1 SUMMARY

The module HSL_MA42 **solves one or more sets of sparse linear equations, $\mathbf{Ax}=\mathbf{b}$ or $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$, by the frontal method,** optionally using direct access files for the matrix factors. Use is made of high level BLAS kernels. The code has low in-core memory requirements. The matrix **A** may be input by the user in either of the following ways:

(i) by elements in a finite-element calculation,

(ii) by equations (matrix rows).

In both cases, the coefficient matrix and right-hand side(s) are of the form

$$\mathbf{A}=\sum_{k=1}^{m}\mathbf{A}^{(k)}, \quad \mathbf{b}=\sum_{k=1}^{m}\mathbf{b}^{(k)} .$$

In case (i), the summation is over finite elements. The matrix $\mathbf{A}^{(k)}$ is nonzero only in those rows and columns which correspond to variables in the k-th element. The vector $\mathbf{b}^{(k)}$ is nonzero only in those rows which correspond to variables in element k.

In case (ii), the summation is over equations and $\mathbf{A}^{(k)}$ and $\mathbf{b}^{(k)}$ are nonzero only in row k.

In both cases, for each k, the user must supply a list specifying which columns of **A** are associated with $\mathbf{A}^{(k)}$, and arrays containing $\mathbf{A}^{(k)}$ and $\mathbf{b}^{(k)}$ in packed form. The nature of this packed form is defined more precisely in section 2.2 (arguments VARS, REALS, and RHS).

The frontal method is a variant of Gaussian elimination and involves the factorization

$$\mathbf{A}=\mathbf{PLUQ},$$

where **P** and **Q** are permutation matrices, **L** is a lower triangular matrix, and **U** is an upper triangular matrix. In HSL_MA42, the row and column indices of the variables in **UQ** and **PL** are stored separately from the values of the entries in the factors **UQ** and **PL**. At an intermediate stage of the factorization, l say, the 'front' normally contains those variables associated with one or more of the matrices $\mathbf{A}^{(k)}$, $k=1,2,...,l$, which are also present in one or more of $\mathbf{A}^{(k)}$, $k=l+1,...,m$, although for stability reasons it may include a few more variables from $\mathbf{A}^{(k)}$, $k=1,2,...,l$. The frontal matrix is held in-core but a principal feature of HSL_MA42 is that it offers the user the option of holding the factors in direct access files. During the factorization, data is put into in-core buffers (workspace arrays) then, once a buffer is full, it is written to a direct access file. This use of direct access files allows large problems to be solved in a predetermined and relatively small amount of in-core memory. Another feature of HSL_MA42 is that the user may reduce storage requirements further by choosing not to store the factor **PL**. The factor **PL** need only be stored if the user wishes either to solve subsequently for further right-hand sides or to solve systems $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$.

ATTRIBUTES — Version: 1.3.0. (10 April 2013) **Types:** Real (single, double). **Remark:** HSL_MA42 is a Fortran 90 version of MA42. **Helpful:** MC62, MC63. **Calls:** I_AMAX, _AXPY,_GER, _GEMV, _TPSV, _GEMM, _TRSM. **Language:** Fortran 90. **Original date:** August 1995. **Origin:** J.A. Scott, 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_MA42_SINGLE
```

Double precision version

```
USE HSL_MA42_DOUBLE
```

For each problem, the user must declare a structure DATA of a derived type MA42_DATA defined by the module. This structure holds all the data for the problem and must be passed as an argument on every call to the subroutines.

There are seven subroutines for user calls:

(a) The initialization subroutine MA42_INITIALIZE must first be called. This subroutine need only be called once prior to calling MA42_ANALYZE, MA42_SYMBOLIC, MA42_FILES, MA42_FACTORIZE, MA42_SOLVE, and MA42_FINAL.

(b) MA42_ANALYZE must be called for each element (or equation) to specify which variables are associated with it.

(c) The use of MA42_SYMBOLIC is optional. If the user wishes to perform a symbolic factorization, MA42_SYMBOLIC must be called for each element (or equation).

(d) If direct access files are to be used, MA42_FILES must be called once prior to calling MA42_FACTORIZE and MA42_SOLVE. Otherwise, MA42_FILES should not be called.

(e) MA42_FACTORIZE must be called for each element (or equation) to specify the entries of $\mathbf{A}^{(k)}$ and, optionally, $\mathbf{b}^{(k)}$. MA42_FACTORIZE uses the data from MA42_ANALYZE to factorize the matrix $\mathbf{A}=\sum_{k=1}^{m}\mathbf{A}^{(k)}$ and, if $\mathbf{b}^{(k)}$ are specified, MA42_FACTORIZE solves the equations $\mathbf{Ax}=\mathbf{b}$ with right-hand side(s) $\mathbf{b}=\sum_{k=1}^{m}\mathbf{b}^{(k)}$.

(f) The use of MA42_SOLVE is optional. MA42_SOLVE uses the factors produced by MA42_FACTORIZE to rapidly solve either further systems of the form $\mathbf{Ax}=\mathbf{b}$ or systems of the form $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$.

(g) MA42_FINAL should be called once after all other calls to subroutines in the module are complete. MA42_FINAL deallocates all arrays which have been allocated by the module.

2.2 Argument lists

In each call, optional arguments follow the argument DATA. Since we reserve the right to add additional optional arguments in future releases of the code, **we strongly recommend that all optional arguments be called by keyword, not by position.**

2.2.1 The derived data type

For each problem, the user is required to declare a structure of type MA42_DATA thus:

```
USE HSL_MA42_SINGLE
...
TYPE (MA42_DATA) DATA
...
```

or

```
USE HSL_MA42_DOUBLE
...
```

```
TYPE (MA42_DATA) DATA
...
```

2.2.2 The initialization subroutine

For each problem, the user must first call the initialization subroutine MA42_INITIALIZE.

```
CALL MA42_INITIALIZE(DATA[,EQUATION])
```

DATA is a scalar structure of type MA42_DATA. On exit, the components of DATA that control the action of the subroutines in the module HSL_MA42 contain default values. If the user wishes to use values other than the defaults, the corresponding components of DATA should be reset after the call to MA42_INITIALIZE. In particular, if the user is not going to call MA42_SOLVE and wants to save storage by not storing the factor **PL**, DATA%LFACTOR should be reset to .FALSE. Full details of the control components of DATA are given in section 2.3.

EQUATION is an optional scalar INTENT(IN) argument of type default LOGICAL. If input is by equations (case(ii)), EQUATION must be present and set to .TRUE. Otherwise, input is by elements (case(i)). **Default:** EQUATION = .FALSE.

2.2.3 Specification of which variables belong in each element or equation

A call of the following form must be made for each element (case (i)) or each equation (case (ii)):

