1 SUMMARY

To **compute an orthogonal factorization of a sparse overdetermined matrix A and optionally to solve the least squares problem** $\min \|\mathbf{b}-\mathbf{Ax}\|_2$**.** Given a sparse matrix **A** of order $m \times n$, $m \geq n$, of full column rank, this subroutine computes the **QR** factorization $\mathbf{A}=\mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}$ where **Q** is an $m \times m$ orthogonal matrix and **R** is an $n \times n$ upper triangular matrix.

Given an m-vector **b**, this subroutine solves the least squares problem $\min \|\mathbf{b}-\mathbf{Ax}\|_2$, either computing the solution **x** using Golub's method (Numerical methods for solving least squares problems. *Numer. Math.* **7**, 1965, 206-216), $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{Q}^{\mathrm{T}}\mathbf{b}$, or using the seminormal equations method ($\mathbf{R}^{\mathrm{T}}\mathbf{Rx}=\mathbf{A}^{\mathrm{T}}\mathbf{b}$). In the latter case, the **Q** factor need not be stored.

Given an n-vector **b**, this subroutine may also compute the minimum 2-norm solution of the linear system $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$, by solving $\begin{bmatrix}\mathbf{R}^{\mathrm{T}}\mathbf{0}\end{bmatrix}\mathbf{z}=\mathbf{b}$ and performing the multiplication $\mathbf{x}=\mathbf{Qz}$, or, if the **Q** factor is not stored, by solving $\mathbf{R}^{\mathrm{T}}\mathbf{Rz}=\mathbf{b}$ and performing the multiplication $\mathbf{x}=\mathbf{Az}$.

The subroutine can also solve systems with the coefficient matrix $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}$ or $\begin{bmatrix}\mathbf{R}^{\mathrm{T}}\mathbf{0}\end{bmatrix}$, or will compute the product of **Q** or $\mathbf{Q}^{\mathrm{T}}$ with a vector.

The method used is based on the multifrontal approach and makes use of Householder transformations. Because an ordering for the columns is chosen using the pattern of the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, this code is not designed for matrices with full rows.

The code is written in Fortran 77. A version exists for globally addressable parallel computers. The parallel versions are machine dependent but only require simple features like starting parallel tasks and locks. In principle, a code can be supplied for any shared memory parallel machine, but the only platforms on which the shared memory code has been tested are the SGI Origin 2000 and the CRAY C98.

ATTRIBUTES — Versions: `MA49A/AD`. **Calls:** The `HSL` routines: `MA59`, `MC41`, and `MC47`. The `BLAS` routines: `I_AMAX`, `_AXPY`, `_COPY`, `_DOT`, `_GEMM`, `_GER`, `_GEMV`, `_NRM2`, `_SYRK`, `_SWAP`, `_TRSM`, `_TRSV`. **Date:** May 1998. **Language:** Fortran 77 with parallel extensions to Fortran language. **Origin:** P. R. Amestoy (ENSEEIHT-IRIT, Toulouse, France), I. S. Duff (Rutherford Appleton Laboratory), and C. Puglisi (CERFACS, Toulouse, France). **Conditions on external use:** (i), (ii), (iii) and (iv).

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences

There are two entries:

(a) `MA49I/ID` sets default values for the components of the arrays that hold control parameters. The default values are such that the routine keeps **Q** and solves the least-squares problem associated with an input matrix in coordinate form, using Golub's method in a uniprocessor environment. The user must call `MA49I/ID` before calling `MA49A/AD`. If non-default values for any of the control parameters are required, they should be set immediately after the call to `MA49I/ID`.

(b) `MA49A/AD` provides factorization and solution facilities. The user must supply the sparse matrix either in coordinate form (the default) or by rows as a set of sparse vectors. The input is discussed in Section 2.1.2. The user can use `MA49A/AD` to first preprocess the matrix to obtain the block upper triangular form of the matrix (Pothen and Fan, *ACM Trans. Math. Softw.* **16** 1990, 303-324). This option is available only if the solution of the least-squares problem with Golub's method has been requested. Information on this form is given in Section 2.5. Options exist to perform iterative refinement to improve the solution, and to perform error analysis.

As in most sparse direct codes, the solution is divided into three phases. An **analysis** phase examines the structure of the matrix and generates an ordering and data structures to facilitate the second phase. In this second phase, the **factorization** phase, the factors of the matrix (here the **QR** factors) are generated. These factors are used to solve systems of equations in the **solve** phase. The parameter `JOB` is used to determine which phase is invoked. Note that new matrices with the same sparsity pattern can be factorized without calling the analysis phase again, and the factors

generated in the factorization phase can be used in several solve phases. MA49A/AD offers similar functionality to other sparse direct solvers, depending on the value of the parameter JOB. These are:

(i) JOB = 1 performs the analysis phase. This phase accepts the pattern of **A** and chooses a column permutation to preserve sparsity. It uses the pattern of $\mathbf{A}^T\mathbf{A}$ but ignores numerical values. This entry then constructs subsidiary information for the factorization phase. An option exists for the user to specify the column permutation, in which case only the necessary information for the factorization phase will be generated.

(ii) JOB = 2 performs the factorization phase. This phase performs the **QR** factorization of a matrix **A** using the information from a previous analysis phase. The **Q** factor may be discarded.

(iii) JOB = 3 performs the solve phase. In this phase, operations or solutions are performed using the factors generated by the factorization phase. It solves either the least-squares problem $\min \|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2$ or computes the minimum 2-norm solution of $\mathbf{A}^T\mathbf{x}=\mathbf{b}$, or performs the matrix-vector multiplication $\mathbf{Q}\mathbf{b}$ or $\mathbf{Q}^T\mathbf{b}$ or computes the solution of $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ or $\begin{bmatrix}\mathbf{R}^T\mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$.

