1 SUMMARY

To **solve a sparse unsymmetric system of** *m* **linear equations** in *n* unknowns using Gaussian elimination. There are facilities for choosing a good pivot order, factorizing a matrix with a given pivot order, and solving a system of equations using the factorized matrix. Error estimates are available.

Warning: From version 2.0.0, the sizes of the arrays CNTL, ICNTL, and INFO were increased; and RINFO became an array.

ATTRIBUTES — Version: 2.2.0. (26 March 2013) **Types:** Real (single, double). **Remark:** This supersedes MA28. **Calls:** MA50, MC13, MC21, MC29, MC71. **Origin:** I.S. Duff and J.K. Reid, Rutherford Appleton Laboratory. **Original date:** May 1993.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences

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

(a) MA48I/ID can be called to set default values for the control parameters in ICNTL and CNTL. It would normally be called once prior to any calls of MA48A/AD, MA48B/BD, or MA48C/CD.

(b) MA48A/AD prepares data structures for factorization, optionally permuting the matrix **A** to block upper triangular form, and optionally choosing a pivot sequence that leads to the factorization of the permuted matrix **PAQ**, using a pivotal strategy designed to compromise between maintaining sparsity and controlling loss of accuracy through roundoff. There is an option for the user to limit pivoting to the diagonal and an option to input the permutations **P** and **Q**.

(c) MA48B/BD factorizes a matrix **A**, given data provided by MA48A/AD. There are two main options within MA48B/BD, a "normal" call that performs further row permutations to control loss of accuracy by roundoff and a "fast" option that uses information from the normal call including the additional row permutations. An option exists for the case where some columns are unchanged from the previous factorization.

(d) MA48C/CD uses the factors produced by MA48B/BD to solve $\mathbf{Ax}=\mathbf{b}$ or $\mathbf{A}^T\mathbf{x}=\mathbf{b}$. An estimate of the error may be obtained.

Once a single matrix has been factorized by MA48B/BD, further matrices may be factorized more quickly if they have the same structure and the same pivot sequence is acceptable. In many applications it is expected that calls of MA48A/AD, MA48B/BD (normal call), MA48B/BD (fast option), and MA48C/CD will occur successively more often.

2.1.1 To set default values of controlling parameters

MA48I/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 MA48I(CNTL,ICNTL)
```

The double precision version

```
CALL MA48ID(CNTL,ICNTL)
```

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 choose a pivot sequence

The single precision version

```
CALL MA48A(M,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,IW,INFO,RINFO)
```

The double precision version

