1 SUMMARY

This package **solves a sparse unsymmetric system of n linear equations** in n unknowns using an unsymmetric multifrontal variant of Gaussian elimination. There are facilities for choosing a good pivot order, factorizing another matrix with a nonzero pattern identical to that of a previously factorized matrix, and solving a system of equations using the factorized matrix. An option exists for solving triangular systems using the factors from the Gaussian elimination.

ATTRIBUTES — Version: 1.1.0. **Types:** Real (single, double). **Calls:** `FD15`, `MC13`, `MC21`, `_GEMV`, `I_AMAX`, `_GEMM`, `_TRSV`, `_TRSM`. **Original date:** October 1995. **Origin:** T. A. Davis, University of Florida, and I. S. Duff, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Overview of user-callable routines

There are four routines that can be called by the user:

(a) `MA38I/ID` must be called to set default values for the control parameters `CNTL` and `ICNTL`, and values in the array `KEEP`. It would normally be called once prior to any calls to the other `MA38` routines.

(b) `MA38A/AD` computes the pivot ordering and performs both symbolic and numerical **LU** factorization of an $n \times n$ unsymmetric sparse matrix. It optionally permutes the matrix **A** to block upper triangular form, and the user can choose to factorize either the matrix or its transpose. The pivot selection is based on a Markowitz-style ordering, with upper bounds on the number of entries in each row and column (exact counts are not computed). The pivoting strategy is a tradeoff between maintaining sparsity and controlling loss of accuracy through roundoff. A parameter is available for controlling this balance. There is an option for the user to set a preference for pivoting on the diagonal. The numerical factorization is based on the factorization of rectangular frontal matrices, using dense matrix kernels.

If **A** is not permuted to block upper triangular form, the factorization is **PAQ**=**LU**, where **P** and **Q** are row and column permutations needed for pivoting during factorization, **L** is lower triangular with a unit diagonal, and **U** is upper triangular. If **A** is permuted to block upper triangular form, then each diagonal block is factorized separately. Off-diagonal blocks are not affected. The final permutation matrices **P** and **Q** include the permutation to block upper triangular form and the permutations due to pivoting during the factorization of the diagonal blocks.

(c) `MA38B/BD` factorizes a matrix **A** that has the same nonzero pattern as a matrix previously factorized by `MA38A/AD`. It uses the same pivot order as found by `MA38A/AD`. This routine is generally significantly faster than `MA38A/AD`.

(d) `MA38C/CD` uses the factors produced by `MA38A/AD` or `MA38B/BD` to solve $\mathbf{Ax}=\mathbf{b}$ or $\mathbf{A}^T\mathbf{x}=\mathbf{b}$, optionally performing iterative refinement. An option exists to solve triangular systems whose coefficient matrix is the factor **L**, **U**, $\mathbf{L}^T$, or $\mathbf{U}^T$.

2.1.1 To set default values of controlling parameters

`MA38I/ID` sets default values for the components of the arrays that hold control parameters for the factorization and solution routines.

The single precision version

```
CALL MA38I(KEEP,CNTL,ICNTL)
```

The double precision version

```
CALL MA38ID(KEEP,CNTL,ICNTL)
```

KEEP is an INTEGER array of length 20 that need not be set by the user. Some of its components are given values by MA38I/ID. Its contents are not of concern to the user but it must be preserved between calls to the subroutines in this package.

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 (see Section 2.2 for details).

ICNTL is an INTEGER array of length 20 that need not be set by the user. On return it contains default values (see Section 2.2 for details).

2.1.2 To factorize a sparse matrix

MA38A/AD computes an **LU** factorization of a unsymmetric sparse matrix **A**. The matrix is optionally preordered to block upper triangular form. Pivoting within each diagonal block is performed during factorization to maintain sparsity and numerical stability.

The single precision version:

```
CALL MA38A (N, NE, JOB, TRANSA, LVALUE, LINDEX, VALUE, INDEX, KEEP,
            CNTL, ICNTL, INFO, RINFO)
```

The double precision version:

```
CALL MA38AD (N, NE, JOB, TRANSA, LVALUE, LINDEX, VALUE, INDEX, KEEP,
             CNTL, ICNTL, INFO, RINFO)
```

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

JOB is an INTEGER variable that must be set by the user to indicate whether an ordered form of the input matrix is preserved (JOB=1) or not (JOB≠1). If the iterative refinement option (ICNTL(8)>0) is to be used by MA38C/CD, then JOB must be set to 1. JOB is not altered by the subroutine.

TRANSA is a LOGICAL variable that must be set by the user. If TRANSA is .TRUE., the matrix $\mathbf{A}^T$ is factorized ($\mathbf{PA}^T\mathbf{Q}=\mathbf{LU}$). Otherwise, the matrix **A** is factorized (**PAQ**=**LU**). TRANSA is not altered by the subroutine.

LVALUE is an INTEGER variable that must be set by the user to the length of array VALUE. LVALUE is not altered by the subroutine. **Restriction:** LVALUE≥2*NE.

LINDEX is an INTEGER variable that must be set by the user to the length of array INDEX. LINDEX is not altered by the subroutine. **Restriction:** LINDEX≥3*NE+2*N+1.