A solve call must be preceded by a factorization call, which in turn must be preceded by an analysis call. Since the information passed from one call to the next is not corrupted by the second, several factorization calls with JOB=2 for matrices with the same sparsity pattern but different values may follow a single analysis call with JOB=1, and similarly several solve calls with JOB=3 can be used for different right-hand sides **b**.

The parameter JOB can also have values 4, 5, or 6, and they invoke combinations of the three basic operations: analysis, factorization and solve (see Section 2.1.2).

2.1.1 To set default values of controlling parameters

The single precision version

```
CALL MA49I(CNTL, ICNTL, KEEP)
```

The double precision version

```
CALL MA49ID(CNTL, ICNTL, KEEP)
```

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 10 that need not be set by the user. On return it contains default values. For further information see Section 2.2.

ICNTL is an INTEGER array of length 30 that need not be set by the user. On return it contains default values. For further information see Section 2.2.

KEEP is an INTEGER array of length 50 that need not be set by the user. MA49I/ID sets default values for some of the components of KEEP. This array must be preserved by the user between calls to MA49A/AD.

2.1.2 To factorize the matrix and solve systems of equations

The single precision version

```
 CALL MA49A (JOB, M, N, NE, IRN, JCN, IPTR, JPTR, A, B, X, KEEP,
*            IS, MAXIS, S, MAXS, Q, MAXQ, CNTL, ICNTL, INFO, RINFO)
```

The double precision version

```
 CALL MA49AD (JOB, M, N, NE, IRN, JCN, IPTR, JPTR, A, B, X, KEEP,
*            IS, MAXIS, S, MAXS, Q, MAXQ, CNTL, ICNTL, INFO, RINFO)
```

JOB is an INTEGER variable that must be set by the user to determine the actions to be performed by MA49A/AD. Possible values of JOB are:

- 1 Analysis
- 2 Numerical factorization
- 3 Solution
- 4 Analysis and numerical factorization
- 5 Numerical factorization and solution
- 6 Analysis, numerical factorization and solution

M is an INTEGER variable that must be set by the user to the number of rows *m* of the matrix **A**. It is not altered by the subroutine. **Restriction:** M ≥ N.

N is an INTEGER variable that must be set by the user to the number of columns *n* of the matrix **A**. It is not altered by the subroutine. **Restriction:** N ≥ 1.

NE is an INTEGER variable that must be set by the user to the number of entries in **A**. It is not altered by the subroutine. **Restriction:** NE ≥ 1.

IRN is an INTEGER array of length NE.

If the matrix is input in coordinate form (the default with ICNTL(8)=0), IRN(K), K=1, ..., NE must on a JOB=1, 4, or 6 call (involving the analysis phase), be set by the user to hold the row indices of the matrix entries. It is altered by the subroutine during the analysis phase and, on exit, will hold the map to reorder by rows the real values contained in A.

If the matrix is input by rows (ICNTL(8)=1), IRN need not be set by the user. It will not be accessed unless the block triangular form option is invoked (ICNTL(6)=1) or out-of-range entries are present, in which case it will hold the map to reorder by rows the real values contained in A.

Duplicate entries are summed and out-of-range entries ignored. IRN must be passed unchanged to a call with JOB=2 or 5.

JCN is an INTEGER array of length NE.

If the matrix is input in coordinate form (the default with ICNTL(8)=0), JCN(K), K=1, ..., NE must, on a JOB=1, 4, or 6 call, be set by the user to hold the column indices of the matrix entries so that the entries of the matrix are given by the set (IRN(K), JCN(K)), K=1, ..., NE. It is altered by the subroutine during the analysis phase and, on exit, will hold the column indices ordered by rows.

If the matrix is input by rows, (ICNTL(8)=1), JCN must be set by the user to hold the column indices ordered by rows. It will be altered by the subroutine only if the block triangular form option is invoked (ICNTL(6)=1) or out-of-range entries are present.

Duplicate entries are summed and out-of-range entries ignored. JCN must be passed unchanged to a call with JOB=2 or 5.

IPTR is an INTEGER array of length M+1.

If the matrix is input in coordinate form (ICNTL(8)=0), IPTR need not be set by the user. On exit, IPTR(K) will hold the position in JCN of the first column index in row K and IPTR(M+1) the position immediately after the last column index in JCN.

If the matrix is input by rows, (ICNTL(8)=1), IPTR(K) must, on a JOB=1, 4, or 6 call, be set to the position in JCN of the first column index in row K and IPTR(M+1) to the position immediately after the last column index in JCN. It will be altered by the subroutine only if the block triangular form option is invoked (ICNTL(6)=1) or out-of-range entries are present.

Duplicate entries are summed and out-of-range entries ignored. On exit, IPTR(M+1)–1 will be equal to NE less the number of entries with an index out-of-range. IPTR must be passed unchanged to a call with JOB=2 or 5.

JPTR is an INTEGER array of length N+1. It is accessed only if the block upper triangular option is invoked (ICNTL(6)=1) (see Section 2.5). In this case, JPTR need not be set by the user and will hold, on exit from an analysis phase (JOB=1, 4 or 6), information on the block triangular structure that must be passed unchanged to other calls for a matrix of the same structure. If the block triangular option is not invoked, JPTR is not accessed and it can be declared to have size 1.

A is a REAL (DOUBLE PRECISION in the D version) array of length NE. It is not accessed if JOB=1.

If the matrix is input in coordinate form (ICNTL(8)=0), the user must, on a factorization call with JOB=2, 4, 5, or 6, set A(K) to the value of the entry in row IRN(K) and column JCN(K) of the matrix, where IRN(K) and JCN(K), K=1, ..., NE, are as originally supplied by the user to the analysis phase. It is altered by the subroutine during the factorization phase, where it is reordered following the map contained in IRN.