```
CALL MA48AD(M,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,IW,INFO,RINFO)
```

`M` is an `INTEGER` variable that must be set by the user to the number *m* of rows in the matrix **A**. `M` is not altered by the subroutine. **Restriction:** `M`$\geq$`1`.

`N` is an `INTEGER` variable that must be set by the user to the number *n* of columns in the matrix **A**. `N` is not altered by the subroutine. **Restriction:** `N`$\geq$`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`$\geq$`1`.

`JOB` is an `INTEGER` variable that must be set by the user to indicate whether the pivot sequence is to be chosen automatically (`JOB=1` or 3) or whether it is to be specified by the user (`JOB=2`). If `JOB=3`, the subroutine tries to restrict pivoting to the diagonal. This option can be useful if the matrix is symmetric in structure. `JOB` is not altered by the subroutine. **Restriction:** `1`$\leq$`JOB`$\leq$`3`.

`LA` is an `INTEGER` variable that must be set by the user to the length of arrays `A`, `IRN`, and `JCN`. Usually a sufficient value is `3*NE` although this value is very problem dependent. If it is too small, a suggested value is returned in `INFO(3)` (see Section 2.2). After a successful call, the smallest value that would suffice is returned in `INFO(3)`. `LA` is not altered by the subroutine. **Restriction:** `LA`$\geq$`2*NE`.

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LA`. `A(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(11)` (see Section 2.2). The first `NE` entries of `A` are not altered by the subroutine.

`IRN` is an `INTEGER` array of length `LA`. `IRN(k)` must be set by the user to hold the row index of the entry stored in `A(k)`, `k=1`, ..., `NE`. On return, the leading part of the array holds the row indices of the entries of the matrix when permuted to block upper triangular form, with duplicates accumulated and recommended pivots on the diagonal. The first `NE` entries of `IRN` must be passed unchanged to `MA48B/BD` and `MA48C/CD`.

`JCN` is an `INTEGER` array of length `LA`. `JCN(k)` must be set by the user to hold the column index of the entry stored in `A(k)`, `k=1`, ..., `NE`. On return, `JCN(k)` holds the position in `IRN` of the entry that was input in `A(k)`, `k=1`, ..., `NE`. The first `NE` entries of `JCN` must be passed unchanged to `MA48B/BD`.

`KEEP` is an `INTEGER` array of length `M+5*N+4*N/ICNTL(6)+7`, where `ICNTL(6)` is described in Section 2.2 and has default value 1. `KEEP` need not be set by the user if `JOB=1`. If an input permutation is provided (`JOB=2`), then `KEEP(i)` must be set to the position of row `i` in the permuted matrix, `i=1`, ... `M`, and `KEEP(M+j)` must be set to the index of the column that is in position `j` in the permuted matrix, `j=1`, ..., `N`. For stability reasons, the row permutation may be altered by the subroutine. The whole of `KEEP` must be passed unchanged to subsequent calls to `MA48B/BD`. If `ICNTL(8)` has its default value 0, then the length of `KEEP` can be reduced by `max(N/ICNTL(6), 1)`.

`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 `MA48I/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 may be set by a call to `MA48I/ID`. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

IW is an INTEGER array of length 6*M+3*N that is principally used as workspace. IW(1:N) must be set by the user if JOB≠2 and ICNTL(8) does not have its default value 0 (see Section 2.2); in this case, MA48A/AD will place each column j for which IW(j)=0 at the end of the pivot sequence within its block and any call to MA48B/BD with JOB≠1 will save work by assuming that only these columns are changed since the previous call; the value of IW(1:N) is altered by the subroutine.

INFO is an INTEGER array of length 20 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MA48A/AD, a value for INFO(1) of zero 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, see Section 2.2.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of size 10 that need not be set by the user. On output, RINFO(1) holds the number of floating-point operations needed to factorize the matrix.

2.1.3 To factorize a matrix given a recommended pivotal sequence

The single precision version

```
CALL MA48B(M,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,W,IW,INFO,RINFO)
```

The double precision version