VALUE is a REAL (DOUBLE PRECISION in the D version) array of length LVALUE. VALUE(K), K=1, ..., NE must be set by the user to hold the values of the matrix entries. They may be in any order. If there is more than one entry for a particular position, the values are accumulated. The number of such multiple entries is returned in INFO(2) (see Section 2.2). On exit, VALUE will hold information on the matrix factors and, if JOB=1, a reordered copy of the original matrix. It must be passed unchanged to subsequent calls to MA38B/BD and MA38C/CD.

INDEX is an INTEGER array of length LINDEX that must be set by the user so that entry VALUE(K), K=1, ..., NE of the matrix **A** is in row INDEX(K), column INDEX(NE+K). Entries with row or column indices outside the range 1 to n are ignored by the routine and their number is recorded in INFO(3) (see Section 2.2). On exit, INDEX will hold integer information on the matrix factors and, if JOB=1, integer information on a reordered copy of the

original matrix. It must be passed unchanged to subsequent calls to MA38B/BD and MA38C/CD.

KEEP is an INTEGER array of length 20. KEEP should be preserved between calls to MA38A/AD and subsequent calls to MA38B/BD or MA38C/CD. KEEP must be unchanged since the previous call to MA38I/ID.

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 are set by a call to MA38I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

ICNTL is an INTEGER array of length 20 that contains control parameters and must be set by the user. Default values for the components are set by a call to MA38I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

INFO is an INTEGER array of length 40 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MA38A/AD, a non-negative value for INFO(1) indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.1. For details of the information output in the 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. On output, RINFO holds information on the factorization (see Section 2.2).

2.1.3 To factorize a subsequent sparse matrix

MA38B/BD factorizes a matrix of the same nonzero pattern as a matrix previously factorized by MA38A/AD. No variations are made in the pivot order computed by MA38A/AD.

The single precision version:

```
CALL MA38B (N, NE, JOB, TRANSA, LVALUE, LINDEX, VALUE, INDEX, KEEP,
            CNTL, ICNTL, INFO, RINFO)
```

The double precision version:

```
CALL MA38BD (N, NE, JOB, TRANSA, LVALUE, LINDEX, VALUE, INDEX, KEEP,
             CNTL, ICNTL, INFO, RINFO)
```

N is an INTEGER variable that must be unchanged since the previous call to MA38A/AD. It is not altered by the subroutine.

NE is an INTEGER variable that must be set by the user to the number of entries in the matrix to be factorized. It is not altered by the subroutine. **Restriction:** NE ≤ KEEP(4)/2.

JOB is an INTEGER variable that must be set by the user to indicate whether an ordered form of the input matrix is preserved (JOB=1) or not (JOB≠1). If the iterative refinement option (ICNTL(8)>0) is to be used by MA38C/CD, then JOB must be set to 1. JOB is not altered by the subroutine.

TRANSA is a LOGICAL variable that is normally unchanged since the previous call to MA38A/AD. It is not altered by the subroutine.

LVALUE, LINDEX are INTEGER variables that must be set to the lengths of arrays VALUE and INDEX, respectively. They are not altered by the subroutine.

VALUE, INDEX are as in the call to MA38A/AD but now hold the values and indices of the new matrix **A** to be factorized. The user must not change any other entries in these arrays after the previous call to MA38A/AD or MA38B/BD.

KEEP is as in the call to MA38A/AD. It must be unchanged from the previous call to MA38A/AD.

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 are set by a call to MA38I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

ICNTL is an INTEGER array of length 20 that contains control parameters and must be set by the user. Default values for the components are set by a call to MA38I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

INFO is an INTEGER array of length 40 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MA38B/BD, a non-negative value for INFO(1) indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.2. For details of the information output in the 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. On output, RINFO holds information on the factorization (see Section 2.2).

2.1.4 To solve equations Ax=b or A^Tx=b or systems using the factors L, U, L^T, or U^T

Given **LU** factors computed by MA38A/AD or MA38B/BD, and the right-hand-side, **b**, MA38C/CD computes the solution, **x**. This routine handles all permutations, so that **b** and **x** are in terms of the original matrix **A** and not in terms of the permuted matrix. Options exist for solving triangular systems with one of the factors and for performing iterative refinement.

This routine solves systems of equations with a singular matrix by replacing trailing zero blocks of the factors by the identity matrix.

The single precision version:

```
CALL MA38C (N, JOB, TRANSC, LVALUE, LINDEX, VALUE, INDEX, KEEP,
            B, X, W, CNTL, ICNTL, INFO, RINFO)
```

The double precision version:

```
CALL MA38CD (N, JOB, TRANSC, LVALUE, LINDEX, VALUE, INDEX, KEEP,
             B, X, W, CNTL, ICNTL, INFO, RINFO)
```

N is an INTEGER variable that must be unchanged since the previous call to MA38A/AD. It is not altered by the subroutine.

JOB is an INTEGER variable that must be set by the user to determine which system to solve (see parameter TRANSC). This parameter is not altered by MA38C/CD.

TRANSC is a LOGICAL variable that must be set by the user. The parameters JOB, TRANSA, and TRANSC determine which system is solved viz.