```
CALL MA42_ANALYZE(VARS,DATA[,LENLAST])
```

VARS is a rank 1 array INTENT(IN) argument of type default INTEGER. VARS must contain the indices of the variables associated with the element (or equation). These indices need not be in increasing order but must be distinct. **Restrictions:** VARS(:) $\geq$ 1 and VARS(I) $\neq$ VARS(J), I $\neq$ J.

DATA is as in the call to MA42_INITIALIZE. On each exit, DATA%LARGEST_INDEX holds the largest integer so far used to index a variable. Note that if the variables are not numbered contiguously, DATA%LARGEST_INDEX will exceed the number of variables in the problem (see DATA%PROBLEM_SIZE in section 2.4).

LENLAST is an optional scalar INTENT(IN) argument of type default INTEGER. If present, LENLAST is only accessed on the first call to MA42_ANALYZE and must be set by the user to be at least as large as the largest integer used to index a variable. **Default:** LENLAST = 1000. **Restriction:** LENLAST $\geq$ 1.

2.2.4 Symbolic factorization of A

The in-core memory requirements and the lengths of the files to hold the factors are dependent upon the frontsize (see section 2.6). The user may compute lower bounds for the maximum frontsize and for the lengths of the files for **UQ** and **PL**, together with an estimate for the length of the file for the row and column indices, by performing a symbolic factorization. This is done by making a call of the following form to MA42_SYMBOLIC for each element (case (i)) or each equation (case (ii)). The elements (or equations) must have the same index lists and be in exactly the same order as when MA42_ANALYZE was called.

Note that all calls to MA42_ANALYZE must be completed before the first call to MA42_SYMBOLIC.

```
CALL MA42_SYMBOLIC(VARS,DATA)
```

VARS is as in the corresponding calls to MA42_ANALYZE but MA42_SYMBOLIC does not check VARS for duplicate indices.

DATA is as in the call to MA42_INITIALIZE. On exit from the final call, DATA%FRONT_BOUND(1:2) holds lower bounds on the maximum row and column frontsizes. For element entry, the row and column frontsizes are equal. In addition, DATA%FILE_BOUND(1:2) holds lower bounds on the lengths (in words) of the files for the factors **UQ** and **PL**, and DATA%FILE_BOUND(3) holds an estimate (usually an underestimate) of the length (in words) of the integer file for the row and column indices. Note that, because of the way singular matrices are

treated, the bounds may not be reliable if the matrix is singular. They may also be an over estimate if zeros in the front are exploited (DATA%EXPLOIT is set to .TRUE., see Section 2.3).

2.2.5 To setup direct access files

If the user wishes to use direct access files to store the matrix factors (which will keep in-core memory requirements small), a call of the following form must be made:

```
CALL MA42_FILES(LENBUF,LENFLE,DATA[,FILENAME])
```

LENBUF is a rank 1 array INTENT(IN) argument of type default INTEGER and dimension 3. LENBUF(1) must hold the length (in words) of the buffer (in-core workspace array) associated with the direct access file for **UQ** (including the corresponding right-hand sides), LENBUF(2) must hold the length (in words) of the buffer associated with the direct access file for **PL**, and LENBUF(3) must hold the length (in words) of the integer buffer associated with the direct access file for the row and column indices. LENBUF(2) is not accessed if the user has reset DATA%LFACTOR to .FALSE. (see section 2.3). Advice on setting LENBUF is given in section 2.6.1. **Restriction:** LENBUF(:) ≥ 1.

LENFLE is a rank 1 array INTENT(IN) argument of type default INTEGER and dimension 3. LENFLE(1) must hold the length (in words) of the direct access file for **UQ** (including the corresponding right-hand sides), LENFLE(2) must hold the length (in words) of the direct access file for **PL**, and LENFLE(3) must hold the length (in words) of the direct access file for the row and column indices. LENFLE(2) is not accessed if the user has reset DATA%LFACTOR to .FALSE.. Advice on setting LENFLE is given in section 2.6.1. **Restriction:** LENFLE(:) ≥ LENBUF(:).

DATA is as in the call to MA42_INITIALIZE. On exit, DATA%STRM(1:3) contains the unit numbers which will be used for the direct access files.

FILENAME is an optional rank 1 array INTENT(IN) argument of type default character length 50 and dimension 3. If present, FILENAME(I), I = 1, 2, 3, must be set by the user to the name of the direct access file for the **UQ** factor, the **PL** factor, and the row and column indices, respectively. FILENAME(2) is not accessed if the user has reset DATA%LFACTOR to .FALSE. If FILENAME is not present, unnamed direct access files are used and are not kept after the termination of the calling program.

2.2.6 To factorize A and optionally solve Ax = b

A call of the following form must be made for each element (case (i)) or each equation (case (ii)). The elements (or equations) must have the same index lists and be in exactly the same order as when MA42_ANALYZE was called.

Note that all the calls to MA42_ANALYZE (and to MA42_SYMBOLIC, if used) must be completed before the first call to MA42_FACTORIZE.

```
CALL MA42_FACTORIZE(VARS,REALS,DATA[,RHS,FRONTSIZE,LENBUF,X])
```

VARS is as in the corresponding calls to MA42_ANALYZE but MA42_FACTORIZE does not check VARS for duplicate indices.

REALS is a rank 2 array INTENT(INOUT) argument of type default REAL (or double precision REAL in the DOUBLE version) and dimensions *nvar* by *nvar* (element entry) or 1 by *nvar* (equation entry) where *nvar* = SIZE(VARS). REALS must contain $\mathbf{A}^{(k)}$ in packed form. That is, for element entry, REALS(I,J) must contain the contribution to entry VARS(I), VARS(J) in the matrix **A** from the current element (I, J = 1, 2,..., *nvar*). Contributions to the same entry from different elements are summed. For equation entry (EQUATION = .TRUE.), REALS(1,J) must contain the coefficient of variable VARS(J) in the current equation (J = 1, 2,..., *nvar*). The contents of this array are changed by the routine.

DATA is as in the call to MA42_INITIALIZE. On exit from the final call, some of the components of DATA contain information on the factorization. Details of the information contained in DATA are given in section 2.4.

RHS is an optional rank 2 array INTENT(INOUT) argument of type default REAL (or double precision REAL in the

DOUBLE version) and dimensions *nvar* by *nrhs* (element entry) or 1 by *nrhs* (equation entry) where *nvar* = SIZE(VARS) and *nrhs* is the number of right-hand sides. If present on the first call, RHS must be present on each call and must contain $\mathbf{b}^{(k)}$ in packed form. That is, for element entry, RHS(I,J) must contain the contribution to component VARS(I) of the J-th right-hand side from the current element (I = 1, 2,..., *nvar*, J = 1, 2,..., *nrhs*). Contributions to the same component from different elements are summed. For equation entry, RHS(1,J) must contain the contribution to the J-th right-hand side for the current equation (J = 1, 2,..., *nrhs*). The contents of this array are changed by the routine.

FRONTSIZE is an optional array INTENT(IN) argument of type default integer and dimension 2. If present, FRONTSIZE is only accessed on the first call to MA42_FACTORIZE and must be set by the user to hold the maximum row and column frontsizes allowed. For element entry (the default), FRONTSIZE(2) is not accessed (for element entry, the row and column frontsizes are equal). FRONTSIZE has a crucial effect on the amount of in-core memory required (see below). Advice on choosing FRONTSIZE is given in section 2.6.2. See also the error returns -12 and +4 in section 2.5. **Default:** FRONTSIZE(:) = 200. **Restriction:** FRONTSIZE(:) ≥ 1.

LENBUF is an optional array INTENT(IN) argument of type default integer and dimension 3. LENBUF is not accessed if MA42_FILES has been called. Otherwise, if present, LENBUF is accessed only on the first call to MA42_FACTORIZE and, in this case, LENBUF(1) must hold the length (in words) of the (in-core) file for **UQ** (including the corresponding right-hand sides), LENBUF(2) must hold the length (in words) of the file for **PL**, and LENBUF(3) must hold the length (in words) of the integer file for the row and column indices. LENBUF(2) is not accessed if the user has reset DATA%LFACTOR to .FALSE. (see section 2.3). Advice on setting LENBUF is given in section 2.6.1. **Default:** LENBUF = (/100000, 100000, 50000/). **Restriction:** LENBUF(:) ≥ 1.

X is an optional rank 2 array INTENT(OUT) argument of type default REAL (or double precision REAL in the DOUBLE version) and dimensions DATA%LARGEST_INDEX by *nrhs* where *nrhs* is the number of right-hand sides. X must be present on the last call to MA42_FACTORIZE if RHS is present; X is not accessed on the other calls to MA42_FACTORIZE. On exit from the final call, if I has been used to index a variable, X(I,J) holds the solution for variable I to system J and is set to zero otherwise.

2.2.7 To solve further systems $\mathbf{Ax}=\mathbf{b}$ or systems $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$

