA57_FACTORIZE. Static pivoting is only invoked if STATIC_TOLERANCE is greater than zero and the accumulated number of delayed pivots exceeds STATIC_LEVEL*N. The default value is zero.

RANK_DEFICIENT is an INTEGER scalar used by MA57_FACTORIZE that has default value 0. If RANK_DEFICIENT is set to 1, then when small entries (defined by TOLERANCE) are detected, they are removed and the corresponding pivots placed at the end of the factorization. This can be particularly efficient if the matrix is highly rank deficient.

CONSIST is a REAL scalar that is used by MA57_FREDHOLM_ALTERNATIVE to determine whether the system being solved is consistent. It has default value 10^{-20}.

2.2.3 Derived data type for information from MA57_ANALYSE

The module contains a derived type called MA57_AINFO with the following components that are initialized to their default values when the object comes into existence

FLAG is an INTEGER scalar. The value zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.1.

MORE is an INTEGER scalar that provides further information in the case of an error, see Section 2.4.1.

OOR is an INTEGER scalar which is set to the number of entries with one or both indices out of range.

DUP is an INTEGER scalar which is set to the number of duplicate off-diagonal entries.

STAT is an INTEGER scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

NSTEPS is an INTEGER scalar which is set to the number of nodes in the assembly tree (number of major steps in the factorization).

MAXFRT is an INTEGER scalar that holds the largest front size.

OPSA is a REAL scalar which is set to the number of floating-point additions required by the assembly of frontal matrices if no pivoting is performed. Numerical pivoting may increase the number of operations.

OPSE is a REAL scalar which is set to the number of floating-point operations required by the factorization if no pivoting is performed. Numerical pivoting may increase the number of operations.

NRLTOT and NIRTOT are INTEGER scalars that give the total amount of REAL and INTEGER words respectively required for a successful factorization without the need for data compression, provided no numerical pivoting is performed.

NRLNEC and NIRNEC are INTEGER scalars that give the total amount of REAL and INTEGER words required respectively for successful factorization allowing data compression, provided no numerical pivoting is performed.

NRLADU and NIRADU are INTEGER scalars that give the number of REAL and INTEGER words required respectively to hold the matrix factors if no numerical pivoting is performed.

NCMPA is an INTEGER scalar that holds the number of compresses of the internal data structure performed by MA57_ANALYSE.

ORDERING is an INTEGER scalar that records the actual ordering used by MA57_ANALYSE using the same numbering as for CONTROL%ORDERING.

2.2.4 Derived data type for information from MA57_FACTORIZE

The module contains a derived type called MA57_FINFO with the following components that are initialized to their default values when the object comes into existence

FLAG is an INTEGER scalar. The value of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.2.

MORE is an INTEGER scalar that provides further information in the case of an error, see Section 2.4.2.

STAT is an INTEGER scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

MAXFRT is an INTEGER scalar that holds the largest front size.

OPSA is a REAL scalar which is set to the number of floating-point additions performed during assembly.

OPSE is a REAL scalar which is set to the number of floating-point operations performed during factorization.

OPSB is a REAL scalar which is set to the number of additional floating-point operations performed during factorization because of use of the BLAS.

NRLTOT and NIRTOT are INTEGER scalars that give the total amount of REAL and INTEGER words respectively required for a successful factorization without the need for data compression, provided the same pivots are used.

NRLNEC and NIRNEC are INTEGER scalars that give the amount of REAL and INTEGER words required respectively for successful factorization allowing data compression, provided the same pivots are used.

NEBDU is an INTEGER scalar that gives the total number of entries in the factorization.

NRLBDU and NIRBDU are INTEGER scalars that give the amount of REAL and INTEGER words used respectively to hold the factorization.

NCMPBR and NCMPBI are INTEGER scalars that hold the number of compresses of the real and integer data structure respectively required by the factorization.

NTWO is an INTEGER scalar that holds the number of 2×2 pivots used during the factorization.

NEIG is an INTEGER scalar that holds the number of negative eigenvalues of **A**.

RANK is an INTEGER scalar that holds the rank of the original factorization. Note that, if static pivoting is used (see CONTROL%STATIC_TOLERANCE in Section 2.2.2), then the rank of the factorized matrix will always be *n*.

DELAY is an INTEGER scalar that holds the number of pivots passed up the tree because of numerical pivoting

considerations.

SIGNC is an INTEGER scalar that holds the number of sign changes of pivot when MA57_CONTROL%PIVOTING is set to 3.

MODSTEP is an INTEGER scalar that holds the pivot step at which matrix modification is first performed when MA57_CONTROL%PIVOTING is set to 4. If no matrix modification is performed, MODSTEP is set to 0.

MAXCHANGE is a REAL scalar that is set to the value of the largest change made to a pivot when MA57_AINFO%MODSTEP is positive.