If TRANSA and TRANSC are false, then **PAQ=LU** was performed, and the system solved is:

JOB=0 $\mathbf{Ax}=\mathbf{b}$

JOB=1 $\mathbf{P}^T\mathbf{Lx}=\mathbf{b}$

JOB=2 $\mathbf{UQ}^T\mathbf{x}=\mathbf{b}$

If TRANSA is false and TRANSC is true, then **PAQ=LU** was performed, and the system solved is:

JOB=0 $\mathbf{A}^T\mathbf{x}=\mathbf{b}$

JOB=1 $\mathbf{L}^T\mathbf{Px}=\mathbf{b}$

JOB=2 $\mathbf{QU}^T\mathbf{x}=\mathbf{b}$

If TRANSA is true and TRANSC is false, then $\mathbf{PA}^T\mathbf{Q}=\mathbf{LU}$ was performed, and the system solved is:

JOB=0 $\mathbf{A}^T\mathbf{x}=\mathbf{b}$

JOB=1 $\mathbf{P}^T\mathbf{Lx}=\mathbf{b}$

JOB=2 $\mathbf{UQ}^T\mathbf{x}=\mathbf{b}$

If TRANSA is true and TRANSC is true, then $\mathbf{PA}^T\mathbf{Q}=\mathbf{LU}$ was performed, and the system solved is:

JOB=0 $\mathbf{Ax}=\mathbf{b}$

JOB=1 $\mathbf{L}^T\mathbf{Px}=\mathbf{b}$

JOB=2 $\mathbf{QU}^T\mathbf{x}=\mathbf{b}$

This parameter is not altered by MA38C/CD. Iterative refinement can be done only when JOB is 0.

LVALUE, LINDEX are INTEGER variables that must be unchanged since the previous call to MA38A/AD. They are not altered by the subroutine.

VALUE, INDEX are as in the call to MA38A/AD and now hold information on the matrix factorization. The user must not change these arrays after the previous call to MA38A/AD or MA38B/BD.

KEEP is as in the call to MA38A/AD. It must be unchanged since the last call to MA38A/AD or MA38B/BD.

B is a REAL (DOUBLE PRECISION in the D version) array of length N that must be set by the user to hold the right-hand side. It is not altered by the routine.

X is a REAL (DOUBLE PRECISION in the D version) array of length N that need not be set by the user. On exit, it holds the computed solution.

W is a REAL (DOUBLE PRECISION in the D version) array of length 2*N or 4*N that is used as workspace. If ICNTL(8)≥1, then the length of W must be 4*N, and W(I),I=1, ..., N will hold the residual vector on exit. Otherwise the length of W is 2*N.

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 are set by a call to MA38I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

ICNTL is an INTEGER array of length 20 that contains control parameters and must be set by the user. Default values for the components are set by a call to MA38I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

INFO is an INTEGER array of length 40 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MA38C/CD, a non-negative value for INFO(1) indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.3. For details of the information output in the 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. On output, RINFO holds information on the factorization (see Section 2.2).

2.2 Arrays for control and information

The components of the arrays CNTL and ICNTL control the action of MA38A/AD, MA38B/BD, and MA38C/CD. Default values for these components are set by MA38I/ID. The elements of the array INFO provide information on the action of MA38A/AD, MA38B/BD, and MA38C/CD. Note that the length of these four arrays are declared longer than the number of components shown below. Components beyond those shown below are not used in the present code (in fact they are set to zero) but are present to accommodate possible upgrades to this software.

CNTL(1) is used by MA38A/AD to control the relative numerical pivot tolerance. If zero, then no relative numerical test is made. If greater than zero, a pivot $a_{ij}^{[k]}$ must satisfy the threshold partial pivoting test: $|a_{ij}^{[k]}| \geq \texttt{CNTL(1)} * \max_i |a_{ij}^{[k]}|$ where the notation $a_{ij}^{[k]}$ refers to an entry in the partially factorized matrix just prior to step k of Gaussian elimination. A value of CNTL(1) less than zero is treated as zero, and a value greater than one is treated as one. Typical range: 0.001 to 0.1. Default value: 0.1.

CNTL(2) is used by MA38A/AD to control how much a frontal matrix can grow due to amalgamation (see discussion in Section 4.1). A value of 1.0 means that no fill-in due to amalgamation will occur. Some amalgamation is necessary for efficient use of the Level 3 BLAS. A value of CNTL(2) less than one is treated as one, and a value greater than N is treated as N. Typical range: 1.5 to 3.0. Default value: 2.0.

CNTL(3) is set to the machine epsilon by a call to FD15A/AD from MA38I/ID. Machine epsilon is the smallest number, ε such that $1+\varepsilon$ differs from 1. It would not normally be changed by the user. It is used in the termination test for the iterative refinement.

ICNTL(1) has default value 6 and holds the unit number to which the error and warning messages are sent. A negative value suppresses all messages.

ICNTL(2) has default value 6 and holds the unit number to which diagnostic printing is sent. A negative value suppresses all such printing.

ICNTL(3) is used by the subroutines to control printing of error, warning, and diagnostic 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 diagnostics (only first ten entries of arrays printed, and all duplicate and invalid entries).
- 4 Errors and warnings and most information on input and output parameters printed.
- 5 Errors and warnings and all information on input and output parameters printed.