```
CALL MA42_SOLVE(B,X,DATA[,TRANS])
```

B is a rank 2 array INTENT(INOUT) argument of type default REAL (or double precision REAL in the DOUBLE version) and dimensions DATA%LARGEST_INDEX by *nrhs*, where *nrhs* is the number of right-hand sides. On entry, B must be set so that if I has been used to index a variable, B(I,J) is the corresponding component of the right-hand side for the J-th system (J=1,2,..., *nrhs*). The contents of this array are changed by the routine.

X is a rank 2 array INTENT(OUT) argument of type default REAL (or double precision REAL in the DOUBLE version) and the same dimensions as B. On exit, if I has been used to index a variable, X(I,J) holds the solution for variable I to system J and is set to zero otherwise.

DATA is as in the call to MA42_INITIALIZE.

TRANS is an optional scalar INTENT(IN) argument of type default LOGICAL. If present and TRANS = .TRUE., systems of the form $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$ are to be solved. Otherwise, systems of the form $\mathbf{Ax}=\mathbf{b}$ are to be solved. **Default:** TRANS = .FALSE..

2.2.8 The terminating subroutine

```
CALL MA42_FINAL(DATA)
```

DATA is as in the call to MA42_INITIALIZE.

2.3 Control parameters

The components of DATA that control the action of MA42_ANALYZE, MA42_SYMBOLIC, MA42_FILES , MA42_FACTORIZE, and MA42_SOLVE are as follows. Default values are set by MA42_INITIALIZE.

DATA%ACTION is a rank 1 of type default LOGICAL and dimension 2 and with default value .TRUE.. If DATA%ACTION(1) = .FALSE., MA42_ANALYZE terminates if LENLAST is too small (see DATA%ERROR = -5 in section 2.5). Otherwise, a warning is issued (see DATA%ERROR = +3 in section 2.5) and the computation continues. The smallest value which will suffice for LENLAST is returned in DATA%LARGEST_INDEX (see section 2.4). If DATA%ACTION(2) = .FALSE., MA42_FACTORIZE terminates if FRONTSIZE is too small (see DATA%ERROR = -12 and DATA%ERROR = +4). Otherwise, a warning is issued and the computation continues but may be less efficient than if FRONTSIZE was large enough. The smallest value which will suffice for FRONTSIZE is returned in DATA%MAX_FRONT (see section 2.4).

DATA%ALPHA is a scalar of type default REAL (or double precision REAL in the DOUBLE version) with default value 0.1. An element of the frontal matrix is normally only considered suitable for use as a pivot if it is of absolute value at least as large as DATA%ALPHA times the entry of largest absolute value in its column.

DATA%LFACTOR is a scalar of type default LOGICAL with default value .TRUE. In this case, the **PL** factor is stored. If the use does not wish to store the **PL** factor (it must be stored if the user wishes to call MA42_SOLVE either to solve for further right-hand sides or to solve transpose systems), DATA%LFACTOR should be set to .FALSE..

DATA%OUTPUT is a rank 1 array of type default INTEGER and dimension 2. DATA%OUTPUT(1) is the stream number for error messages (default 6) and DATA%OUTPUT(2) is the stream number for warning messages (default 6). Printing is suppressed if DATA%OUTPUT(:) $\leq$ 0.

DATA%PIVOT is a rank 1 array of type default INTEGER and dimension 3 which controls the search for pivots. DATA%PIVOT(1) has default value 0. If DATA%PIVOT(1) = 1 and entry is by elements, only diagonal pivots are chosen until the last element has been entered and no more diagonal pivots can be chosen. At this point, off-diagonal pivots are used. If DATA%PIVOT(1) $\neq$ 1, off-diagonal pivoting is allowed. If entry is by equations, off-diagonal pivots are used for all values of DATA%PIVOT(1). DATA%PIVOT(2) has default value 0. If DATA%PIVOT(2) is greater than 0, then, when the number of potential pivot columns is greater than or equal to DATA%PIVOT(2), an elimination will be performed even if the best pivot candidate does not satisfy the threshold criterion determined by DATA%ALPHA. DATA%PIVOT(3) has default value 0. If DATA%PIVOT(2:3) > 0, then, when the number potential pivot columns is greater than or equal to DATA%PIVOT(2), only DATA%PIVOT(3) of the potential pivot columns are searched for a pivot. The best pivot candidate (largest relative to the other nonzeros in its column) from these DATA%PIVOT(3) columns is then used as a pivot.

DATA%PIVOT_SIZE is a scalar of type default INTEGER with default value 1. DATA%PIVOT_SIZE is the minimum number of variables that are eliminated at each stage of the factorization. Increasing DATA%PIVOT_SIZE in general increases the number of floating-point operations and real storage requirements but allows greater advantage to be taken of Level 3 BLAS and reduces integer storage.

DATA%SINGULAR is a scalar of type default LOGICAL with default value .TRUE.. If the matrix is found to be singular during the decomposition and DATA%SINGULAR is equal to .TRUE., an error flag is set and the computation terminates (see DATA%ERROR = -14 in section 2.5). If DATA%SINGULAR = .FALSE., a warning is given, the computation continues and components of the solution vector X corresponding to zero pivots are set equal to zero (see also DATA%DEFICIENCY in section 2.4 and DATA%ERROR = +1 in section 2.5).

DATA%SMALL is a scalar of type default REAL (or double precision REAL in the DOUBLE version) with default value zero. The matrix is declared singular if, during the factorization, the entry of largest absolute value in any column is less than or equal to DATA%SMALL.

DATA%STATIC is a scalar of type default LOGICAL with default value .TRUE.. If DATA%STATIC = .FALSE., static condensations are not performed. Static condensations are eliminations performed within an individual element (or equation) when a variable appears only in that single element (or equation).

DATA%EXPLOIT is a scalar of type default LOGICAL with default value that controls whether or not the code looks for blocks of zeros in the front. If the elements (or equations) are well-ordered, looking for zeros may add a (small) time overhead without significantly reducing the number of entries in the factors and the flop count. However,

if the elements (or equations) are poorly ordered, substantial savings can be made by exploiting zeros. The default value is .FALSE..

2.4 Information returned to the user

Some components of DATA provide information on the action of MA42_ANALYZE, MA42_SYMBOLIC, MA42_FILES, MA42_FACTORIZE, and MA42_SOLVE. The information returned in DATA to the user is as follows:

DATA%BUFFERS is a rank 1 array of type default INTEGER of dimension 3 which holds the number of buffers used for the factor **UQ** (including right-hand sides), the factor **PL**, and the row and column indices (see also DATA%MAX_BUFFERS).

DATA%DEFICIENCY is a scalar of type default INTEGER which holds, on exit from the final call to MA42_FACTORIZE, with DATA%ERROR = +1 and DATA%SINGULAR = .FALSE., an estimate of the deficiency of the matrix. Otherwise, DATA%DEFICIENCY is set to 0.

DATA%DET is a scalar of type default REAL (or double precision REAL in the DOUBLE version) which holds the natural logarithm of the modulus of the determinant of the matrix **A** (see also DATA%SIGN_DET). If the matrix is found to be singular, DATA%DET is set to zero.