```
CALL MA48BD(M,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,W,IW,INFO,RINFO)
```

M,N,NE are INTEGER variables that must be unchanged since the previous call to MA48A/AD. They are not altered by the subroutine.

JOB is an INTEGER variable that must be set by the user to indicate whether this is a normal call (JOB=1), a subsequent fast call (JOB=2), or a subsequent intermediate call (JOB=3). A JOB=1 call accepts any values for the matrix entries; for the first call, they are usually exactly the same as those passed to MA48A/AD. The JOB=2 call is usually 10-50% faster than the JOB=1 call, but may be numerically unstable if the matrix entries are markedly different from the earlier JOB=1 call. The JOB=3 saves work by recomputing only part of the factorization and is not applicable when ICNTL(8) has its default value of 0 (see Section 2.2). The JOB=2 and JOB=3 calls are not permitted if entries have been dropped (INFO(6) > 0) in the previous call to MA48B/BD with JOB=1 or 3 (see CNTL(3) and CNTL(4) in Section 2.2). JOB is not altered by the subroutine. **Restriction:** $1 \leq$ JOB ≤ 3.

LA is an INTEGER variable that must be set by the user to the length of arrays A and IRN. A suitable value for LA is very problem dependent. We have seen cases where a value of 2*NE will suffice and other cases where over 10*NE is necessary. If it is too small, a suggested value is returned in INFO(4) (see Section 2.2). After a successful call, the smallest value that would suffice is returned in INFO(4). LA is not altered by the subroutine. **Restriction:** LA $\geq$ 2*NE.

A is a REAL (DOUBLE PRECISION in the D version) array of length LA. A(k), k=1, ..., NE must be set by the user to hold the entries of the matrix **A**. The entries must be in exactly the same positions as the corresponding entries were within the corresponding call to MA48A/AD. On return, A holds information on the original matrix and the factorization and must be passed unchanged to MA48C/CD.

IRN is an INTEGER array of length LA. The first NE entries of IRN must be unchanged since the call to MA48A/AD and are not altered by the subroutine. The rest of the array is altered on a JOB=1 or JOB=3 call. The whole array must be passed unchanged to MA48C/CD and to subsequent JOB=2 or JOB=3 calls.

JCN is an INTEGER array of length NE whose entries must be unchanged from the first NE entries of JCN as output from the call to MA48A/AD. JCN is not altered by the subroutine.

KEEP is an INTEGER array of length M+5*N+4*N/ICNTL(6)+7 that must be unchanged since the call to MA48A/AD. KEEP is altered by the subroutine on a JOB=1 call and is not altered on a call with JOB=2. KEEP must be passed unchanged to MA48C/CD and subsequent calls to MA48B/BD. If ICNTL(8) has its default value 0, then the length of KEEP can be reduced by max(N/ICNTL(6), 1).

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 MA48I/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 may be set by a call to MA48I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

W is a REAL (DOUBLE PRECISION in the D version) array of length M that is used for workspace.

IW is an INTEGER array of length 2*M+2*N that is used as workspace.

INFO is an INTEGER array of length 20 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MA48B/BD, a value for INFO(1) of zero 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, see Section 2.2.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of size 10 that need not be set by the user. On output, RINFO(1) holds the number of floating-point operations needed to factorize the matrix.

2.1.4 To solve equations Ax=b or $A^Tx=b$

The single precision version

```
      CALL MA48C(M,N,TRANS,JOB,LA,A,IRN,KEEP,CNTL,ICNTL,
     *           RHS,X,ERROR,W,IW,INFO)
```

The double precision version