ICNTL(4) is used to control block triangularization. If equal to 1, then **A** is preordered to block upper triangular form, using MC13E/ED and MC21B/BD. For values other than 1, block triangularization is not performed. Default value: 1.

ICNTL(5) is the maximum number of columns examined during a pivot search. A value of ICNTL(5) less than 1 is treated as 1, and a value greater than N is treated as N. Default value: 4.

ICNTL(6) is used to control pivoting. If equal to 1, then pivots on the diagonal of **A** are preferred over pivots off the diagonal. If **A** is preordered to block upper triangular form, then the diagonal of the permuted matrix is preferred. If 0, then no preference is made. Setting ICNTL(6) to 1 is useful for matrices that are symmetric in structure and diagonally dominant, since fill-in is often less if symmetry is preserved. This is only a preference in the sense that, when searching a column for a pivot, the diagonal entry is accepted if it passes the threshold test (see CNTL(1)), otherwise an off-diagonal in that column can still be chosen. Default value: 0.

ICNTL(7) is the block size for the numerical factorization of the dense frontal matrices. It controls the trade-off between Level 2 and Level 3 BLAS. The best value of ICNTL(7) depends on the computer being used. A value of ICNTL(7) less than one is treated as one, and a value greater than N is treated as N. Typical range: 16 to 64. Default value: 16 (this is good on a Cray YMP but may need to be changed for different computing platforms).

ICNTL(8) is the maximum number of steps of iterative refinement. When set to its default value of 0, no iterative refinement is performed. If ICNTL(8) $\geq$ 1, then the residual will be returned in entries 1 to N of W, and an estimate of the backward error in RINFO(7) and RINFO(8).

INFO(1) has the value zero if the call was successful, a positive value in the case of a warning, and a negative value in the event of an error (see Section 2.3).

INFO(2) is set to the number of duplicates found.

INFO(3) is the number of entries dropped from the original matrix because they have out-of-range indices.

INFO(4) is the number of entries dropped from the original matrix by MA38B/BD because they are not within the nonzero pattern of the **LU** factors of the matrix previously factorized by MA38A/AD or MA38B/BD.

INFO(5) is the number of entries in **A**, after removing duplicates and invalid entries.

INFO(6) is the number of entries in the block diagonal portion of **A**.

INFO(7) is the number of entries in the off-diagonal blocks (where INFO(5) = INFO(6)+INFO(7)).

INFO(8) is the number of 1-by-1 blocks in the block upper triangular form of **A**.

INFO(9) is the number of diagonal blocks in the block upper triangular form of **A**.

INFO(10) is the number of entries in the strictly lower triangular part of **L** (excluding the diagonal).

INFO(11) is the number of entries in the strictly upper triangular part of **U** (excluding the diagonal).

INFO(12) is the number of entries in **L** + **U** (where INFO(12) = INFO(10) + INFO(11) + N + INFO(7)).

INFO(13) is the number of frontal matrices.

INFO(14) is the number of garbage collections caused by insufficient space in INDEX. Garbage collections are performed on both INDEX and VALUE when the available space in either array is exhausted. If INFO(14) is excessively high, performance can be degraded. Try increasing the length of INDEX if that occurs (or try reducing fill-in using the suggestions listed in Section 2.3.1 under error –3).

INFO(15) is the number of garbage collections caused by insufficient space in VALUE. If INFO(15) is too high, performance can be degraded. Try increasing the length of VALUE if that occurs (or try reducing fill-in using the suggestions listed in Section 2.3.1 under error –3).

INFO(16) is the number of pivots selected on the diagonal of the original matrix **A**.

INFO(17) is the number of numerically acceptable pivots found during factorization. It is an estimate of the numerical rank of the matrix.

INFO(18) is the amount of space used in INDEX on a call to MA38A/AD.

INFO(19) is the minimum length of INDEX for MA38A/AD to succeed. However, many garbage collections might be required if the problem were rerun with LINDEX set to this value.

INFO(20) is the amount of space used in VALUE on a call to MA38A/AD.

INFO(21) is the minimum length of VALUE for MA38A/AD to succeed. However, many garbage collections might be required if the problem were rerun with LVALUE set to this value.

INFO(22) is the minimum length of INDEX required for a subsequent call to MA38B/BD. It is computed by both MA38B/BD and MA38A/AD.

INFO(23) is the minimum length of VALUE required for a subsequent call to MA38B/BD. It is computed by both MA38B/BD and MA38A/AD. Some garbage collections are likely if LVALUE is reduced to this amount. (Note that reducing LVALUE and/or LINDEX requires the user to shift the **LU** factors stored in INDEX, VALUE, and KEEP).

INFO(24) is the number of steps of iterative refinement performed.

RINFO(1) is the theoretical number of floating-point operations performed during **LU** factorization. This count does not take into consideration the floating-point operations skipped in the BLAS because of numerically zero entries in the frontal matrices, nor does it include the extra floating-point additions performed by the assembly phase. It is typically higher than the actual number of floating-point operations performed, but may be lower. (RINFO(1) = RINFO(4)+RINFO(5)+RINFO(6)).

RINFO(2) is the number of floating-point operations performed during the assembly operations.

RINFO(3) is the number of floating-point operations performed during pivot searches.