DATA%ERROR is a scalar of type integer used by each of the subroutines as an error and a warning flag. If a call to a routine in the MA42 package is successful, on exit DATA%ERROR has value 0. A nonzero value of DATA%ERROR indicates an error has been detected or a warning issued (see section 2.3). If an error is detected, the information contained in other components of DATA may be incomplete.

DATA%FLOPS is a scalar of type default REAL (or double precision REAL in the DOUBLE version) which holds the number of floating-point operations in the innermost loops. This count includes operations performed during static condensation.

DATA%IOSTAT is a scalar of type default INTEGER which holds the IOSTAT parameter (the IOSTAT parameter is a machine and complier-dependent parameter which is set to zero if a Fortran READ, WRITE, OPEN, or INQUIRE statement is correctly executed and to a positive integer otherwise). See error returns -9, -10, -20, and -23 in section 2.3.

DATA%LARGEST_INDEX is a scalar of type default INTEGER which (on exit from the final call to MA42_ANALYZE) holds the largest integer used to index a variable.

DATA%MAX_BUFFERS is a rank 1 array of type default INTEGER of dimension 3 which holds the maximum number of buffers used to hold a block of pivot rows, a block of pivot columns, and the integers for a block of pivot rows and columns.

DATA%MAX_FRONT is a rank 1 array of type default INTEGER of dimension 2 which holds the maximum number of rows and columns in the frontal matrix.

DATA%NONZEROS is a rank 1 array of type default INTEGER of dimension 2 which holds the number of nonzeros in the **UQ** and **PL** factors.

DATA%PIVOT_INFO is a rank 1 array of type default INTEGER of dimension 7 which is used to return information relating to the pivots chosen and the pivot search. For element entry, DATA%PIVOT_INFO(1) holds the number of diagonal pivots used and DATA%PIVOT_INFO(2) holds the number of off-diagonal pivots used. For equation entry (EQUATION = .TRUE.), DATA%PIVOT_INFO(1) is zero and DATA%PIVOT_INFO(2) holds the number of pivots chosen. DATA%PIVOT_INFO(3) holds the size of the largest pivot block used. DATA%PIVOT_INFO(4) holds the number of pivots chosen which did not satisfy threshold criterion based on the value of DATA%ALPHA. DATA%PIVOT_INFO(5) holds the number of columns examined during pivot searches. DATA%PIVOT_INFO(6) holds the number of non-zeros tested for stability as pivots. DATA%PIVOT_INFO(7) holds the number of non-zeros accessed during the pivot selection process.

DATA%PROBLEM_SIZE is a scalar of type default INTEGER which holds the total number of variables in the problem. This will be less than DATA%LARGEST_INDEX iF the variables are not numbered contiguously.

DATA%SIGN_DET is a scalar of type default INTEGER which is set to +1 (respectively, -1) if the determinant of the matrix is positive (negative). If the matrix is found to be singular, DATA%SIGN_DET is set to 0. (See also DATA%DET).

DATA%STAT is a scalar of type default INTEGER which holds the STAT parameter (the STAT parameter is a machine and complier-dependent parameter which is set to zero if a Fortran ALLOCATE statement is correctly executed and to a positive integer otherwise). See error return -22 in section 2.3.

DATA%STATIC_INFO is a rank 1 array of type default INTEGER of dimension 2 which is used to return information relating to static condensation. DATA%STATIC_INFO(1) holds the number of static condensations performed and DATA%STATIC_INFO(2) the number of potential static condensations. This may be greater than DATA%STATIC_INFO(1) because numerical considerations may prevent immediate elimination of internal variables.

DATA%STORAGE is a rank 1 array of type default INTEGER of dimension 3 which holds the total storage for the **UQ** factor (including right-hand sides),the **PL** factor, and the row and column indices.

DATA%STRM is a rank 1 array of type default INTEGER of dimension 3 which, if direct access files are used, holds the unit numbers used for the direct access files, and is set to zero otherwise.

2.5 Error diagnostics

On successful completion, the subroutines in the HSL_MA42 module will exit with the parameter DATA%ERROR set to 0. Other values for DATA%ERROR and the reasons for them are given below.

A negative value for DATA%ERROR is associated with a fatal error. If DATA%ERROR is negative, a self-explanatory message is, in each case, output on unit DATA%OUTPUT(1) (see section 2.3). The negative values for DATA%ERROR and the subroutine calls which can return them are:

–1 MA42_INITIALIZE has not been called prior to MA42_ANALYZE. (MA42_ANALYZE first entry only).

–2 LENLAST ≤ 0 on entry to MA42_ANALYZE. (MA42_ANALYZE first entry only).

–3 A variable index in the current element (or equation) is out of range. (MA42_ANALYZE, MA42_SYMBOLIC, and MA42_FACTORIZE).

–4 Duplicate occurrences of the same variable index found in the current element (or equation). (MA42_ANALYZE).

–5 LENLAST is too small and DATA%ACTION(1) = .FALSE. (MA42_ANALYZE).

–6 Array X not present when expected. (MA42_FACTORIZE final entry only).

–7 RHS not present on current call to MA42_FACTORIZE but was present on the first call. (MA42_FACTORIZE).

–8 Error in a dimension of an array (MA42_ANALYZE, MA42_SYMBOLIC, and MA42_FACTORIZE).

–9 Error detected when writing to a direct access file. The IOSTAT parameter is returned in DATA%IOSTAT. (MA42_FACTORIZE).

–10 Error detected when reading a direct access file. The IOSTAT parameter is returned in DATA%IOSTAT. (MA42_FACTORIZE and MA42_SOLVE).

–11 Non-positive frontsize. (MA42_FACTORIZE first entry only).

–12 Maximum frontsizes too small to permit the factorization and DATA%ACTION(2) = .FALSE. However, a symbolic factorization has been performed and a lower bound on the space required is given in the output message and in DATA%FRONT_BOUND. See also error +4. (MA42_FACTORIZE).

–13 A variable appears again after it has been fully summed (either an index list for an element (or equation) has been altered since MA42_ANALYZE was called or the order of the elements or equations has been changed). (MA42_SYMBOLIC and MA42_FACTORIZE).

–14 Singularity detected in the matrix during the factorization with the control parameter DATA%SINGULAR =

.TRUE. (MA42_FACTORIZE).

–15 File length less than buffer length. (MA42_FILES).

–16 MA42_FILES not called and LENBUF too small. The error message gives information on the buffer lengths necessary to prevent this failure on a subsequent run with identical data. However, it may be better to use direct access files. See also error +5. (MA42_FACTORIZE).

–17 Insufficient space allocated to one or more of the direct access files to permit a successful factorization. The message gives the space necessary for subsequent success To avoid this error with a subsequent run on identical data, the user must set LENFLE to be at least LENBUF*DATA%BUFFERS. See also error +6. (MA42_FACTORIZE).