If the matrix is input by rows, (ICNTL(8)=1), the user must, on a factorization call with JOB=2, 4, 5, or 6, set A(K) to the value of the entry corresponding to the value of JCN(K) originally supplied by the user to the analysis phase. It will be altered by the subroutine during the factorization phase if the block triangular form option is invoked (ICNTL(6)=1) or out-of-range entries are present. In this case, A is reordered following the map contained in IRN.

A must be passed unchanged to a call with JOB=3.

B is a REAL (DOUBLE PRECISION in the D version) array, which is only accessed if JOB = 3, 5, or 6. In this case, B must be of length M if $\mathbf{Ax}=\mathbf{b}$ or $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ is to be solved, or if $\mathbf{Qb}$ or $\mathbf{Q}^{\mathrm{T}}\mathbf{b}$ are being computed. It must be of length N if $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$ or $\begin{bmatrix}\mathbf{R}^{\mathrm{T}}\mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ is to be solved. On entry, B(I), I=1,2, ... must hold the I-th component of the right-hand side of the equations being solved or of the vector being multiplied. It is altered by the routine. If an operation is being performed with the **Q** factor (ICNTL(13)=7 or 8), then, on exit, B(I), I=1,2, ... M, will hold the components of the product.

X is REAL (DOUBLE PRECISION in the D version) array, which is only accessed if JOB = 3, 5, or 6 and need not be set by the user. In this case, X must be of length N if $\mathbf{Ax}=\mathbf{b}$ or $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ is to be solved. It must be of length M if $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$ or $\begin{bmatrix}\mathbf{R}^{\mathrm{T}}\mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ is to be solved. On exit, X(I), I=1, 2, ... will hold the I-th component of the solution vector.

KEEP is an INTEGER array of length 50 that need not be set by the user. On an analysis call, it must be unchanged since the call to MA49I/ID. On a factorization call, it must be unchanged since the corresponding analysis call. On a solve call, it must be unchanged since the corresponding factorization call. KEEP is altered on an analysis or a factorization call.

IS is an INTEGER array of length MAXIS. It need not be set by the user unless the column permutation is provided (ICNTL(7)=1). In this case, IS(I), I=1, ..., N must hold the position of variable I in the column order. Note that, even when the permutation is provided by the user, the analysis must be performed before numerical factorization. IS must be preserved between corresponding calls to MA49A/AD.

MAXIS is an INTEGER variable that must be set by the user to the size of the array IS. MAXIS must be at least 2*NE+13*N+M+1 to be able to run the analysis phase, and 3*NE+15*N+3*M+3 if the block triangular option is invoked (ICNTL(6)=1). For numerical factorization, MAXIS must be greater than INFO(16) as output from the analysis phase on the same matrix. It is not altered by the subroutine.

S is a REAL (DOUBLE PRECISION in the D version) array of length MAXS. It need not be set by the user but must be preserved between corresponding calls to MA49A/AD.

MAXS is an INTEGER variable that must be set by the user to the size of the working array S. It is not accessed in the analysis phase (JOB=1). After the analysis phase, an estimate of the minimum space required to run the numerical factorization is available in INFO(17) or INFO(18), for a multiprocessor or uniprocessor environment, respectively. Increasing the value of MAXS can improve the performance of the factorization phase in a multiprocessor environment. It is not altered by the subroutine.

Q is a REAL (DOUBLE PRECISION in the D version) array of length MAXQ. It need not be set by the user. In the default option, it must be preserved between corresponding calls to MA49A/AD. If the **Q** factor is discarded (ICNTL(9)=1), Q is not accessed and so can be declared to have size 1.

MAXQ is an INTEGER variable that must be set by the user to the size of the working array Q. If the **Q** factor is discarded (ICNTL(9)=1), it can be set to 1. It is not accessed in the analysis phase (JOB=1). After the analysis phase, an estimate of the space required to run the factorization and the solve phases is available in INFO(23). Note that the estimate given in INFO(23) is usually a large upper bound on the space actually needed. It is not altered by the subroutine.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 10 that contains control parameters and must be set by the user. Default values for the components may be set by a call to MA49I/ID. Details of the control parameters are given in Section 2.2.

ICNTL is an INTEGER array of length 30 that contains control parameters and must be set by the user. Default values for the components may be set by a call to MA49I/ID. Details of the control parameters are given in Section 2.2.

INFO is an INTEGER array of length 50 that need not be set by the user. On return from MA49A/AD, a non-negative value for INFO(1) indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3. For the meaning of the value of other components of INFO, see Section 2.2.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of length 20 that need not be set by the user. This array supplies information on the execution of MA49A/AD. For the meaning of the value of the components of RINFO, see Section 2.2.

2.2 Arrays for control and information

The components of the arrays CNTL and ICNTL control the action of MA49A/AD. Default values are set by MA49I/ID. Only the first two components of CNTL and the first 14 components of ICNTL are presently used by MA49A/AD. The

components of the arrays INFO and RINFO provide information on the action of MA49A/AD.

CNTL(1) has default value zero and is used to test for singularity. The matrix will be declared singular if the norm of a column in the reduced matrix is less than or equal to this value.

CNTL(2) has default value 0.5 and is used by the solve phase to determine when to stop iterative refinement (see Section 2.4). 0 < CNTL(2) < 1. Values out of this range will be treated as the default value.

CNTL(3) to CNTL(10) are set to zero by MA49I/ID but are not currently used by MA49A/AD.

ICNTL(1) has default value 6. It is the output stream for error messages. If it is negative, these messages will be suppressed.

ICNTL(2) has default value 6. It is the output stream for warning messages. If it is negative, these messages are suppressed.

ICNTL(3) has default value 6. It is the output stream for monitoring printing. If it is negative, these messages are suppressed.

ICNTL(4) has default value –1. It is the output stream for the printing of statistics. If it is negative, the statistics are not printed.