```
      CALL MA48CD(M,N,TRANS,JOB,LA,A,IRN,KEEP,CNTL,ICNTL,
     *           RHS,X,ERROR,W,IW,INFO)
```

M,N are INTEGER variables that must be unchanged since the previous call to MA48B/BD. They are not altered by the subroutine.

TRANS is an LOGICAL variable that must be set to .TRUE. if $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is to be solved and to .FALSE. if $\mathbf{Ax} = \mathbf{b}$ is to be solved. TRANS is not altered by the subroutine.

JOB is an INTEGER variable that must be set by the user to control the action of the subroutine. The following choices are available:–

- 1 Calculate the solution without iterative refinement or error estimation.
- 2 Calculate the solution without iterative refinement but with the calculation of the relative backward errors (see Section 2.8).
- 3 Calculate the solution with iterative refinement and calculation of the relative backward errors (see Section 2.8).
- 4 Calculate the solution with iterative refinement, calculation of the relative backward errors and estimation of the ∞–norm of the relative error in the solution (see Section 2.8). If this is requested when $M \neq N$, then action is taken as if JOB had been set to 3.

JOB is not altered by the subroutine. **Restriction:** $1 \leq JOB \leq 4$.

LA is an INTEGER variable that must be unchanged since the previous call to MA48B/BD. It is not altered by the subroutine.

A is a REAL (DOUBLE PRECISION in the D version) array of length LA that must be unchanged since the previous call to MA48B/BD. It is not altered by the subroutine.

IRN is an INTEGER array of length LA that must be unchanged since the previous call to MA48B/BD. It is not altered by the subroutine.

KEEP is an INTEGER array that must be unchanged since the call to MA48B/BD. KEEP 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 MA48I/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 may be set by a call to MA48I/ID. Details of the control parameters are given in Section 2.2. This array is not altered by the subroutine.

RHS is a REAL (DOUBLE PRECISION in the D version) array of length M if $\mathbf{Ax} = \mathbf{b}$ is to be solved and of length N if $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is to be solved. The user must set RHS to contain the vector **b**. It is used as workspace.

X is a REAL (DOUBLE PRECISION in the D version) array of length N if $\mathbf{Ax} = \mathbf{b}$ is to be solved and of length M if $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is to be solved. It need not be set by the user and contains the solution on return.

ERROR is a REAL (DOUBLE PRECISION in the D version) array of length 3 that need not be set by the user. On return from a call with JOB$\geq$2, ERROR(1) and ERROR(2) contain the relative backward errors (see Section 2.8). For square matrices, on return from a call with JOB=4, ERROR(3) contains an estimate of the ∞–norm of the relative error in the solution.

W is a REAL (DOUBLE PRECISION in the D version) workspace array of length 3**LRHS*+*LX*, where *LRHS* is the length of RHS and *LX* the length of X. If JOB=1, only the first *LRHS*+max(M, N) entries are accessed.

IW is an INTEGER array of length M if $\mathbf{Ax} = \mathbf{b}$ is to be solved and of length N if $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is to be solved. It is used as workspace.

INFO is an INTEGER array of length 20 that need not be set by the user. It contains information about the execution of the subroutine. On exit from MA48A/AD, a value for INFO(1) of zero 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, see Section 2.2.

2.2 Arrays for control and information

The elements of the arrays CNTL and ICNTL control the action of MA48A/AD, MA48B/BD, and MA48C/CD. Default values for the elements are set by MA48I/ID. The elements of the array INFO provide information on the action of MA48A/AD, MA48B/BD, and MA48C/CD.

CNTL(1) has default value 0.5. and is used by MA48A/AD to control the switch from sparse to full matrix processing when using MA50 to factorize the diagonal blocks. The switch is made when the ratio of number of entries in the reduced matrix to the number that it would have as a full matrix is greater than CNTL(1). A value greater than 1.0 is treated as 1.0.

CNTL(2) has default value 0.1 and is used for threshold pivoting by MA50 when called from either MA48A/AD or MA48B/BD. Values near zero emphasize sparsity and values near one emphasize stability. If CNTL(2) < 0.0, it is regarded as having the value 0.0; if CNTL(2) > 1.0, it is regarded as having the value 1.0.

CNTL(3) has default value zero. Any entry whose modulus is less than CNTL(3) will be dropped from the factorization by MA50 when called from either MA48A/AD or MA48B/BD. The factorization will then require less storage but will be inaccurate.

CNTL(4) has default value zero. If it is set to a positive value, MA48A/AD or MA48B/BD will treat any pivot whose modulus is less than CNTL(4) as zero. If the matrix is rectangular or rank deficient, it is possible that entries with modulus less than CNTL(4) are dropped from the factorization.

CNTL(5) has default value 0.5. It is used by MA48C/CD to monitor the convergence of the iterative refinement. If successive corrections do not decrease by a factor of at least CNTL(5), convergence is deemed to be too slow and MA48C/CD terminates with INFO(1) set to –8.

The remaining components of CNTL are not used at present.