SMAX is a REAL scalar that is set to the value of the largest scaling factor.

SMIN is a REAL scalar that is set to the value of the smallest scaling factor.

STATIC is an INTEGER scalar that is set to 1 if static pivots have been chosen and is otherwise set to 0.

2.2.5 Derived data type for information from MA57_SOLVE

The module contains a derived type called MA57_SINFO with the following components initialized to their default values when the object comes into existence. MA57_SINFO is also used by the subroutines in Sections 2.5.4 to 2.5.7.

FLAG is an INTEGER scalar. The value of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.4.3.

COND and COND2 are REAL scalars that hold the condition number of the matrix if the optional parameter COND was included in the call to MA57_SOLVE (see Sections 2.3.4 and 2.5.8).

BERR and BERR2 are REAL scalars that hold the backward error (scaled residual) for the matrix if the optional parameter COND was included in the call to MA57_SOLVE (see Sections 2.3.4 and 2.5.8).

ERROR is a REAL scalar that holds an estimate of the error in the solution if the optional parameter COND was included in the call to MA57_SOLVE (see Sections 2.3.4 and 2.5.8).

STAT is an INTEGER scalar. In the case of the failure of an allocate or deallocate statement, it is set to the STAT value.

2.2.6 Derived data type for the factors of a matrix

The module contains a derived type called MA57_FACTORS with private components.

2.3 Argument lists

We use square brackets [] to indicate optional arguments.

2.3.1 The initialization subroutine

The initialization subroutine must be called for each structure used to hold the factors. It may also be called for a structure used to control the subroutines. Each argument is optional. A call with no arguments has no effect.

```
CALL MA57_INITIALIZE([FACTORS][,CONTROL])
```

FACTORS is optional, scalar, of INTENT(OUT), and of type MA57_FACTORS. On exit, its allocatable array components will be deallocated.

CONTROL is optional, scalar, of INTENT(OUT), and of type MA57_CONTROL. On exit, its components will have been given the default values specified in Section 2.2.2.

2.3.2 To analyse the sparsity pattern

```
CALL MA57_ANALYSE(MATRIX,FACTORS,CONTROL,AINFO[,PERM])
```

MATRIX is scalar, of INTENT(IN), and of type ZD11_TYPE. The user must set the components N, NE, ROW, and COL, and they are not altered by the subroutine. **Restrictions:** MATRIX%N $\geq$ 1 and MATRIX%NE $\geq$ 0.

FACTORS is scalar, of INTENT(INOUT), and of type MA57_FACTORS. It must have been initialized by a call to MA57_INITIALIZE or have been used for a previous calculation. In the latter case, the previous data will be lost but the allocatable arrays will not be reallocated unless they are found to be too small.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

AINFO is scalar, of INTENT(OUT), and of type MA57_AINFO. Its components provide information about the execution, as explained in Section 2.2.3.

PERM is an array of size (n), optional, of INTENT(IN), and of type INTEGER. If present, PERM(i), i = 1, 2, ..., n, should be set to the position of variable i in the pivotal sequence.

2.3.3 To perform a factorization

```
CALL MA57_FACTORIZE(MATRIX,FACTORS,CONTROL,FINFO)
```

MATRIX is scalar, of INTENT(IN), and of type ZD11_TYPE. The components N and NE must be unaltered since the call to MA57_ANALYSE. The user must set the component VAL to hold the real values of the entries. None of the components are altered by the subroutine.

FACTORS is scalar, of INTENT(INOUT), and of type MA57_FACTORS. It must be unaltered since the call to MA57_ANALYSE or a subsequent call to MA57_FACTORIZE.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

FINFO is scalar, of INTENT(OUT), and of type MA57_FINFO. Its components provide information about the execution, as explained in Section 2.2.4.

2.3.4 To solve a set of equations

```
CALL MA57_SOLVE(MATRIX,FACTORS,X,CONTROL,SINFO[,RHS,ITER,COND])
```

MATRIX is scalar, of INTENT(IN), and of type ZD11_TYPE. It must be unaltered since the call to MA57_FACTORIZE and is not altered by the subroutine.

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE and is not altered by the subroutine.

X is an array with 1 or 2 dimensions, of INTENT(INOUT), and of type REAL. If RHS is absent, X must be set by the user to the vector **b** or the matrix **B** and on return it holds the solution **x** or **X**. If RHS is present, X must be set by the user to an approximate solution and on return it holds an improved solution, obtained by one cycle of iterative refinement without any use of arithmetic with additional precision.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

SINFO is scalar, of INTENT(OUT), and of type MA57_SINFO. Its components provide information about the execution, as explained in Section 2.2.5.

RHS is an array of the same shape as X, optional, of INTENT(IN), and of type REAL. If present, it must be set by the user to the vector **b** or the matrix **B**.