ICNTL(5) is used by MA49A/AD to control printing of error, warning, and monitoring messages. It has default value 2. Possible values are:

- <1 No messages output.
- 1 Only error messages printed.
- 2 Errors and warnings printed.
- 3 Errors and warnings and terse monitoring (only first ten entries of arrays printed).
- ≥4 Errors and warnings and all information on input and output parameters printed.

ICNTL(6) has default value 0 and is only accessed on a call that includes an analysis phase (JOB = 1, 4, or 6). If ICNTL(6)=1, the routine checks the reducibility of the matrix. For all other values of ICNTL(6), the reducibility is not checked. If the matrix is found to be reducible, the routine computes its block upper triangular form, and INFO(3) will be set to 1. In this case, only the least-squares solution using Golub's method (ICNTL(13)=1) can be computed. An attempt to use a different solver will result in the error return INFO(1)=–12. If ICNTL(6)=1 and the matrix is reducible, ICNTL(11) and ICNTL(12) are considered to be 0 and so neither iterative refinement nor error analysis are performed. If the column ordering is provided by the user (ICNTL(7)=1), then ICNTL(6) is treated as 0.

ICNTL(7) has default value 0 and must be set by the user to 1 if the column order in IS is to be used. For any other value of ICNTL(7), a suitable column order will be chosen automatically.

ICNTL(8) has default value 0 and is used to describe the format of the input matrix. Possible values of ICNTL(8) are:

- 0 the input matrix is given in coordinate form: the user must set the arrays IRN, JCN.
- 1 the input matrix is given by rows: the user must set the arrays JCN, IPTR. The IRN array is not accessed unless there are out-of-range entries or ICNTL(6)=1.

Other values of ICNTL(8) provoke an immediate error return with INFO(1)=–12.

ICNTL(9) has default value 0 and is accessed on a call that includes the factorization phase (JOB = 2, 4, 5, or 6). If ICNTL(9)=1, the **Q** factor is discarded. In this case, the Q array is not accessed and can be declared to have size 1. Any other value of ICNTL(9) is treated as 0.

ICNTL(10) has default value 0. If ICNTL(10)=1, Level 3 BLAS kernels will be used during the factorization. Any other value of ICNTL(10) is treated as 0.

ICNTL(11) has default value 0. This control is only operational when solving least-squares problems (ICNTL(13)=1 or 2). A positive value for ICNTL(11) will return the infinity norm of the residual, the infinity norm of the input matrix, the infinity norm of the computed solution, and an estimate of the backward error in RINFO(3), RINFO(4), RINFO(5), and RINFO(6), respectively, together with an estimate of the condition number and an estimate of the forward error in RINFO(7) and RINFO(8), respectively (see Section 2.4 for more details). If ICNTL(11) ≤ 0, the RINFO components are set to zero.

ICNTL(12) has default value 0. It corresponds to the maximum number of steps of iterative refinement. If

ICNTL(12) > 0, the infinity norm of the residual, the infinity norm of the input matrix, the infinity norm of the computed solution, and an estimate of the backward error will be returned in RINFO(3), RINFO(4), RINFO(5), and RINFO(6), respectively (see Section 2.4 for more details). Usually, a single step of iterative refinement suffices for getting an improved solution with a small backward error. If ICNTL(12) ≤ 0, iterative refinement is not performed and the RINFO components are set to zero.

ICNTL(13) has default value 1 and is only accessed on a call that includes a solve phase (JOB = 3, 5, or 6). If ICNTL(13) < 1 or ICNTL(13) > 8, MA49A/AD returns immediately with the error INFO(1)=–12. Possible values of ICNTL(13) are:

1 or 2 the least-squares problem, $\min \|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2$, is solved. The vector **b** should be of size *m* and **x** of size *n*.

1 the least-squares solution is computed by solving $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{Q}^{\mathrm{T}}\mathbf{b}$.

2 the least-squares solution is computed by solving $\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{x}=\mathbf{A}^{\mathrm{T}}\mathbf{b}$.

3 or 4 the minimum 2-norm solution of $\mathbf{A}^{\mathrm{T}}\mathbf{x}=\mathbf{b}$ is computed. The vector **b** should be of size *n* and **x** of size *m*.

3 the minimum 2-norm solution is computed by solving $\begin{bmatrix}\mathbf{R}^{\mathrm{T}}\mathbf{0}\end{bmatrix}\mathbf{z}=\mathbf{b}$ and performing the multiplication $\mathbf{x}=\mathbf{Q}\mathbf{z}$.

4 the minimum 2-norm solution is computed by solving $\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{z}=\mathbf{b}$ and performing the multiplication $\mathbf{x}=\mathbf{A}\mathbf{z}$.

5 or 6 the solution of systems with the matrix **R** are performed

5 the solution of $\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ is computed;

6 the solution of $\begin{bmatrix}\mathbf{R}^{\mathrm{T}}\mathbf{0}\end{bmatrix}\mathbf{x}=\mathbf{b}$ is computed;

7 or 8 the product of a vector by the orthogonal matrix **Q** or $\mathbf{Q}^{\mathrm{T}}$ is performed.

7 the matrix-vector multiplication **Qb** is performed and **b** is overwritten by the result.

8 the matrix-vector multiplication $\mathbf{Q}^{\mathrm{T}}\mathbf{b}$ is performed and **b** is overwritten by the result.

ICNTL(14) has default value 1 and is equal to the number of processors. Values of ICNTL(14) ≤ 0 are treated as 1.

ICNTL(15) to ICNTL(30) are set to zero by MA49I/ID but are not currently used by MA49A/AD.

INFO(1) is zero if the routine is successful, is negative if an error occurred, and is positive for a warning (see Section 2.3).

INFO(2) holds additional information concerning the error (see Section 2.3).

Statistics produced after analysis phase

INFO(3) If ICNTL(6)=1 and if the matrix is reducible, it is set equal to 1. Otherwise, it is set equal to 0.