–18 Non-positive buffer size. (MA42_FILES and MA42_FACTORIZE first entry only).

–19 MA42_SOLVE has been called but DATA%LFACTOR = .FALSE. (MA42_SOLVE).

–20 Error in Fortran OPEN statement. The IOSTAT parameter is returned in DATA%IOSTAT. (MA42_ANALYZE and MA42_FILES).

–21 Failed to find a unit to which a file could be connected (MA42_ANALYZE and MA42_FILES).

–22 Error in Fortran ALLOCATE statement. The STAT parameter is returned in DATA%STAT. (MA42_ANALYZE, MA42_FACTORIZE, and MA42_SOLVE).

–23 Error in Fortran INQUIRE statement. The IOSTAT parameter is returned in DATA%IOSTAT. (MA42_ANALYZE and MA42_FILES).

–24 MA42_SOLVE has been called after a call to MA42_FINAL. (MA42_SOLVE).

–25 An error was returned on an earlier call. (MA42_ANALYZE, MA42_SYMBOLIC, MA42_FILES, MA42_FACTORIZE, and MA42_SOLVE).

There are three warning messages which are associated with DATA%ERROR = +1, +2, and +3. DATA%ERROR = +3 can only be returned by MA42_ANALYZE and DATA%ERROR = +1 and +2 can only be returned by MA42_FACTORIZE. In each case, a self-explanatory message is output on unit DATA%OUTPUT(2) (see section 2.3). The warnings are:

+1 The matrix **A** has been found to be singular and the control parameter DATA%SINGULAR = .FALSE. (see section 2.3). If the sequence of calls to MA42_FACTORIZE is completed, on exit from the final call DATA%DEFICIENCY (see section 2.4) will hold an estimate of the deficiency of the matrix. DATA%ERROR = +1 will overwrite DATA%ERROR = +2 and +3.

+2 Pivots have been chosen which do not satisfy the threshold criterion determined by DATA%ALPHA (see section 2.3). If the sequence of calls to MA42_FACTORIZE is completed, on exit from the final call DATA%PIVOT_INFO(4) (see section 2.4) will hold the number of such pivots. DATA%ERROR = +2 will overwrite DATA%ERROR = +3.

+3 If returned by MA42_ANALYZE, the argument LENLAST (either the user-supplied value or the default if LENLAST is not present) is too small (that is, an index in the element or equation list VARS exceeds LENLAST). The computation continues and the minimum value for LENLAST which will avoid this warning on a subsequent run with identical data is returned after the final call to MA42_ANALYZE in DATA%LARGEST_INDEX. If returned by MA42_FACTORIZE, the argument FRONTSIZE (either the user-supplied value or the default if FRONTSIZE is not present) is too small. The computation continues but may be less efficient than if FRONTSIZE was large enough. The minimum value for FRONTSIZE which will avoid this warning on a subsequent run with identical data is returned after the final call to MA42_FACTORIZE in DATA%MAX_FRONT.

There are three error returns associated with a positive value for DATA%ERROR. These can only be returned by MA42_FACTORIZE. In each case, a message is output on unit DATA%OUTPUT(2). The user is encouraged to continue the sequence of calls to MA42_FACTORIZE, at the end of which a negative value of DATA%ERROR will be returned, an error message will be output on unit DATA%OUTPUT(1), and information to enable success on a subsequent run with identical data will be output, although the factorization will not have been completed and the information contained in

DATA will not be complete. Note that, if the user does not complete the sequence of calls, MA42_ANALYZE must be recalled before the sequence of calls to MA42_FACTORIZE is restarted. The positive values for DATA%ERROR associated with an error are:

+4 Maximum frontsizes too small and DATA%ACTION(2) = .FALSE.. If the user completes the sequence of calls, a lower bound for the maximum front size required will be returned in DATA%FRONT_BOUND (see error -12). DATA%ERROR = +4 will overwrite DATA%ERROR = +1, +2, +3, +5, and +6.

+5 MA42_FILES not called and LENBUF too small. If the user completes the sequence of calls, the buffer lengths required for a successful run will be output (see error -16). DATA%ERROR = +5 will overwrite DATA%ERROR = +1, +2, and +3.

+6 MA42_FILES was called but LENFLE was too small. If the user completes the sequence of calls, the amount of space required for subsequent success will be given (see error -17). DATA%ERROR = +6 will overwrite DATA%ERROR = +1, +2, and +3.

2.6 Choosing file sizes and frontsizes.

2.6.1 Setting LENBUF and LENFLE

In HSL_MA42, 3 files are required: one for the factor **UQ** (and the corresponding right-hand sides), one for the factor **PL**, and one for the row and column indices of the variables in **UQ** and **PL**. The user can choose whether these files are held in-core or whether they are written to direct access files. The user's choice will depend upon the problem size and the amount of in-core memory available. The indices require a file of length at most (5+2*maxf*)*n*, where *maxf* is the maximum frontsize and *n* is the order of the system. In practice, a file of length considerably less than (5+2*f*)*n*, where *f* is the average frontsize, will normally suffice. The file for **UQ** needs to be of length about (*f*+*nrhs*)*n*, where *nrhs* is the number of right-hand sides input to MA42_FACTORIZE. If MA42_SOLVE is to be called, the file for **PL** needs to be of length about *fn*; otherwise **PL** need not be stored. The default in HSL_MA42 is to store the **PL** factor (DATA%LFACTOR=.TRUE.).

If the user wishes to hold the files in-core, default files sizes of 100000 can be used by omitting the optional argument LENBUF from the first call to MA42_FACTORIZE. Alternatively, MA42_SYMBOLIC can be used to obtain estimates of the required filesizes. To allow pivots to be chosen to avoid numerical instability, LENBUF(1) should be set somewhat larger than DATA%FILE_BOUND(1) + DATA%LARGEST_INDEX **nrhs*, (where *nrhs* is the number of right-hand sides input to MA42_FACTORIZE) and LENBUF(2) and LENBUF(3) should be set somewhat larger than DATA%FILE_BOUND(2) and DATA%FILE_BOUND(3), respectively.

To use direct access files, a call must be made to MA42_FILES. The user must specify the buffer sizes LENBUF and the filesizes LENFLE in this call. For efficiency, LENBUF(1) and LENBUF(3) should be at least 10(*maxf*+*nrhs*) and 10(2*maxf*+5), respectively, where *nrhs* is the number of right-hand sides input to MA42_FACTORIZE. If the **PL** factor is to stored, a value for LENBUF(2) of at least 10*maxf* is recommended. MA42_SYMBOLIC can be used to help set LENFLE. LENFLE(1) should be set somewhat larger than DATA%FILE_BOUND(1) + DATA%LARGEST_INDEX **nrhs*, and LENFLE(2) and LENFLE(3) should be set somewhat larger than DATA%FILE_BOUND(2) and DATA%FILE_BOUND(3), respectively.

2.6.2 Choosing the maximum frontsize

In addition to the in-core buffers, in-core memory (approximately) equal to the product of the row and column frontsizes is required. To keep the amount of in-core memory low, the user should order the element matrices $\mathbf{A}^{(k)}$ so that the frontsize is small. For example, a very rectangular grid should be ordered pagewise parallel to the short side of the rectangle. For finite-element calculations, we recommend using the HSL routine MC63 to obtain an efficient element ordering prior to using HSL_MA42. For equation entry, the equations should be ordered so that the matrix **A** is banded. MC62 may be used to obtain a good ordering.