ITER is a scalar, optional, of INTENT(IN), and of type INTEGER. It can only be present if RHS is. If present, iterative refinement is performed (see Section 2.5.8).

COND is a scalar, optional, of INTENT(IN), and of type INTEGER. It can only be present if ITER is. If present, the condition number of the matrix is computed and is returned in SINFO%COND and SINFO%COND2, the backward error in SINFO%BERR and SINFO%BERR2, and an estimate of the error in SINFO%ERROR (see Section 2.5.8).

2.3.5 The finalization subroutine

```
CALL MA57_FINALIZE(FACTORS,CONTROL,INFO)
```

FACTORS is scalar, of INTENT(INOUT), and of type MA57_FACTORS. On exit, its allocatable array components will be deallocated. Without such finalization, the storage occupied is unavailable for other purposes.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

INFO is scalar, of INTENT(OUT), and of type INTEGER. On return, the value 0 indicates success. Any other value is the STAT value of a DEALLOCATE statement that has failed.

2.4 Error diagnostics

2.4.1 When analysing the sparsity pattern

A successful return from MA57_ANALYSE is indicated by AINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. Possible negative values are:

-1 Value of MATRIX%N out of range. MATRIX%N < 1. AINFO%MORE is set to value of MATRIX%N.

-2 Value of MATRIX%NE out of range. MATRIX%NE < 0. AINFO%MORE is set to value of MATRIX%NE.

-3 Failure of an allocate or deallocate statement. AINFO%STAT is set to the STAT value.

-9 The array PERM does not hold a permutation. AINFO%MORE holds first component at which an error was detected.

-10 The ordering from MeTiS was requested but the package was not available.

A positive flag value is associated with a warning message which will be output on unit AINFO%MP. Possible positive values are:

+1 Index (in MATRIX%ROW or MATRIX%COL) out of range. Action taken by subroutine is to ignore any such entries and continue. AINFO%OOR is set to the number of such entries. Details of the first ten are printed on unit CONTROL%MP.

+2 Duplicate indices. Action taken by subroutine is to keep the duplicates and then to sum corresponding reals when MA57_FACTORIZE is called. AINFO%DUP is set to the number of faulty entries. Details of the first ten are printed on unit CONTROL%MP.

+3 Both out-of-range indices and duplicates exist.

2.4.2 When factorizing the matrix

A successful return from MA57_FACTORIZE is indicated by FINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. In this case, no factorization will have been calculated. Possible negative values are:

-1 Value of MATRIX%N differs from the MA57_ANALYSE value. FINFO%MORE holds value of MATRIX%N.

-2 Value of MATRIX%NE out of range. MATRIX%NE < 0. FINFO%MORE holds value of MATRIX%NE.

-3 Failure of an allocate or deallocate statement. FINFO%STAT is set to the STAT value.

-5 Zero pivot detected (CONTROL%PIVOTING has the value 2 or 3). FINFO%MORE is set to the pivot step at which this was detected.

-6 A change of sign of pivots has been detected (CONTROL%PIVOTING has the value 2). FINFO%MORE is set to the pivot step at which this was detected.

-7 The real array that holds data for the factors needs to be bigger than CONTROL%MAXLA.

-8 The integer array that holds data for the factors needs to be bigger than CONTROL%MAXLIW.

A positive flag value is associated with a warning message which will be output on unit CONTROL%MP. In this case, a factorization will have been calculated.

+4 Matrix is rank deficient. In this case, FINFO%RANK will be set to the rank of the original factorization, but the factorization is altered by changing all the zero pivots to one. This will enable the subsequent solution of consistent sets of equations.

+5 Pivots have different signs when CONTROL%PIVOTING has the value 3. FINFO%NEIG is set to the number of negative eigenvalues. Details of the first ten are printed on unit CONTROL%MP. FINFO%MORE is set to the number of sign changes.

2.4.3 When solving the system

A successful return from MA57_SOLVE is indicated by SINFO%FLAG having the value zero. A negative value is associated with an error message which will be output on unit CONTROL%LP. In this case, no solution will have been calculated. Possible negative values are:

-3 Failure of an allocate or deallocate statement. SINFO%STAT is set to the STAT value.

-11 Iterative refinement has failed to converge.

A positive flag value can only be obtained when calling MA57_FREDHOLM_ALTERNATIVE

+1 System is singular and inconsistent. A Fredholm alternative vector is returned as well as a minimum norm solution.

2.5 Further features

In this section, we describe features for enquiring about and manipulating the parts of the factorization constructed. These features will not be needed by a user who wants simply to solve systems of equations with matrix **A**.