If INFO(3)=1 (for definition of terms see Section 2.5),

INFO(4) number of rows in the horizontal (underdetermined) block.

INFO(5) number of columns in the horizontal (underdetermined) block.

INFO(6) number of connected components in the horizontal (underdetermined) block.

INFO(7) number of rows (and columns) in the square (exactly determined) block.

INFO(8) number of strong components in the square (exactly determined) block.

INFO(9) number of rows in the vertical (overdetermined) block.

INFO(10) number of columns in the vertical (overdetermined) block.

INFO(11) number of connected components in the vertical (overdetermined) block.

INFO(12) number of non-zero entries in the off-diagonal blocks.

INFO(13) Number of nodes in the assembly tree.

INFO(14) Estimated number of entries in the **R** factor (an upper bound).

INFO(15) Estimated maximum number of entries in the frontal matrices.

INFO(16) Minimum value of MAXIS estimated by the analysis phase to run the numerical factorization and solve phases. Used by the factorization and solve phases in the event of an error with INFO(1)=–6.

INFO(17) Minimum value of MAXS estimated by the analysis phase to run the numerical factorization and solve phases in a multiprocessor environment.

INFO(18) Minimum value of MAXS estimated by the analysis phase to run the numerical factorization and solve phases in a uniprocessor environment. Used by the factorization and solve phases in the event of an error with INFO(1)=–7.

INFO(19) Estimated number of reals used to store the **R** factor.

INFO(20) Estimated number of integers used to store the **R** factor.

INFO(21) Estimated size (in real words) used to store the **Q** factor. Used by the factorization and solve phases in the event of an error with INFO(1)=–8.

INFO(22) Number of entries in the $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ matrix.

INFO(23) Estimated value of MAXQ to run the numerical factorization and solve phases.

INFO(24) Estimated maximum order of the frontal matrices.

INFO(25) Minimum value of MAXIS for the analysis phase.

INFO(26) Not used in current version of the code.

Statistics produced after factorization

INFO(27) Total number of Householder transformations.

INFO(28) Maximum order of the frontal matrices.

INFO(29) Number of entries in the **R** factor.

INFO(30) Size of the real space used to store the **R** factor.

INFO(31) Size of the integer space used to store the **R** factor.

INFO(32) Actual space used in the Q array (minimum value of MAXQ).

INFO(33) Number of memory compresses on the real data structures in the S array.

INFO(34) Number of processors used to run the factorization.

INFO(35) Not used in current version of the code.

Statistics produced after solution

INFO(36) Number of steps of iterative refinement.

Warning information

INFO(37) Associated with the warning message +1, number of out-of-range indices.

INFO(38) Associated with the warning message +2, number of null rows.

INFO(39) Associated with the warning message +16, additional real space needed to run the code on ICNTL(14) processors.

INFO(40) to INFO(50) are not currently used by MA49A/AD.

Statistics produced after factorization

RINFO(1) Number of floating-point operations performed during the factorization.

RINFO(2) Not used.

Statistics produced after the solution of the least-squares problem

RINFO(3) to RINFO(14) are initialized to zero. RINFO(3), RINFO(4), RINFO(5), RINFO(6), are only meaningful if ICNTL(12) > 0 or ICNTL(11) > 0. If ICNTL(11) > 0, then RINFO(7) and RINFO(8) have meaningful data. See Section 2.4 for further information.

RINFO(3) Infinity norm of the residual **r**, where

$$\|\mathbf{r}\|_\infty = \max_{i=1,m} |(\mathbf{b}-\mathbf{A}\mathbf{x})_i|.$$

RINFO(4) Infinity norm of the input matrix, where

$$\|\mathbf{A}\|_\infty = \max_{i=1,m} \sum_{j=1}^{n} |a_{ij}|.$$

RINFO(5) Infinity norm of the computed solution, where

$$\|\mathbf{x}\|_\infty = \max_{i=1,n} |x_i|.$$

RINFO(6) Estimate of the backward error.

RINFO(7) Condition number of matrix **A**.

RINFO(8) Estimate of the forward error.

RINFO(9) to RINFO(14) contain additional information on the error analysis and are set only if ICNTL(11) ≥ 0.

RINFO(15) to RINFO(20) are not currently used by MA49A/AD.

2.3 Error diagnostics

A successful return from MA49A/AD is indicated by a value of INFO(1) equal to zero.

A negative value of INFO(1) is associated with an error message which will be optionally output on unit ICNTL(1). Possible negative values for INFO(1) are given below.

–1 Value of N is out-of-range. Value given is held in INFO(2).

–2 Value of M less than N. Value given for M is held in INFO(2).

–3 Value of NE is out-of-range. Value given is held in INFO(2).

–4 JOB has an incorrect value or analysis was not performed prior to the factorization, or factorization was not performed prior to the solve. Value given is held in INFO(2).

–5 Failure due to insufficient integer space. Error during the analysis phase. MAXIS must be increased at least to the value held in INFO(25). Input matrix must be regenerated and analysis performed again.

–6 Failure due to insufficient integer space. Error during factorization or solve phases. MAXIS must be increased at least to the value held in INFO(16). If INFO(2)=1, error occurred during factorization phase. Real values of the input matrix must be regenerated and factorization performed again. If INFO(2)=2, error occurred during solve phase. Real values of the right-hand side must be regenerated and solve performed again.

–7 Failure due to insufficient space in the S array. Error during factorization or solve phases. MAXS must be increased to at least the value held in INFO(18). If INFO(2)=1, error occurred during factorization phase. Real values of the input matrix must be regenerated and factorization performed again. If INFO(2)=2, error occurred during solve phase. Real values of the right-hand side must be regenerated and solve performed again.