RINFO(4) is the number of floating-point operations in the numerical factorization performed using an in-line version of the Level 1 BLAS (_SCAL).

RINFO(5) is the number of floating-point operations in the numerical factorization performed using Level 2 BLAS (_GEMV and _TRSV).

RINFO(6) is the number of floating-point operations in the numerical factorization performed using Level 3 BLAS (_GEMM and _TRSM).

RINFO(7) and RINFO(8) are used to hold information on the backward error. See Section 2.4.

2.3 Error diagnostics

2.3.1 Error diagnostics for MA38A/AD

If the subroutine encounters no errors or warnings, the value of INFO(1) will be zero on exit from MA38A/AD. A positive value indicates a warning; a negative value indicates an error. The errors and warnings that can arise are given below. In each case an explanatory message is output on stream ICNTL(1).

–1 N has a value less than 1.

–2 NE has a value less than 1.

–3 Insufficient integer workspace. Increase the length of INDEX, or decrease fill-in. LINDEX must be set to at least the value of INFO(19) to progress beyond the point of failure. However, note that setting LINDEX to INFO(19) does not guarantee a successful call. Fill-in can usually be decreased by increasing ICNTL(5) and decreasing CNTL(1). Fill-in can sometimes be decreased by factorizing the transpose matrix then calling MA38C/CD with TRANSC=.TRUE. None of these suggestions for reducing fill-in are guaranteed to work, since the pivot search is a heuristic.

–4 Insufficient real workspace. Increase the length of VALUE, or decrease fill-in. LVALUE must be set to at least the value of INFO(21) to progress beyond the point of failure. However, note that setting LVALUE to INFO(21) does not guarantee a successful call. See the comments for error –3.

–5 Both errors –3 and –4 are operative.

+1 Some row and column indices are out-of-range. Such invalid entries are ignored in subsequent calculations. Their number is given by INFO(3). If ICNTL(3) is greater than or equal to 3, then a list of the invalid entries is printed on the diagnostic output.

+2 Duplicates found. Their number is given by INFO(2). If ICNTL(3) is greater than or equal to 3, then a list of the duplicates is printed on the diagnostic output.

+3 Both warnings +1 and +2 are operative.

+4 The matrix is singular (INFO(17) is less than N).

+5 Both warnings +1 and +4 are operative.

+6 Both warnings +2 and +4 are operative.

+7 Warnings +1, +2 and +4 are all operative.

2.3.2 Error diagnostics for MA38B/BD

If MA38B/BD, performs a successful decomposition, INFO(1) will have a non-negative value on exit. A positive value indicates a warning; a negative value indicates an error.

Possible nonzero values for INFO(1) are given below: In each case an explanatory message is output on stream ICNTL(1).

–2 NE has a value less than 1 or greater than KEEP(4)/2.

–3 Insufficient integer workspace. LINDEX must be greater than or equal to INFO(22), as computed by MA38A/AD. LINDEX may need to be higher than INFO(22) if the nonzero pattern is not the same as the matrix factorized by MA38A/AD. Increase the length of INDEX, or factorize the matrix with MA38A/AD while reducing fill-in using the suggestions given for error –3 in Section 2.3.1.

–4 Insufficient real workspace. LVALUE must be greater than or equal to INFO(23), as computed by MA38A/AD. LVALUE may need to be higher than INFO(23) if the nonzero pattern is not the same as the matrix factorized Increase the length of VALUE, or factorize the matrix with MA38A/AD while reducing fill-in using the suggestions given for error –3 in Section 2.3.1.

–5 Both errors –3 and –4 are operative.

–6 A zero pivot has been encountered. The pivot order as computed by MA38A/AD cannot be used for this matrix.

–7 Invalid **LU** factors. Probably caused by not properly passing the factors computed by MA38A/AD to MA38B/BD, or by corrupting the contents of KEEP/VALUE/INDEX between the call to MA38A/AD and the call to MA38B/BD.

+1 Some row and column indices are out-of-range (their number is given by INFO(3)), or not within the nonzero pattern of the **LU** factors of the matrix previously factorized by MA38A/AD or MA38B/BD (their number is given by INFO(4)). Such entries are ignored in subsequent calculations.

+2 Duplicates found. Their number is given by INFO(2). If ICNTL(3) is greater than or equal to 3, then a list of the duplicates is printed on the diagnostic output.

+3 Both warnings +1 and +2 are operative.

+4 The matrix is singular (INFO(17) is less than N).

+5 Both warnings +1 and +4 are operative.

+6 Both warnings +2 and +4 are operative.

+7 Warnings +1, +2 and +4 are all operative.

2.3.3 Error diagnostics for MA38C/CD

If MA38C/CD performs a successful solution, INFO(1) will have a non-negative value on exit. Negative values indicate an error. Possible nonzero values for INFO(1) are now described. In each case an explanatory message is output on stream ICNTL(1).

–7 Invalid **LU** factors. Probably caused by not properly passing the factors computed by MA38A/AD or MA38B/BD to MA38C/CD, or by corrupting the contents of KEEP/VALUE/INDEX between the call to MA38A/AD or MA38B/BD and the call to MA38C/CD.