The algorithm produces an $\mathbf{L}\,\mathbf{D}\,\mathbf{L}^T$ factorization of a permutation of a scaled version of **A**, where **L** is a unit lower triangular matrix and **D** is a block diagonal matrix with blocks of order 1 and 2. It is convenient to write this factorization in the form

$$\mathbf{PSASP}^T = \mathbf{LDL}^T,$$

where **P** is a permutation matrix and **S** the diagonal scaling matrix. The following subroutines are provided:

1. MA57_ENQUIRE returns **P**, **D**, or **S**.
2. MA57_ALTER_D alters **D**. Note that this means that we no longer have a factorization of the given matrix **A**.
3. MA57_PART_SOLVE solves one of the systems of equations $\mathbf{LSx} = \mathbf{PSb}$, $\mathbf{S}^{-1}\mathbf{DS}^{-1}\mathbf{x} = \mathbf{b}$, or $\mathbf{SL}^T\mathbf{P}^T\mathbf{S}^{-1}\mathbf{x} = \mathbf{b}$, for one or more right-hand sides (**b** or **B**, respectively).
4. MA57_SPARSE_LSOLVE performs the forward solution using $\mathbf{LD}^{1/2}$ but exploits sparsity in the right-hand side.
5. MA57_FREDHOLM_ALTERNATIVE solves **Ax**=**b**. If the system is inconsistent, it also returns a vector that is in the null-space of **A** and has a non-zero scalar product with the right-hand side **b**.
6. MA57_LMULTIPLY computes the product of **L** or $\mathbf{L}^T$ with a vector.
7. MA57_GET_FACTORS returns the factors **L** and **D** for the matrix in the standard Compressed Sparse Column (CSC) format. and the permutation **P** such that $\mathbf{PAP}^T$=$\mathbf{LDL}^T$. It also returns the scaling factors.

2.5.1 To return information on the analysis and factorization

```
CALL MA57_ENQUIRE(FACTORS[,PERM,PIVOTS,D,PERTURBATION,SCALING])
```

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

PERM is an array of size (n), optional, of INTENT(OUT), and of type INTEGER. If present, PERM will be set to the pivot permutation selected by MA57_ANALYSE.

PIVOTS is an array of size (n), optional, of INTENT(OUT), and of type INTEGER. If present, the index of pivot i will be placed in PIVOTS(i), i = 1, 2, ..., n, with its sign negative if it is the index of a 2 x 2 block.

D is an array of size (2, n), optional, of INTENT(OUT), and of type REAL. If present, the diagonal entries of $\mathbf{D}^{-1}$ will be placed in D(1,i), i = 1, 2, ..., n and the off-diagonal entries of $\mathbf{D}^{-1}$ will be placed in D(2,i), i = 1, 2, ..., n–1.

PERTURBATION is an array of size (n), optional, of INTENT(OUT), and of type REAL. If present and MA57_CONTROL%PIVOTING is set to 4, PERTURBATION will be set to the perturbation to the diagonal of the matrix (see Section 2.2.2). If present and MA57_CONTROL%PIVOTING is not set to 4, PERTURBATION will be set to zero.

SCALING is an array of size (n), optional, of INTENT(OUT), and of type REAL. If present and MA57_CONTROL%SCALING is set to 1, SCALING will be set to the values of the diagonal scaling matrix **S**. If present and MA57_CONTROL%SCALING is not set to 1, SCALING will be set to the vector with all entries equal to one.

2.5.2 To alter D

```
CALL MA57_ALTER_D(FACTORS,D,INFO)
```

FACTORS is scalar, of INTENT(INOUT), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

D is an array of size (2, n), of INTENT(IN), and of type REAL. The diagonal entries of $\mathbf{D}^{-1}$ will be altered to D(1,i), i = 1, 2, ..., n and the off-diagonal entries of $\mathbf{D}^{-1}$ will be altered to D(2,i), i = 1, 2, ..., n–1.

INFO is scalar, of INTENT(OUT), and of type INTEGER. On return, the value 0 indicates success and the value $i>0$ indicates that D(2, i) is nonzero, but is not part of a block of order 2 of **D**.

2.5.3 To perform a partial solution

```
CALL MA57_PART_SOLVE(FACTORS,CONTROL,PART,X,INFO)
```

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

PART is scalar, of INTENT(IN), and of type CHARACTER. It must have one of the values

L for solving $\mathbf{LSx} = \mathbf{PSb}$ or $\mathbf{LSX} = \mathbf{PSB}$

D for solving $\mathbf{S}^{-1}\mathbf{DS}^{-1}\mathbf{x} = \mathbf{b}$ or $\mathbf{S}^{-1}\mathbf{DS}^{-1}\mathbf{X} = \mathbf{B}$, or

U for solving $\mathbf{SL}^T\mathbf{P}^T\mathbf{S}^{-1}\mathbf{x} = \mathbf{b}$ or $\mathbf{SL}^T\mathbf{P}^T\mathbf{S}^{-1}\mathbf{X} = \mathbf{B}$.

X is an array with 1 or 2 dimensions, of INTENT(INOUT), and of type REAL. It must be set by the user to the vector **b** or the matrix **B** and on return it holds the solution **x** or **X**.