–8 Failure due to insufficient space in the Q array. Error during factorization or solve phases. MAXQ must be increased to at least the value held in INFO(21). If INFO(2)=1, error occurred during factorization phase. Real values of the input matrix must be regenerated and factorization performed again. If INFO(2)=2, error occurred during solve phase. Real values of the right-hand side must be regenerated and solve performed again.

–9 Error return from analysis phase. The matrix has null columns and is therefore structurally rank deficient.

–10 Error return from analysis or numerical factorization phases. The matrix has more than M – N null rows, and is therefore structurally rank deficient.

–11 Error return from numerical factorization. The matrix is numerically singular. The matrix will be declared singular if the norm of a column in the reduced matrix is less than or equal to the value of CNTL(1). INFO(2) holds the index of the frontal matrix when the singularity was detected.

–12 Some values of the ICNTL parameter are wrong or incompatible with other values. INFO(2) contains additional information.

INFO(2)= 1 Error return from analysis phase. ICNTL(8) has a value out-of-range.

INFO(2)= 2 Error return from factorization phase. The input matrix has been permuted to block upper triangular form (ICNTL(6)=1) but the **Q** factor has been discarded (ICNTL(9)=1).

INFO(2)= 3 Error return from solve phase. The **Q** factor has been discarded (ICNTL(9)=1) and a

computation with this factor has been requested (ICNTL(13)=1, 3, 7, or 8).

INFO(2)= 4 Error return from solve phase. The input matrix has been permuted to block upper triangular form (ICNTL(6)=1) and ICNTL(13) ≠ 1.

INFO(2)= 5 Error return from solve phase. ICNTL(13) out of range.

A positive value of INFO(1) is associated with a warning message which will be output on unit ICNTL(2). Possible positive values for INFO(1) are given below.

+1 Index (in IRN or JCN) out-of-range. Action taken by the subroutine is to ignore any such entries and continue. INFO(37) is set to the number of faulty entries. Details of the first ten are printed on unit ICNTL(2).

+2 During analysis, null rows have been found. INFO(38) is set to the number of null rows.

+4 During analysis, the block upper triangular option has been invoked (ICNTL(6)=1) and the column order in IS is to be used (ICNTL(7)). ICNTL(6) is considered as zero.

+8 Real working array Q might be too small to run the numerical factorization. In fact, the estimated space to store the **Q** factor (INFO(21)) is usually a large upper bound of the space actually needed.

+16 Real working array S used during numerical factorization is not large enough to be able to run on ICNTL(14) processors. The code is run on INFO(34) processors. MAXS must be increased to at least INFO(39) to be able to run on ICNTL(14) processors.

+32 During error analysis, the max-norm of the computed solution was found to be zero.

+64 Warning return from the iterative refinement routine. More than ICNTL(12) iterations are required.

+x x =3, ..., 127 are combinations of the above warnings corresponding to summing the constituent warnings.

2.4 Error estimates and iterative refinement

We calculate an estimate of the sparse backward error for a computed least-squares solution using the error analysis given in Arioli, Duff, and de Rijk (1989) and Björck (1991), and the theory and measure for sparse error developed by Arioli, Demmel, and Duff (1989).
The computed solution is the exact solution of a perturbed problem where the size of the perturbation is given by the component-wise relative backward error (RINFO(6)). If RINFO(6) is of the order of the machine precision, we are guaranteed that the computed solution is the exact solution of a small perturbation of the problem. The constituent parts of the backward error are returned in RINFO(9) to RINFO(12). The reader should consult the references for further information on these.
An appropriate condition number for **A** is also computed and is returned in RINFO(7). The constituent parts of the condition number are returned in RINFO(13) and RINFO(14). An upper bound for the forward error, computed from the backward error and the condition number, is returned in RINFO(8).

Iterative refinment can be used to improve the solution. The user must set a maximum number, but the iterations will cease if the estimated backward error is at rounding level, or if it does not decrease sufficiently (by at least the factor set in CNTL(2)).

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.

Arioli, M. Duff, I. S., and de Rijk, P.P.M. (1989). On the augmented system approach to sparse least-squares problems. *Numer. Math.* **55**, 667-684.

Björck Å. (1991). Component-wise perturbation analysis and error bounds for linear least squares solutions. *BIT* **31**, 238-244.

2.5 Permutation to block triangular form

We offer the user the option of permuting the input matrix to block upper triangular form. This option is controlled by ICNTL(6) and can be used only if the least-squares solution is to be computed using Golub's method. If the matrix is reducible, it will be explicitly permuted to block upper triangular form. The diagonal blocks will be stored by rows and the off-diagonal blocks by columns, using the arrays IPTR and JPTR.

IPTR(K) holds the pointer to the first entry in JCN of row K of the diagonal blocks, K=1, ... M and IPTR(M+1) the position after the last entry of the diagonal blocks. (Entries in heavily shaded region in figure.)

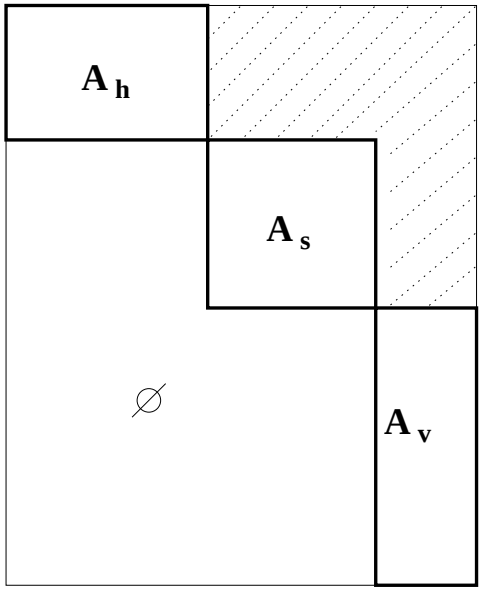
JPTR(K) holds the pointer to the first entry in JCN of column K of the off-diagonal blocks, K=1, ... N and IPTR(N+1) the position after the last entry of the diagonal blocks. (Entries in lightly shaded region in figure.)