If the user has obtained an efficient ordering, estimates of the maximum row and column frontsizes required are likely to be available. These should be used to set the argument FRONTSIZE on the first call to MA42_FACTORIZE. If

no estimates are available, the user can either use the default maximum frontsize of 200 (by omitting the optional argument FRONTSIZE from the first call to MA42_FACTORIZE), or use MA42_SYMBOLIC to obtain a lower bound on the maximum frontsize required. If MA42_SYMBOLIC is used, to allow pivots to be chosen to avoid numerical instability, the user should set FRONTSIZE somewhat larger than DATA%PRIVATE%FRONT_BOUND. If there is insufficient space for the factorization (DATA%ERROR = -12), on exit from the final call to MA42_FACTORIZE, DATA%FRONT_BOUND will hold lower bounds on the maximum frontsize necessary for a successful factorization. (See also the error return DATA%ERROR = +4 in section 2.5)

3 GENERAL INFORMATION

3.1 Summary of information.

Other routines called directly: The BLAS routines ISAMAX/IDAMAX, SAXPY/DAXPY, SGER/DGER, SGEMV/DGEMV, STPSV/DTPSV, SGEMM/DGEMM, STRSM/DTRSM.

Input/output: In the event of errors, diagnostic messages are printed. The output streams for these messages are controlled by DATA%OUTPUT (see section 2.3). Stream DATA%OUTPUT(1) is used for error messages (DATA%ERROR < 0) and stream DATA%OUTPUT(2) for warnings (DATA%ERROR > 0).

Restrictions:

LENBUF(:) ≥ 1 (MA42_FILES and MA42_FACTORIZE, first call).

LENFLE(:) ≥ LENBUF(:) (MA42_FILES).

If present, LENLAST ≥ 1 (MA42_ANALYZE, first call).

VARS(:) ≥ 1 (MA42_ANALYZE, MA42_SYMBOLIC, and MA42_FACTORIZE).

VARS(I) ≠ VARS(J), I ≠ J (MA42_ANALYZE).

If present, FRONTSIZE(:) > 0 (MA42_FACTORIZE).

Portability: Fortran 90.

4 METHOD

The method used is a modification of the unsymmetric frontal scheme of Hood (1976). HSL_MA42 is a Fortan 90 version of the Fortran 77 HSL routine MA42 (Duff and Scott 1993). MA42 was developed from the work by Cliffe, Jackson, Rae, and Winters (1978) and from the earlier HSL routine MA32 (see Duff 1981, 1983).

The elements or equations are assembled into an in-core frontal matrix one at a time. A variable which has appeared for the last time (i.e. does not occur in future elements or equations) is fully summed and is available for use as a pivot in the Gaussian elimination.

Eliminations are performed whenever a variable in the frontal matrix is fully summed and satisfies a numerical tolerance. Once all possible eliminations for the current element (or equation) have been performed, the pivot rows and, optionally, the pivot columns are written to in-core buffers and thence, if requested, to direct access files. In order to prevent the amount of in-core memory required becoming too large, the user should order the elements (or equations) so that the same variable does not occur in elements (or equations) which are widely apart in the ordering. Thus, for example, in a finite-element problem with a narrow pipe geometry, elements should be ordered across the cross-section of the pipe rather than along its length. For finite-element calculations, an efficient element ordering can be obtained using the HSL routine MC43 (see Duff, Reid, and Scott 1989). An estimate of the in-core memory required may be obtained by calling MA42_SYMBOLIC, which performs a symbolic factorization.

References.

Cliffe, K.A., Jackson, C.P., Rae, J., and Winters, K.H. (1978) Finite element flow modelling using velocity and pressure variables. Report AERE R.9202, HMSO London.

Duff, I.S. (1981) MA32 – A package for solving sparse unsymmetric systems using the frontal method. Report AERE R.10079, HMSO London.

Duff, I.S. (1983) Enhancements to the MA32 package for solving sparse unsymmetric equations. Report AERE R.11009, HMSO London.

Duff, I.S., Reid, J.K., and Scott, J.A. (1989) The use of profile reduction algorithms with a frontal code. Int. J. Numer. Meth. Engng **28**, 2555-2568.

Duff, I.S. and Scott, J.A. (1993) MA42 – A new frontal code for solving sparse unsymmetric systems. Rutherford Appleton Laboratory Report, RAL-93-064.

Hood, P. (1976) Frontal solution program for unsymmetric matrices. Int. J. Numer. Meth. Engng **10**, 379-399.

5 EXAMPLE OF USE

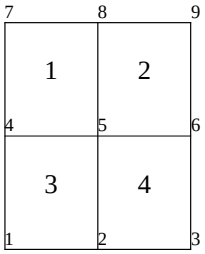
We give an example of the code required to solve a set of equations using the moduleHSL_MA42 when input is by elements (example 5.1) and when input is by equations (example 5.2).

Example 5.1 illustrates the simplest calling sequence for HSL_MA42. Direct access files are not used and no symbolic factorization is performed. In this example, we wish to solve for one right-hand side at the same time as the factorization and we do not wish to retain the factors for solving further systems.

Example 5.2 illustrates the full calling sequence for the HSL_MA42 package. In this example, we wish to solve $\mathbf{Ax}=\mathbf{b}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$. We do not supply right-hand sides with the equations but thereafter we want to solve each system for two right-hand sides. Direct access files are used to hold the factors and MA42_SYMBOLIC is used to perform a symbolic factorization.

5.1 Example of element input

We wish to solve the following simple finite-element problem in which the finite-element mesh comprises four 4-noded quadrilateral elements with one freedom at each node i, $1 \le i \le 6$ (the nodes 7, 8, and 9 are assumed constrained).



The four elemental matrices $\mathbf{A}^{(k)}$ $(1 \le k \le 4)$ are

$$\begin{matrix} 4 \\ 5 \end{matrix}\begin{pmatrix} 2. & 1. \\ 1. & 7. \end{pmatrix} \qquad \begin{matrix} 5 \\ 6 \end{matrix}\begin{pmatrix} 3. & 2. \\ 2. & 8. \end{pmatrix} \qquad \begin{matrix} 4 \\ 5 \\ 1 \\ 2 \end{matrix}\begin{pmatrix} 4. & 3. & 2. & 3. \\ 3. & 1. & 3. & 2. \\ 2. & 3. & 6. & 1. \\ 3. & 2. & 1. & 5. \end{pmatrix} \qquad \begin{matrix} 5 \\ 6 \\ 2 \\ 3 \end{matrix}\begin{pmatrix} 2. & 1. & 8. & 3. \\ 1. & 3. & 2. & 2. \\ 8. & 2. & 2. & 5. \\ 3. & 2. & 5. & 4. \end{pmatrix},$$

where the variable indices are indicated by the integers before each matrix (columns are identified symmetrically to rows). The corresponding elemental vectors $\mathbf{b}^{(k)}$ $(1 \le k \le 4)$ are

$$\begin{pmatrix} 3. \\ 8. \end{pmatrix} \qquad \begin{pmatrix} 5. \\ 10. \end{pmatrix} \qquad \begin{pmatrix} 12. \\ 9. \\ 12. \\ 11. \end{pmatrix} \qquad \begin{pmatrix} 14. \\ 8. \\ 17. \\ 14. \end{pmatrix}.$$

The following program is used to solve this problem. In this program we read the element data into arrays ELTPTR (location of first entry of element), ELTVAR (variable indices), VALUE (numerical values), and RHSVAL (right-hand sides). This method of storing the element data is used here for illustrative purposes; the user may prefer, for example, to read in the element data from a direct access file.