INFO is scalar, of INTENT(OUT), and of type INTEGER. On return, the value 0 indicates success. Any other value is the STAT value of an ALLOCATE or DEALLOCATE statement that has failed.

2.5.4 To perform forward elimination exploiting sparsity in right-hand side

```
CALL MA57_SPARSE_LSOLVE(FACTORS,CONTROL,NZRHS,IRHS,NZSOLN,ISOLN,X,SINFO)
```

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to

MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

NZRHS is scalar, of INTENT(IN), and of type INTEGER. It must be set by the user to the number of nonzero entries in the right-hand side.

IRHS is an array of size NZRHS, of INTENT(IN), and of type INTEGER. It must be set by the user to a list of the components of the right-hand side that are nonzero.

NZSOLN is scalar, of INTENT(OUT), and of type INTEGER. It will be set by the routine to the number of nonzero entries in the solution.

ISOLN is an array, of INTENT(OUT), and of type INTEGER. The entries ISOLN(i), i = 1, NZSOLN will be set by the routine to a list of the components of the solution that are nonzero.

X is an array of size (n), of INTENT(INOUT), and of type REAL. It must be set by the user to the vector **b** and on return it holds the solution **x**.

SINFO is scalar, of INTENT(OUT), and of type MA57_SINFO. If SINFO%FLAG is equal to 0, then the execution was successful; if it is equal to –3, there was an error in an ALLOCATE or DEALLOCATE call and the STAT value is returned in SINFO%STAT.

2.5.5 To solve an indefinite potentially singular set of equations.

This subroutine computes the same solution of $\mathbf{Ax}=\mathbf{b}$ as MA57_SOLVE but, if the system is deemed to be inconsistent (see parameter CONTROL%CONSIST), then it will also return a Fredholm alternative vector **y** satisfying $\mathbf{Ay}=0$ and $\mathbf{y}^T\mathbf{b}\neq 0$.

```
CALL MA57_FREDHOLM_ALTERNATIVE(FACTORS,CONTROL,X,FREDX,SINFO)
```

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2. For this call, it is advised to set CONTROL%TOLERANCE to 10^{-12} in the prior call to MA57_FACTORIZE. CONTROL%CONSIST should be set to the value of the residual at which the equations may be considered consistent.

X is an array of size (n), of INTENT(INOUT), and of type REAL. It must be set by the user to the vector **b** and on return it holds a solution **x** of the set of equations. When the system is inconsistent this will be the minimizer of $\|\mathbf{Ax}-\mathbf{b}\|_{(\mathbf{LL}^T)^{-1}}$ of minimum norm.

FREDX is an array of size (n), of INTENT(OUT), and of type REAL. It need not be set by the user. If the system is inconsistent, on exit it will hold the vector **y** from the Fredholm alternative. If the system is consistent, it will not be accessed by the subroutine.

SINFO is scalar, of INTENT(OUT), and of type MA57_SINFO. If SINFO%FLAG is equal to 0, then the execution was successful; if it is equal to –3, there was an error in an ALLOCATE or DEALLOCATE call and the STAT value is returned in SINFO%STAT. If SINFO%FLAG is equal to 1 then the system is inconsistent and a Fredholm alternative vector, **y**, will be returned in FREDX.

2.5.6 To form a matrix-vector product using L

This subroutine computes $\mathbf{y}=\mathbf{S}^{-1}\mathbf{P}^T\mathbf{LSx}$ or $\mathbf{y}=\mathbf{SL}^T\mathbf{P}^T\mathbf{S}^{-1}\mathbf{x}$ for a given vector **x**.

```
CALL MA57_LMULTIPLY (FACTORS,CONTROL,TRANS,X,Y,SINFO)
```

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

TRANS is scalar of INTENT(IN), and of type CHARACTER. If it is set to 'N' (or 'n'), then the vector $\mathbf{S}^{-1}\mathbf{P}^T\mathbf{LSx}$ is returned in Y, otherwise $\mathbf{SL}^T\mathbf{P}^T\mathbf{S}^{-1}\mathbf{x}$ is returned in Y.

X is an array of size (n), of INTENT(IN), and of type REAL. It must be set by the user to the vector **x**.

Y is an array of size (n), of INTENT(OUT), and of type REAL. It will hold the product vector **y**.

SINFO is scalar, of INTENT(OUT), and of type MA57_SINFO. If SINFO%FLAG is equal to 0, then the execution was successful; if it is equal to –3, there was an error in an ALLOCATE or DEALLOCATE call and the STAT value is returned in SINFO%STAT.