+8 Iterative refinement option is requested (ICNTL(8)>0) when original matrix not kept (JOB≠1 in call to MA38A/AD or MA38B/BD), or when a triangular system is being solved (JOB≠0 in call to MA38C/CD). The system is solved without iterative refinement.

2.4 Error estimates

We calculate an estimate of the sparse backward error using the theory and measure developed by Arioli, Demmel, and Duff (1989). We use the notation $\bar{\mathbf{x}}$ for the computed solution and a modulus sign on a vector or matrix to indicate the vector or matrix obtained by replacing all entries by their moduli. The scaled residual

$$\frac{|\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{b}|+|\mathbf{A}||\bar{\mathbf{x}}|)_i} \tag{1}$$

is calculated for all equations except those for which the numerator is nonzero and the denominator is small. For the exceptional equations,

$$\frac{|\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{A}||\bar{\mathbf{x}}|)_i+\|\mathbf{A}_i\|_\infty \|\bar{\mathbf{x}}\|_\infty} \tag{2}$$

is used instead, where $\mathbf{A}_i$ is row i of $\mathbf{A}$. The largest scaled residual (1) is returned in RINFO(7) and the largest scaled residual (2) is returned in RINFO(8). If all equations are in category (1), zero is returned in RINFO(8). The computed solution $\bar{\mathbf{x}}$ is the exact solution of the equation

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

where $\delta\mathbf{A}_{ij} \le \max(\texttt{RINFO(7)}, \texttt{RINFO(8)})|\mathbf{A}|_{ij}$ and $\delta\mathbf{b}_i \le \max(\texttt{RINFO(7)}|\mathbf{b}|_i, \texttt{RINFO(8)}\|\mathbf{A}_i\|_\infty \|\bar{\mathbf{x}}\|_\infty)$. Note that $\delta\mathbf{A}$ respects the sparsity in **A**.

Reference

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

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly:

There are 19 routines in MA38 that should not be called directly by the user:
MA38D/DD, MA38E/ED, MA38F/FD, MA38G/GD, MA38H/HD, MA38J/JD, MA38K/KD, MA38L/LD, MA38M/MD, MA38N/ND, MA38O/OD, MA38P/PD, MA38Q/QD, MA38R/RD, MA38S/SD, MA38T/TD, MA38U/UD, MA38Y/YD, MA38Z/ZD.

The following Basic Linear Algebra Subprograms are called:
ISAMAX/IDAMAX, SGEMM/DGEMM, SGEMV/DGEMV, STRSV/DTRSV, and STRSM/DTRSM,

Two HSL routines are called to perform the block upper triangularization: MC13E/ED and MC21B/BD. The FD15A/AD function is called to obtain the machine-dependent epsilon.

Input/output: Errors and warning messages on unit ICNTL(1), diagnostics on ICNTL(2). Output is suppressed if ICNTL(3) $\le$ 0.

Restrictions: N > 0, NE > 0, LVALUE $\ge$ 2*NE, LINDEX $\ge$ 3*NE+2*N+1, NE $\le$ KEEP(4)/2 (MA38B/BD entry).

4 METHOD

4.1 The MA38A/AD routine

Conventional sparse matrix factorization algorithms for general problems rely heavily on indirect addressing. This gives them an irregular memory access pattern that limits their performance on typical parallel-vector supercomputers and on cache-based RISC architectures. In contrast, the *classical* multifrontal method is designed with regular memory access in the innermost loops. This multifrontal method assumes structural symmetry and bases the factorization on an assembly tree generated from the original matrix and an ordering such as minimum degree. The computational kernel, executed at each node of the tree, is one or more steps of **LU** factorization within a square, dense frontal matrix defined by the nonzero pattern of a pivot row and column. These steps of **LU** factorization compute a contribution block (a Schur complement) that is later assembled (added) into the frontal matrix of its parent in the assembly tree.

Although structural asymmetry can be accommodated in the classical multifrontal method by holding the pattern of $\mathbf{A}+\mathbf{A}^T$ and storing explicit zeros, this can have poor performance on matrices whose patterns are very unsymmetric. If we assume from the outset that the matrix may be structurally asymmetric, the situation becomes more complicated. For example, the frontal matrices are rectangular instead of square, and some contribution blocks must be assembled into more than one subsequent frontal matrix. As a consequence, it is no longer possible to represent the factorization by an assembly tree and the more general structure of an assembly directed acyclic graph is required. The MA38 routines are based on this unsymmetric-pattern multifrontal approach. As in the symmetric multifrontal case, advantage is taken of repetitive structure in the matrix by choosing more than one pivot in each frontal matrix. Thus the algorithm can use higher level dense matrix kernels in its innermost loops (Level 3 BLAS). The MA38 algorithm can thus achieve high performance.

MA38 is based on a combined unifrontal/multifrontal algorithm that enables a general fill-in reduction ordering to be applied without some of the data movement of previous multifrontal approaches. For further information, see T. A. Davis and I. S. Duff, *A combined unifrontal/multifrontal method for unsymmetric sparse matrices*, Technical Report TR-95-020, Computer and Information Sciences Department, University of Florida.