JCN(1:NEBTF), where NEBTF=NE–INFO(12), holds the diagonal blocks by rows, and JCN(NEBTF+1:NE) the off-diagonal blocks by columns.

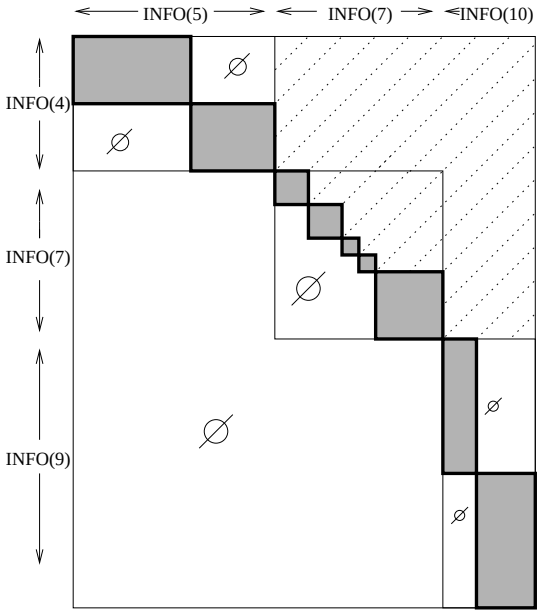
The algorithm used to perform this ordering is GENBTF from Alex Pothen of Old Dominion University, North Carolina. It uses the algorithm described by Pothen and Fan (1990). The reordered matrix is of the form:

Reference

Pothen and Fan, (1990). Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.* **16**, 303-324,



Coarse decomposition



Fine decomposition

INFO(6) = 2
INFO(8) = 5
INFO(11) = 2

3 GENERAL INFORMATION

Use of common: None.

Workspace: Parts of the arrays IS, S and Q are used as workspace by the routines.

Other routines called directly:

BLAS **routines:** I_AMAX, _AXPY, _COPY, _DOT, _GEMM, _GER, _GEMV, _NRM2, _SYRK, _SWAP, _TRSM, _TRSV.

HSL routines: MC41A/AD, MC47B/BD.

Internal routines: GENBTF (Pothen).

All the other subroutines called by the principal subroutines are in the MA49 or the MA59 package. They are: MA49B/BD, MA49C/CD, MA49D/DD, MA49E/ED, MA49F/FD, MA49G/GD, MA49H/HD, MA49J/JD, MA49K/KD, MA49L/LD, MA49M/MD, MA49N/ND, MA49O/OD, MA49P/PD, MA49Q/QD, MA49R/RD, MA49S/SD, MA49T/TD, MA49U/UD, MA49V/VD, MA49W/WD, MA49X/XD, MA49Y/YD. MA49Z/ZD, although at present MA49Z/ZD is a dummy routine that might be used in a later version of the code.
Those in the MA59 package are:

MA59A/AD, MA59B/BD, MA59C/CD, MA59D/DD, MA59E/ED, MA59F/FD, MA59G/GD, MA59H/HD, MA59I/ID, MA59J/JD, MA59K/KD, MA59L/LD, MA59M/MD, MA59N/ND, MA59O/OD, MA59P/PD, MA59Q/QD, MA59R/RD, MA59S/SD, MA59T/TD, MA59U/UD, MA59V/VD, MA59W/WD, MA59X/XD, MA59Y/YD, MA59Z/ZD.

Input/output: Error, warning and monitoring messages only. Error messages on unit ICNTL(1), which has default value 6. Warning and monitoring messages can optionally be printed on unit ICNTL(2) and ICNTL(3), respectively, which have default value 6. Messages can be suppressed by setting ICNTL(I), I=1,2,3 negative. Messages are also suppressed if ICNTL(5) is set to zero. Statistics on the decomposition can be optionally printed on unit ICNTL(4).

Restrictions: N ≥ 1, M ≥ N, NE ≥ 1.

4 METHOD

A parallel version of a multifrontal orthogonal factorization is used. Householder reflections are used.

The analysis phase forms the matrix $\mathbf{A}^T\mathbf{A}$ symbolically and orders it using the Approximate Minimum Degree algorithm of Amestoy, Davis and Duff (1996) which returns a column ordering for **A**. The elimination is represented as an assembly tree.

The factorization phase factorizes the matrix by using the assembly tree generated by the analysis phase. At each stage in the multifrontal approach, elimination is performed on full submatrices. The parallelism of the assembly tree is exploited and when the tree does not provide any more parallelism, the parallelism coming from the frontal matrix is used. Thus this phase can be used to factorize general square/overdetermined matrices and will perform well on shared memory parallel computers. The number of parallel processes used during factorization can be explicitly controlled by the user.

The solve phase uses the factors from the factorization phase to solve systems of equations by loading the appropriate parts of the vectors into an array of the same size as the current frontal matrix and using full matrix code. In contrast to the factorization step, only the parallelism of the assembly tree is exploited.

In addition, MA49A/AD offers preprocessing and post-processing facilities. Permutations to block upper triangular form (Pothen and Fan, 1990) can be applied to reducible systems and can help to reduce computation time and storage and thus solve larger problems. Iterative refinement can be optionally performed after the solution step. Finally MA49A/AD also enables the user to perform error analysis.

If the **Q** factor is held, it is not done so explicitly, rather the set of Householder transformations performed on the frontal matrices are stored. In this way, the operations of **Q** or $\mathbf{Q}^T$ can be performed. It is possible to discard the **Q** factor. In this case, the least-squares problem $\min \|\mathbf{b}-\mathbf{Ax}\|_2$ can be solved by using the seminormal equations method ($\mathbf{R}^T\mathbf{Rx} = \mathbf{A}^T\mathbf{b}$)

The multifrontal approach is discussed in detail by Duff and Reid (1983, 1984), and Duff (1986). The tuned multiprocessor version of the code described here is based on work by Duff (1986), Amestoy and Duff (1989, 1993), Puglisi (1993) and Amestoy, Duff and Puglisi (1996).