2.5.7 To return the factors in a standard form

```
CALL MA57_GET_FACTORS (FACTORS,CONTROL,NZL,IPTRL,LROWS,LVALS,     &
                       NZD,IPTRD,DROWS,DVALS,PERM,INVPERM,SCALE,SINFO)
```

FACTORS is scalar, of INTENT(IN), and of type MA57_FACTORS. It must be unaltered since the call to MA57_FACTORIZE or a subsequent call to MA57_ALTER_D.

CONTROL is scalar, of INTENT(IN), and of type MA57_CONTROL. Its components control the action, as explained in Section 2.2.2.

NZL is scalar, of INTENT(OUT), and of type INTEGER. On return it will hold the number of entries in the factor **L**. This will be equal to FINFO%NEBDU on exit from MA57_FACTORIZE.

IPTRL is an array of size (n+1), of INTENT(OUT), and of type INTEGER. IPTRL(i) will be set to the position in LROWS and LVALS of the first entry in column i of **L**.

LROWS is an array of size (NZL), of INTENT(OUT), and of type INTEGER. It will be set to row indices in the columns of **L**.

LVALS is an array of size (NZL), of INTENT(OUT), and of type REAL. It will be set to the values of entries in the columns of **L**. Thus, entries in column j of **L** are in rows LROWS(k),k=IPTRL(j),IPTRL(j+1)–1 and have values LVALS(k),k=IPTRL(j),IPTRL(j+1)–1.

NZD is scalar, of INTENT(OUT), and of type INTEGER. On return it will hold the number of entries in the factor **D**. This will be equal to 2*FINFO%NTWO + N on exit from MA57_FACTORIZE. This value is always less than or equal to 2*N.

IPTRD is an array of size (n+1), of INTENT(OUT), and of type INTEGER. IPTRD(i) will be set to the position in DROWS and DVALS of the first entry in column i of **D**.

DROWS is an array of size (NZD), of INTENT(OUT), and of type INTEGER. It will be set to row indices in the columns of **D**.

DVALS is an array of size (NZD), of INTENT(OUT), and of type REAL. It will be set to the values of entries in the columns of **D**. Thus, entries in column j of **D** are in rows DROWS(k),k=IPTRD(j),IPTRD(j+1)–1 and have values DVALS(k),k=IPTRD(j),IPTRD(j+1)–1. Note that the whole 2 by 2 pivot is held. Zeros are held explicitly in the case of singular matrices.

PERM is an array of size (n), INTENT(OUT), and of type INTEGER. PERM will be set to the pivot permutation selected by MA57_FACTORIZE.

INVPERM is an array of size (n), INTENT(OUT), and of type INTEGER. INVPERM will be set to the inverse of PERM.

SCALE is an array of size (n), of INTENT(OUT), and of type REAL. It will be set to the values of the scaling factors.

SINFO is scalar, of INTENT(OUT), and of type MA57_SINFO. If SINFO%FLAG is equal to 0, then the execution was successful; if it is equal to –3, there was an error in an ALLOCATE or DEALLOCATE call and the STAT value is

returned in SINFO%STAT.

2.5.8 Iterative refinement, condition number, and error estimates

If the parameter ITER is included in a call to MA57_SOLVE, then iterative refinement is invoked. If COND is also included, information is returned on the condition number and errors. We use the theory developed by Arioli, Demmel, and Duff (1989). We use the notation $\bar{\mathbf{x}}$ for the computed solution and a modulus sign on a vector or matrix to indicate the vector or matrix obtained by replacing all entries by their moduli. We define two scaled residuals:

$$\omega_1 = \max_i \left(\frac{|\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{b}|+|\mathbf{A}||\bar{\mathbf{x}}|)_i} \right)$$

where the max is over all equations except those for which the numerator is nonzero and the denominator is small. For the exceptional equations, we define

$$\omega_2 = \max_i \left(\frac{|\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{A}||\bar{\mathbf{x}}|)_i + \|\mathbf{A}_i\|_\infty \|\bar{\mathbf{x}}\|_\infty} \right).$$

ω_1 and ω_2 are returned in SINFO%BERR and SINFO%BERR2, respectively.

The computed solution $\bar{\mathbf{x}}$ is the exact solution of the equation

$$(\mathbf{A}+\delta\mathbf{A})\mathbf{x} = (\mathbf{b}+\delta\mathbf{b})$$

where $\delta\mathbf{A}_{ij} \le \max(\text{BERR}, \text{BERR2})\,|\mathbf{A}|_{ij}$ and $\delta\mathbf{b}_i \le \max(\text{BERR}\,|\mathbf{b}|_i, \text{BERR2}\,\|\mathbf{A}_i\|_\infty \|\bar{\mathbf{x}}\|_\infty)$. Note that $\delta\mathbf{A}$ respects the sparsity in **A**. An upper bound for the forward error is returned in ERROR that is computed as BERR*COND + BERR2*COND2 where COND and COND2 are condition numbers corresponding to κ_{ω_1} and κ_{ω_2}, respectively as defined in equation (15) of Arioli, Demmel, and Duff (1989).