It is too expensive to compute the actual degrees of the rows and columns of the active submatrix. To do so would require at least as much work as the numerical factorization itself. This would defeat the performance gained from using the dense matrix kernels. Instead, we compute upper bounds for these degrees at a much lower complexity than the true degrees, since they are obtained from the frontal matrix data structures instead of conventional sparse vectors. We avoid forming the union of sparse rows or columns which would have been needed were we to compute the filled patterns of rows and columns in the active submatrix.

The performance we achieve in the MA38 algorithm thus depends equally on two crucial factors: this approximate degree update algorithm and the numerical factorization within dense, rectangular frontal matrices.

A general sparse code must select pivots based on both numerical and symbolic (fill-reducing) criteria. We therefore combine the analysis phase (pivot selection and symbolic factorization) with the numerical factorization (in MA38A/AD). We construct our rectangular frontal matrices dynamically, since we do not know their structure prior to factorization.

At a particular stage, we first choose a pivot from all the active matrix, using sparsity-preserving and numerical criteria. We call this first pivot the seed pivot. Storage for the frontal matrix is allocated to contain the entries in the pivot row and column plus some room for further expansion determined by the input parameter CNTL(2). We denote the current frontal matrix by **F**, and the submatrix comprising the rows and columns not already pivotal by **C**, calling **C** the contribution block.

Subsequent pivots within this frontal matrix are found within the contribution block, **C**. The frontal matrix grows as more pivots are chosen. We assemble contribution blocks from earlier frontal matrices into this frontal matrix as needed. The selection of pivots within this frontal matrix stops when our next choice for pivot would cause the frontal matrix to become larger than the allocated working array. We then complete the factorization of the frontal matrix using Level 3 BLAS (SGEMM or DGEMM), store the **LU** factors, and place the contribution block, **C**, onto a heap in INDEX and VALUE. The contribution block is deallocated when it is assembled into a subsequent frontal matrix. We then continue the factorization by choosing another seed pivot and generating and factorizing a new frontal matrix.

For more information on the unsymmetric-pattern multifrontal method, see T. A. Davis and I. S. Duff, *An unsymmetric-pattern multifrontal method for sparse LU factorization*, Technical Report RAL 93-036, Computer and Information Sciences Department, Rutherford Appleton Laboratory. To appear in *SIAM J. Matrix Anal. and Applics.*

A symmetric analogue of the approximate degree update algorithm used in MA38A/AD is used by MC47A/AD, a routine that performs an approximate minimum degree ordering. Refer to P. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, Technical Report TR-94-039, Computer and Information Sciences Department, University of Florida. To appear in *SIAM J. Matrix Anal. and Applics.*

4.2 The MA38B/BD routine

The MA38B/BD routine uses the patterns of the **LU** factors and the directed acyclic graph computed by MA38A/AD to factorize a subsequent matrix with the same nonzero pattern as a prior matrix factorized by MA38A/AD. Since the pivot search and symbolic factorization are already performed, MA38B/BD performs the numerical factorization only.

It is in fact possible to relax slightly the requirement that the entries of the new matrix are identical to those of the old. In fact, if $(\mathbf{L}+\mathbf{U})_{ij}$ is nonzero, then the entry $(\mathbf{PAQ})_{ij}$ can be present in INDEX and VALUE (where **P** and **Q** are the permutations determined by MA38A/AD). In addition, if the previous matrix was reduced to block upper triangular form, then the off-diagonal blocks of the subsequent matrix can have an arbitrary sparsity pattern. The argument INFO(4) is set to the number of entries that violate this condition and a warning flag is set.

MA38A/AD can require less floating-point operations than MA38B/BD because MA38A/AD interleaves the numerical factorization of a large frontal matrix with its amalgamation. MA38B/BD can perform extra (wasted) work because it

factorizes frontal matrices after amalgamation. Although very uncommon, cases have been observed where MA38B/BD actually takes more time than MA38A/AD because of this effect.