Reference

Amestoy, P. R. Davis, T. A., and Duff, I. S. (1996). The Approximate Minimum Degree algorithm. *SIAM J. Matrix Anal. and Applics.* **17**, 886-905.

Amestoy, P. R. Duff, I. S., and Puglisi, C. (1996). Multifrontal *QR* factorization in a multiprocessor environment. *Num. Lin. Alg. with Applics.* **3**, (4), 275-300.

Amestoy, P. R. and Duff, I. S. (1989). Vectorization of a multiprocessor multifrontal code. *Int. J. Supercomputer Applics.* **3**, (3), 41-59.

Amestoy, P. R. and Duff, I. S. (1993). Memory allocation issues in sparse multiprocessor multifrontal methods. *Int. J. Supercomputer Applics.* **7**, 64-82.

Duff, I.S. and Reid, J. K. (1983). The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Trans. Math. Softw.* **9**, 302-325.

Duff, I.S. and Reid, J. K. (1984). The multifrontal solution of unsymmetric sets of linear systems. *SIAM J. Sci. Stat.*

Comput. **5**, 633-641.

Duff, I.S. (1986). Parallel implementation of multifrontal schemes. *Parallel Computing* **3**, 193-204.

Puglisi, C. (1993). *QR* Factorization of large sparse overdetermined and square matrices using a multifrontal method in a multiprocessor environment. CERFACS Report TH/PA/93/33

Pothen and Fan, (1990). Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.* **16**, 303-324,

5 EXAMPLES OF USE

The use of the package is illustrated on the solution of the single set of equations

$$\begin{pmatrix} 2 & 3 & 4 & & \\ 3 & & -3 & & 6 \\ & -1 & 1 & 2 & \\ & & 2 & & \\ & 4 & & & 1 \\ & & 3 & 3 & \\ & & & & 2 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 20 \\ 24 \\ 9 \\ 6 \\ 13 \\ 21 \\ 10 \end{pmatrix} .$$

The driver `MA49A/AD` is used with `JOB=6` and the solution and error estimates are printed. Note that this example does not illustrate all the facilities.

Program

```
C
C Simple example of use of MA49 Package
C
      PROGRAM MA49ST

C Set array dimensions
C NMAX   = Maximum number of columns of matrix
C MMAX   = Maximum number of rows of matrix
C NZMAX  = Maximum number of entries in matrix
C MAXS   = Maximum size of the REAL working space S
C MAXQ   = Maximum size of the REAL working space Q
C MAXIS  = Maximum size of the INTEGER working space
      INTEGER NMAX, MMAX, NZMAX, MAXS, MAXQ, MAXIS
      PARAMETER (NMAX=10, MMAX=10, NZMAX=20, MAXS=800,
     *           MAXQ=900, MAXIS=800)

C Declare MA49 parameters
      INTEGER JOB, M, N, NE, IRN(NZMAX), JCN(NZMAX)
      DOUBLE PRECISION A(NZMAX), B(MMAX), X(NMAX)
      INTEGER KEEP(50), IS(MAXIS), JPTR(1), IPTR(MMAX)
      DOUBLE PRECISION S(MAXS), Q(MAXQ), CNTL(10)
      INTEGER ICNTL(30), INFO(50)
      DOUBLE PRECISION RINFO(20)

C Local variables
      INTEGER I, INFILE, LP

C Set input and output streams
      INFILE = 5
      LP     = 6

C Read matrix and right-hand side
      READ (INFILE,*) N, M, NE
      READ (INFILE,*) (IRN(I),I=1,NE)
      READ (INFILE,*) (JCN(I),I=1,NE)
      READ (INFILE,*) (A(I),  I=1,NE)
      READ (INFILE,*) (B(I),  I=1,M)

C-----------------------------------
C Initialization of control parameters
C-----------------------------------
```

```
      CALL MA49ID (CNTL, ICNTL, KEEP)
C Input matrix by coordinates.
      ICNTL(8)  = 0
C Request error analysis and iterative refinement
      ICNTL(11) = 1
      ICNTL(12) = 2

C-----------------------------------------------------
C Analyse sparsity pattern using approximate minimum
C degree ordering.
C Solve the least-squares problem with Golub's method,
C and perform error analysis.
C-----------------------------------------------------
      JOB = 6
      CALL MA49AD(JOB, M, N, NE, IRN, JCN, IPTR, JPTR,
     &            A, B, X, KEEP, IS, MAXIS, S, MAXS,
     &            Q, MAXQ, CNTL, ICNTL, INFO, RINFO)
C Print error and residual
      WRITE(LP,'(A/(10F8.4))') 'Solution is:',(X(I),I=1,N)
      WRITE(LP,'(A,F8.4)') 'Forward error is  ',RINFO(7)
      WRITE(LP,'(A,F8.4)') 'Backward error is ',RINFO(8)

C Error if INFO(1) is negative
      IF (INFO(1).LT.0) WRITE(LP,'(A)') ' Error in MA49AD'

      STOP
      END
```

Data

```
   5    7   15
   1    2    4    6    5    2    7    1    5    3    6    2    3    1    3
   2    3    3    3    5    1    5    1    2    4    4    5    2    3    3
 3.0 -3.0  2.0  3.0  1.0  3.0  2.0  2.0  4.0  2.0  3.0  6.0 -1.0  4.0  1.0
20.0 24.0  9.0  6.0 13.0 21.0 10.0
```

Output

```
Solution is:
  1.0000  2.0000  3.0000  4.0000  5.0000
Forward error is       0.0000
Backward error is      0.0000
```