Reference

Arioli, M. Demmel, J. W., and Duff, I. S. (1989). Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. and Applics.* **10**, 165-190.

3 GENERAL INFORMATION

Use of common: None

Workspace: Provided automatically by the module.

Other routines called directly: MA57A/AD, MA57B/BD, MA57C/CD, MA57D/DD, MA57E/ED, MA57I/ID, MC71A/AD.

Other modules used directly: HSL_ZD11_single/double.

Input/output: Error, warning and diagnostic messages only. Error messages on unit CONTROL%LP and warning and diagnostic messages on unit CONTROL%WP and CONTROL%MP, respectively. These have default value 6, and printing of these messages is suppressed if the relevant unit number is set negative. These messages are also suppressed if MA57_CONTROL%LDIAG is less than 1.

Restrictions:

MATRIX%N $\ge$ 1,
MATRIX%NE $\ge$ 0.

MeTiS

The MA57 package uses the MeTiS graph partitioning library available from the University of Minnesota website. If MeTiS is not available then the user must compile with the supplied replacement subroutine METIS_NodeND.

Changes between Version 1.0.0 and Version 2.0.0:

Version 2.0.0 incorporates several additional features to those of Version 1.0.0. We give information on these in this section and indicate where default values of control parameters have changed from the earlier version.

Built-in scaling is now available through a symmetrized version of MC64. This is controlled by the new control parameter CONTROL%SCALING whose default is set so that scaling is performed. The matrix is explicitly scaled internally to the package as are the right-hand side and the solution so that the user need not be concerned with this. Iterative refinement, if requested, is based on the original unscaled matrix. The minimum and maximum scale factors are returned in FINFO%SMIN and FINFO%SMAX, respectively.

The user can now use an ordering from the MeTiS package. This option is invoked by setting CONTROL%ORDERING equal to 4. Since the user must load the MeTiS library separately from HSL, a dummy routine has been included that will return the error FINFO%FLAG=-10 if the MeTiS library is not available.

Static pivoting is now used so that the factorization can be performed using the same storage as predicted by the analysis even if the matrix is not positive definite. This is invoked by setting CONTROL%STATIC_TOLERANCE to a nonzero value. A certain amount of of additional storage can be allowed by setting CONTROL%STATIC_LEVEL to a nonzero value.

Further testing on large problems has determined that a better value for CONTROL%NEMIN is 16 so that this value has now been set as the default for this parameter.

Changes between Version 2.0.0 and Version 3.0.0:

An additional option has been added to the choice of ordering strategies. If CONTROL%ORDERING is set to 5 (now the default), then MA57_ANALYSE will choose automatically between using MeTiS or AMD (avoiding problems with dense rows); first based on matrix order and density and then, if necessary, by running both ordering strategies. The actual ordering used is returned in AINFO%ORDERING.

Changes between Version 3.0.0 and Version 4.0.0:

Pointer arrays have been replaced by allocatables in derived data types. An added control parameter RANK_DEFICIENT allows the option of dropping blocks of small entries during the factorization so that the factorization will be much more efficient if the matrix is highly rank deficient.

Changes between Version 4.0.0 and Version 5.0.0:

Several new facilities were introduced in Version 5.0.0. These are described in Sections 2.5.4 to 2.5.7.

1. Triangular solve exploiting sparsity in right-hand side.
2. Computation of Fredholm alternative vector when system is inconsistent.
3. Matrix-vector product using factor **L**.
4. Return of factors in standard compressed sparse column (CSC) format.

4 METHOD

A version of sparse Gaussian elimination is used.

The MA57_ANALYSE entry (with CONTROL%ORDERING≠1) chooses a pivot ordering based on either nested dissection or minimum degree using a generalized element model of the elimination to avoid storing the filled-in pattern explicitly. The elimination is represented as an assembly tree with the order of elimination determined by a depth-first search of the tree.

The MA57_FACTORIZE entry factorizes the matrix by using the assembly and elimination ordering generated by MA57_ANALYSE. By default, the input matrix is first scaled using a symmetrized version of the HSL code MC64. At each stage in the multifrontal approach, pivoting and elimination are performed on full submatrices and, when diagonal 1×1 pivots would be numerically unstable, 2×2 diagonal blocks are used. The operations on the full

submatrices are performed using the Level 3 BLAS. MA57_FACTORIZE can thus be used to factor indefinite systems and will perform well on machines with caches or levels of memory hierarchy.