```
! Example to illustrate the use of MA42: element entry.

      PROGRAM EXAMPLE

      USE HSL_MA42_DOUBLE

      TYPE (MA42_DATA) DATA

      INTEGER IELT,J,JSTRT,KSTRT,NELT,NVAR,NZ,RHSCRD,VALCRD,NDF,MAXE

      REAL(WP),ALLOCATABLE :: REALS(:,:),RHS(:,:),VALUE(:),X(:,:)
      INTEGER,ALLOCATABLE :: ELTNUM(:),ELTVAR(:)

! Call to MA42_INITIALIZE
     CALL MA42_INITIALIZE(DATA)
! No need to store L factor since MA42_SOLVE is not going to be called.
     DATA%LFACTOR = .FALSE.

! Read in the element data.
! NELT is number of elements.
! ELTVAR contains lists of the variables belonging to the finite
! elements, with those for element 1 preceding those for element
! 2, and so on. ELTNUM contains the number of variables
! in each element. NZ is the total number of entries in the
! element lists and MAXE is the largest number of variables
! in an element.

      READ (5,FMT=*) NELT
      ALLOCATE (ELTNUM(NELT))
      READ (5,FMT=*) ELTNUM

      NZ = SUM(ELTNUM)
      MAXE = MAXVAL(ELTNUM)
      ALLOCATE (ELTVAR(NZ))
      READ (5,FMT=*) ELTVAR

! Calls to MA42_ANALYZE to establish when each variable is fully assembled
      JSTRT = 1
      DO  IELT = 1,NELT
         JSTOP = JSTRT + ELTNUM(IELT) - 1

         CALL MA42_ANALYZE(ELTVAR(JSTRT:JSTOP),DATA)

         IF (DATA%ERROR < 0) GO TO 60
         JSTRT = JSTOP + 1
      END DO

! Allocate solution array X
      NDF = DATA%LARGEST_INDEX
      ALLOCATE (X(NDF,1))
```

```
! Input elemental matrices and right-hand sides.
! VALCRD is the number of numerical values to be input.
! VALUE contains lists of the numerical values in the elemental
! matrices, with element 1 preceding element 2, and so on.
! Since the elemental matrices are symmetric only the upper
! triangular part is stored.

      READ (5,FMT=*) VALCRD
      ALLOCATE (VALUE(VALCRD))
      READ (5,FMT=*) VALUE

! RHSCRD is the number of right-hand side numerical values to
! be input.
! RHS contains lists of the right-hand side numerical values
! corresponding to each of the elements in order.

      READ (5,FMT=*) RHSCRD
      ALLOCATE (RHS(RHSCRD,1))
      READ (5,FMT=*) RHS
      ALLOCATE (REALS(1:MAXE,1:MAXE))

! Call MA42_FACTORIZE.
      JSTRT = 1
      KSTRT = 1
      DO  IELT = 1,NELT
         NVAR = ELTNUM(IELT)
         JSTOP = JSTRT + ELTNUM(IELT) - 1
         DO  J = 1,NVAR
            REALS(J:NVAR,J) = VALUE(KSTRT:KSTRT+NVAR-J)
            REALS(J,J:NVAR) = VALUE(KSTRT:KSTRT+NVAR-J)
            KSTRT = KSTRT + NVAR - J + 1
         END DO
         CALL MA42_FACTORIZE(ELTVAR(JSTRT:JSTOP),REALS(1:NVAR,1:NVAR),  &
                        DATA,RHS=RHS(JSTRT:JSTOP,1:1),X=X)

         IF (DATA%ERROR < 0) GO TO 60
         JSTRT = JSTOP + 1
       END DO
      CALL MA42_FINAL(DATA)

!  Solution is in X
      WRITE (6,FMT=9000)
      WRITE (6,FMT=9010) X(:,:)
! Deallocate arrays.
      DEALLOCATE (ELTNUM,ELTVAR,REALS,RHS,VALUE,X)
      GO TO 70

   60 WRITE (6,FMT=9020)
      WRITE (6,FMT=9030) DATA%ERROR
      CALL MA42_FINAL(DATA)

   70 STOP

 9000 FORMAT (/3X,'The solution is:')
 9010 FORMAT (/6G12.4)
 9020 FORMAT (/3X,'Error return')
 9030 FORMAT (3X,'DATA%ERROR = ',I3)

      END PROGRAM EXAMPLE
```

The input data used for this problem is:

```
 4
 2    2    4    4
 4    5    5    6    4    5    1    2    5    6    2    3
26
 2.   1.   7.   3.   2.   8.   4.   3.   2.   3.   1.   3.
 2.   6.   1.   5.   2.   1.   8.   3.   3.   2.   2.   2.
 5.   4.
12
 3.   8.   5.  10.  12.   9.  12.  11.  14.   8.  17.  14.
```

This produces the following output:

```
THE SOLUTION IS:

1.000       1.000       1.000       1.000       1.000       1.000
```

5.2 Example of equation input

We wish to factorize the matrix **A** given by

$$\mathbf{A} = \begin{pmatrix} 3. & 2. & 5. \\ 1. & 3. & 2. \\ 6. & 1. & 8. \end{pmatrix}.$$

We then want to solve $\mathbf{Ax} = \mathbf{b}$ for the two right-hand sides

$$\begin{pmatrix} 4. \\ 4. \\ 3. \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 5. \\ 15. \\ -4. \end{pmatrix},$$

and to solve $\mathbf{A}^{\mathrm{T}}\mathbf{x} = \mathbf{b}$ for the two right-hand sides

$$\begin{pmatrix} -5. \\ 5. \\ -4. \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 12. \\ 12. \\ 19. \end{pmatrix}.$$

The following program is used to solve this problem. Note that to illustrate the full calling sequence for the MA42 package, we call MA42_SYMBOLIC to obtain a bound on the front size, then call MA42_FILES to open (named) direct access files. MA42_FACTORIZE is called with no right-hand sides and finally MA42_SOLVE is called to solve $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{x} = \mathbf{b}$.

```
! Example to illustrate the use of MA42: equation entry.

      PROGRAM EXAMPLE2

      USE HSL_MA42_DOUBLE

      TYPE (MA42_DATA) DATA

      INTEGER :: I,IEQ,J,JSTRT,JSTOP,NDF,NEQ,NRHS,NZ

      REAL(WP),ALLOCATABLE ::  B(:,:),X(:,:),REALS(:,:)

      INTEGER LENBUF(3),LENFLE(3)
      INTEGER,ALLOCATABLE :: IRN(:),JP(:)
```

```
      INTRINSIC MAX

! Call to MA42_INITIALIZE (equation entry)
      CALL MA42_INITIALIZE(DATA,.TRUE.)

! Read in the data.
! NEQ is number of equations.
! IRN contains lists of the variables belonging to the
! equations, with those for equation 1 preceding those for equation
! 2, and so on. JP(I) points to the position in IRN
! of the first variable in equation I.

      READ(5,FMT=*) NEQ
      ALLOCATE (JP(NEQ+1))
      READ(5,FMT=*) JP
! NZ is the total number of entries in the variable lists.
      NZ = JP(NEQ+1) - 1
      ALLOCATE (IRN(NZ))
      READ(5,FMT=*) IRN

! Calls to MA42_ANALYZE to establish when each variable is fully assembled
      DO  IEQ = 1,NEQ
         JSTRT = JP(IEQ)
         JSTOP = JP(IEQ+1) - 1

         CALL MA42_ANALYZE(IRN(JSTRT:JSTOP),DATA)

         IF (DATA%ERROR < 0) GO TO 120
      END DO

! Calls to MA42_SYMBOLIC to perform symbolic factorization.
      DO  IEQ = 1,NEQ
         JSTRT = JP(IEQ)
         JSTOP = JP(IEQ+1) - 1

         CALL MA42_SYMBOLIC(IRN(JSTRT:JSTOP),DATA)

         IF (DATA%ERROR < 0) GO TO 120
      END DO

! Call to MA42_FILES to open direct access files.
! Choose buffer and file sizes (allow for pivoting).
      LENBUF = 30
      DO  I = 1,3
         LENFLE(I) = INT(1.2*DATA%FILE_BOUND(I))
         LENFLE(I) = MAX(LENFLE(I),LENBUF(I))
      END DO
      CALL MA42_FILES(LENBUF,LENFLE,DATA)
      IF (DATA%ERROR < 0) GO TO 120

! Input the numerical values for each row.
! Number of numerical values to be input is NZ.
! REALS contains lists of the numerical values
! with equation 1 preceding equation 2, and so on.

      ALLOCATE (REALS(1,NZ))
      READ(5,FMT=*) REALS

! Perform decomposition but do not solve for any right-hand sides.
      DO  IEQ = 1,NEQ
         JSTRT = JP(IEQ)
```

```
          JSTOP = JP(IEQ+1) - 1
          CALL MA42_FACTORIZE(IRN(JSTRT:JSTOP),REALS(1:1,JSTRT:JSTOP),   &
                       DATA,FRONTSIZE=INT(1.2*DATA%FRONT_BOUND))

          IF (DATA%ERROR < 0) GO TO 120
       END DO

 !  We wish to solve AX = B for two right-hand sides.
 !  Allocate B and X. Read right-hand sides into B.
       NRHS = 2
       NDF = DATA%LARGEST_INDEX
       ALLOCATE (B(NDF,NRHS),X(NDF,NRHS))
       READ (5,FMT=*) B

       CALL MA42_SOLVE(B,X,DATA)

       IF (DATA%ERROR < 0) GO TO 120

 !  Solution for J-th right-hand side is in X(.,J), J=1,NRHS
       WRITE (6,FMT=9080)
       DO  J = 1,NRHS
          WRITE (6,FMT=9040) J
          WRITE (6,FMT=9050) X(:,J)
       END DO

 ! Now read in right-hand sides for A(T)X = B
       READ (5,FMT=*) B

 ! Final call to MA42_SOLVE
       CALL MA42_SOLVE(B,X,DATA,TRANS = .TRUE.)

       IF (DATA%ERROR < 0) GO TO 120

       CALL MA42_FINAL(DATA)

       WRITE (6,FMT=9070)
       DO  J = 1,NRHS
          WRITE (6,FMT=9040) J
          WRITE (6,FMT=9050) X(:,J)
       END DO

       DEALLOCATE (IRN,JP,B,X,REALS)
       GO TO 130

 ! Print appropriate fatal error diagnostic
   120 WRITE (6,FMT=9000)
       WRITE (6,FMT=9010) DATA%ERROR
       CALL MA42_FINAL(DATA)
   130 STOP

  9000 FORMAT (/3X,'Error return')
  9010 FORMAT (3X,'DATA%ERROR = ',I3)
  9040 FORMAT (/3X,'The solution for right-hand side number ',I2,' is:')
  9050 FORMAT (6G12.4)
  9070 FORMAT (/3X,'*** A(T)X = B ***')
  9080 FORMAT (/3X,'*** AX = B ***')

       END PROGRAM EXAMPLE2
```

The input data used for this problem is:

```
 3
 1    4    7   10
 1    2    3    1    2    3    1    2    3
 3.   2.   5.   1.   3.   2.   6.   1.   8.
 4.   4.   3.
 5.  15.  -4.
-5.   5.  -4.
12.  12.  19.
```

This produces the following output:

```
 *** AX = B ***

 The solution for right-hand side number  1 is:
-1.000       1.000       1.000

 The solution for right-hand side number  2 is:
 1.000       6.000      -2.000

 *** A(T)X = B ***

 The solution for right-hand side number  1 is:
 2.000       1.000      -2.000

 The solution for right-hand side number  2 is:
 1.000       3.000       1.000
```