5 EXAMPLE OF USE

In the example below, we first factorize a small sparse matrix using MA38AD and then solve both $\mathbf{Ax}=\mathbf{b}$ and $\mathbf{A}^T\mathbf{x}=\mathbf{b}$ using MA38CD. We modify the values (but not the sparsity pattern) of the matrix to obtain $\mathbf{A}_2$, refactorize it using MA38BD, and solve $\mathbf{A}_2\mathbf{x}=\mathbf{b}$ twice, using MA38CD both with and without iterative refinement.

```
C MA38 .. test program for specification sheet
C
C Factor and solve a 5-by-5 system, Ax=b, using default parameters,
C where
C     [ 2  3  0  0  0 ]      [  8 ]                  [ 1 ]
C     [ 3  0  4  0  6 ]      [ 45 ]                  [ 2 ]
C A = [ 0 -1 -3  2  0 ], b = [ -3 ]. Solution is x = [ 3 ].
C     [ 0  0  1  0  0 ]      [  3 ]                  [ 4 ]
C     [ 0  4  2  0  1 ]      [ 19 ]                  [ 5 ]
C Solve A'x=b, with solution:
C       x = [  1.8158  1.4561 1.5000 -24.8509 10.2632 ]'
C using the factors of A.   Modify one entry (A(5,2) = 1.0D-14) and
C refactorize.  Solve Ax=b both without and with iterative refinement,
C with true solution (rounded to 4 decimal places)
C       x = [-15.0000 12.6667 3.0000   9.3333 13.0000 ]'
C The backward error is printed and seen to be zero.

C Simple test for MA38 specification sheets
      PROGRAM MA38
C Set array sizes
      INTEGER NMAX, NEMAX, LVALUE, LINDEX
      PARAMETER (NMAX=20, NEMAX=100, LVALUE=300, LINDEX=300)
C Declare MA38 parameters
      INTEGER N, NE, KEEP(20), INDEX(LINDEX), INFO(40), ICNTL(20)
      DOUBLE PRECISION B(NMAX), X(NMAX), W(4*NMAX), VALUE(LVALUE),
     *                 CNTL(10), RINFO(20)
C Local variables
      INTEGER AI(2*NEMAX), I, INFILE, LP
      DOUBLE PRECISION AX(NEMAX)

C Set input data stream and printing stream
        INFILE = 5
        LP     = 6

C Read input matrix and right-hand side.  Keep a copy of the triplet
C form in AI and AX.

        READ(INFILE,*) N, NE
        READ(INFILE,*) (INDEX(I), INDEX(NE+I), I = 1,NE)
        READ(INFILE,*) (VALUE(I), I = 1,NE)
        READ(INFILE,*) (B(I), I = 1,N)
        DO 10 I = 1, NE
           AI(I)    = INDEX(I)
           AI(NE+I) = INDEX(NE+I)
           AX(I)    = VALUE(I)
10      CONTINUE

C Initialize controls.

        CALL MA38ID(KEEP, CNTL, ICNTL)
```

```
C Factorize A. Input matrix is not preserved.

      CALL MA38AD(N, NE, 0, .FALSE., LVALUE, LINDEX, VALUE, INDEX,
     *            KEEP, CNTL, ICNTL, INFO, RINFO)
      IF (INFO(1) .LT. 0) STOP


C Solve Ax = b and print solution.

      CALL MA38CD(N, 0, .FALSE., LVALUE, LINDEX, VALUE, INDEX,
     *            KEEP, B, X, W, CNTL, ICNTL, INFO, RINFO)
      WRITE(LP,'(/A/(F20.4))') 'Solution to Ax=b', (X(I), I=1,N)
      IF (INFO(1) .LT. 0) STOP

C Solve A'x = b and print solution.

      CALL MA38CD(N, 0, .TRUE., LVALUE, LINDEX,  VALUE, INDEX,
     *            KEEP, B, X, W, CNTL, ICNTL, INFO, RINFO)
      WRITE(LP,'(/A/(F20.4))') 'Solution with matrix transposed',
     *                         (X(I), I=1,N)
      IF (INFO (1) .LT. 0) STOP

C Modify one entry of A, and refactorize using MA38BD, keeping ordered
C       copy of original A.

      DO 20 I = 1, NE
         INDEX(I)    = AI(I)
         INDEX(NE+I) = AI(NE+I)
         VALUE(I)    = AX(I)
20    CONTINUE
C       A(5,2) in position 10 of VALUE is (PAQ)_22, the second pivot
C       entry.
      VALUE(10) = 1.0D-14

      CALL MA38BD(N, NE, 1, .FALSE., LVALUE, LINDEX, VALUE, INDEX,
     *            KEEP, CNTL, ICNTL, INFO, RINFO)
      IF (INFO(1) .LT. 0) STOP

C Solve Ax = b without iterative refinement, and print solution.
C This will be inaccurate due to the tiny second pivot entry.

      CALL MA38CD(N, 0, .FALSE., LVALUE, LINDEX,  VALUE, INDEX,
     *            KEEP, B, X, W, CNTL, ICNTL, INFO, RINFO)
      WRITE(LP,'(/A/(F20.4))') 'Solution to system with A changed',
     *                         (X(I), I=1,N)
      IF (INFO(1) .LT. 0) STOP

C Solve Ax = b with iterative refinement, and print solution.
C This is more accurate. We also print the estimate of the backward
C error.

      ICNTL(8) = 10
      CALL MA38CD(N, 0, .FALSE., LVALUE, LINDEX,  VALUE, INDEX,
     *            KEEP, B, X, W, CNTL, ICNTL, INFO, RINFO)
      WRITE(LP,'(/A,A/(F20.4))') 'Solution to system with A changed ',
     *                           'including iterative refinement',
     *                            (X(I), I=1,N)
      WRITE(LP,'(/A/(D20.4))') 'Backward error',RINFO(7),RINFO(8)
      STOP
      END
```

The input is:

```
5 12
1  1    1  2    2  1    2  3    2  5    3  2     3  3
3  4    4  3    5  2    5  3    5  5
2.0     3.0     3.0     4.0     6.0    -1.0     -3.0
2.0     1.0     4.0     2.0     1.0
8.0    45.0    -3.0     3.0    19.0
```

The output is:

```
Solution to Ax=b
              1.0000
              2.0000
              3.0000
              4.0000
              5.0000

Solution with matrix transposed
              1.8158
              1.4561
              1.5000
            -24.8509
             10.2632

Solution to system with A changed
            -15.0000
             12.6121
              3.0000
              9.3061
             13.0000

Solution to system with A changed including iterative refinement
            -15.0000
             12.6667
              3.0000
              9.3333
             13.0000

Backward error
          0.0000D+00
          0.0000D+00
```