If CONTROL%STATIC_TOLERANCE is set to a value greater than zero, static pivoting is invoked. This means that if, at any stage of the multifrontal elimination, there are fully summed rows and columns from which it is not possible to choose pivots because of the threshold criterion defined by CONTROL%U, then we first try to get pivots using a weaker threshold by successively trying values one tenth of the previous until a threshold of √(CONTROL%U*CONTROL%STATIC_TOLERANCE) is reached. If there are still fully summed rows and columns left, then the diagonal entries are replaced by CONTROL%STATIC_TOLERANCE times the largest modulus of an entry in the scaled matrix and are used as 1×1 pivots in the factorization. If, furthermore, CONTROL%STATIC_LEVEL is greater than zero, then CONTROL%STATIC_TOLERANCE is treated as 0 (uneliminated variables are delayed) until CONTROL%STATIC_LEVEL*N fully-summed variables have been delayed.

The MA57_SOLVE entry uses the factors from MA57_FACTORIZE to solve systems of equations either by loading the appropriate parts of the vectors into an array of the current front-size and using full matrix code employing the Level 2 and Level 3 BLAS or by indirect addressing at each stage, depending on the value of CONTROL%SOLVEBLOCKING and the size of the frontal matrix. If static pivots were chosen, then iterative refinement is automatically performed.

A fuller account of this method is given by Duff and Reid (AERE-R.10533, 1982) and Duff and Reid, ACM Trans. Math. Software **9** (1983), 302-325. A description of Version 1.0.0 of the HSL_MA57 package is given by Duff, ACM Trans. Math. Software **30** (2004), 118-144. Some of the features that are new to Version 2.0.0 are described in "Strategies for scaling and pivoting for sparse symmetric indefinite problems" by Duff and Pralet (SIAM Journal Matrix Analysis and Applications **27** (2005), 313-340). A fuller discussion of our static pivoting strategy is given in the report RAL-TR-2005-007 "Towards a stable static pivoting strategy for the sequential and parallel solution of sparse symmetric indefinite systems" by Duff and Pralet. This report has been accepted for publication in the SIAM Journal on Matrix Analysis and Applications and can be obtained from the web site

http://www.numerical.rl.ac.uk/reports/reports.html

5 EXAMPLE OF USE

We illustrate the use of the package on the solution of the single set of equations

$$\begin{pmatrix} 2 & 3 & & & \\ 3 & 0 & 4 & & 6 \\ & 4 & 1 & 5 & \\ & & 5 & 0 & \\ & 6 & & & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 8 \\ 45 \\ 31 \\ 15 \\ 17 \end{pmatrix}$$

Note that this example does not illustrate all the facilities.

Program

```
! Simple example of use of HSL_MA57
program main
   use hsl_ma57_double
   implicit none
   type(zd11_type) matrix
   type(ma57_control) control
   type(ma57_ainfo) ainfo
   type(ma57_finfo) finfo
   type(ma57_sinfo) sinfo
   type(ma57_factors) factors

   integer, parameter :: wp = kind(0.0d0)

   real (wp), allocatable :: b(:),x(:)
   integer i,info,n,ne
```

```
! Read matrix order and number of entries
      read (5,*) n,ne
      matrix%n = n
      matrix%ne = ne

! Allocate arrays of appropriate sizes
      allocate(matrix%val(ne), matrix%row(ne), matrix%col(ne))
      allocate(b(n), x(n))

! Read matrix and right-hand side
      read (5,*) (matrix%row(i),matrix%col(i),matrix%val(i),i=1,ne)
      read (5,*) b

! Initialize the structures
      call ma57_initialize(factors,control)

! Analyse
      call ma57_analyse(matrix,factors,control,ainfo)
      if (ainfo%flag<0) then
         write(6,'(a,i2)') &
            ' Failure of ma57_analyse with ainfo%flag=', ainfo%flag
         stop
      end if

! Factorize
      call ma57_factorize(matrix,factors,control,finfo)
      if (finfo%flag<0) then
         write(6,'(a,i2)') &
            ' Failure of ma57_factorize with finfo%flag=', finfo%flag
         stop
      end if

! Solve without refinement
      x = b
      call ma57_solve(matrix,factors,x,control,sinfo)
      if (sinfo%flag==0) write(6,'(a,/,(3f20.16))')  &
         ' Solution without refinement is',X

! Perform one refinement
      call ma57_solve(matrix,factors,x,control,sinfo,b)
      if (sinfo%flag==0) write(6,'(a,/,(3f20.16))') &
          ' Solution after one refinement is',X

! Clean up
      deallocate(matrix%val, matrix%row, matrix%col, b, x)
      call ma57_finalize(factors,control,info)

end program main
```

Data

```
 5 7
 1 1 2.0
 1 2 3.0
 2 3 4.0
 2 5 6.0
 3 3 1.0
 3 4 5.0
```

```
5 5 1.0
8. 45. 31. 15. 17.
```

Output

```
Solution without refinement is
 0.9999999999999997  2.0000000000000000  2.9999999999999996
 4.0000000000000000  5.0000000000000027
Solution after one refinement is
 1.0000000000000000  2.0000000000000000  3.0000000000000000
 4.0000000000000000  5.0000000000000000
```