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

ICNTL(2) has default value 6 and holds the unit number to which diagnostic printing is sent. A non-positive 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 are output.

2 Error and warning messages output.

3 As for 2, plus scalar parameters and a few entries of array parameters on entry and exit from each subroutine.

4 As for 2, plus all parameter values on entry and exit from each subroutine.

ICNTL(4) has default value 3. If ICNTL(4) has a positive value, each pivot search is limited to a maximum of ICNTL(4) columns. If ICNTL(4) is set to the value 0, a special search technique is used to find the best pivot. This is usually only a little slower, but can occasionally be very slow. It may result in reduced fill-in.

ICNTL(5) is used by MA48B/BD and MA48C/CD to control the full-matrix factorization. It has default value 32. Possible values are:

0 Level 1 BLAS used.

1 Level 2 BLAS used.

≥2 Level 3 BLAS used, with block column size ICNTL(5). (Level 2 BLAS used by MA48C/CD.)

ICNTL(6) has default value 1 and is used by MA48A/AD. It defines the minimum size of a block of the block triangular form other than the final block. If block triangularization is not wanted, ICNTL(6) should be set to a value greater than or equal to N. A non-positive value is regarded as the value 1. For further discussion of this variable, see Section 2.6.

ICNTL(7) has default value 1. If ICNTL(7) is set to 0, MA48A/AD will handle structurally rank deficient matrices.

ICNTL(8) has default value 0. If ICNTL(8) is set to another value, the JOB=1 or JOB=3 call to MA48A/AD will place each column j for which IW(j)=0 at the end of the pivot sequence within its block. Any call to MA48B/BD with JOB=2 or JOB=3 will save work by assuming that only these columns are changed since the previous call.

ICNTL(9) has default value 10. It limits the number of refinement iterations performed by MA48C/CD.

ICNTL(10) has default value 0. If set to 1, there is an immediate return from MA48B/BD if LA is too small, without continuing the decomposition to compute the size necessary.

ICNTL(11) has default value 0. If set to 1 on a JOB=2 call to MA48B/BD and the entries in one of the blocks on the diagonal are unsuitable for the pivot sequence chosen on the previous call, the block is refactorized as on a JOB=1 call.

The remaining components of ICNTL are not used at present.

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

INFO(2) is used by MA48A/AD to monitor the adequacy of the allocated space in arrays A, IRN, and JCN, by counting the number of garbage collections performed on these arrays. If INFO(2) is fairly large (say greater than 10), it may be advantageous to increase the size of the arrays. If INFO(2) is small or zero, then one can perhaps save storage by reducing the size of LA on a subsequent run of MA48A/AD with the same data.

INFO(3) indicates, after a successful call to MA48A/AD, the minimum length to which LA could be reduced while still

permitting a successful call for the given matrix. If, however, the user were to decrease the length of the arrays to that size, the number of garbage collections (INFO(2)) may be very high. In the event of failure because LA is too small (INFO(1)=−3), INFO(3) gives a minimum size for LA that permit the code to proceed further.

INFO(4) indicates, after a successful call to MA48A/AD or MA48B/BD, the minimum size of LA required to enable a successful decomposition by MA48B/BD assuming the same pivot sequence and set of dropped entries can be used. In the event of failure because LA is too small (INFO(1)=−3), INFO(4) gives a suggested size for LA, except on a call to MA48B/BD with ICNTL(10)=1.

INFO(5) is used by MA48A/AD to give an estimate of the rank of the matrix. It may be an overestimate since no processing is performed on the full blocks. INFO(5) is used by MA48B/BD to give the rank that it has computed. On an exit from either subroutine with INFO(1) equal to 0, INFO(5) will be equal to min(M,N).

INFO(6) holds, on exit from MA48A/AD or MA48B/BD, the number of entries dropped from the data structure.

INFO(7) holds, on exit from MA48A/AD, the order of the largest non-triangular block on the diagonal of the block triangular form. If matrix is rectangular, INFO(7) will hold the number of rows.

INFO(8) holds, on exit from MA48A/AD, the sum of the orders of all the non-triangular blocks on the diagonal of the block triangular form. If matrix is rectangular, INFO(8) will hold the number of columns.

INFO(9) holds, on exit from MA48A/AD, the total number of entries in all the non-triangular blocks on the diagonal of the block triangular form.

INFO(10) holds, on exit from MA48A/AD with ICNTL(6) ≤ N, the structural rank of the matrix. If ICNTL(6) > N, INFO(10) is set to min(M, N).

INFO(11) holds, on exit from MA48A/AD, the number of multiple entries in the input matrix. Such entries are summed.

INFO(12) holds, on exit from MA48A/AD, the number of entries with out-of-range indices. Each such entry is ignored.

The remaining components of INFO are not used at present.

RINFO(1) holds, on exit from MA50A/AD or MA50B/BD the number of floating-point operations needed to factorize the matrix.

The remaining components of RINFO are not used at present.

2.3 Error diagnostics

2.3.1 Error diagnostics for MA48A/AD

If the subroutine encounters no errors or complications, the value of INFO(1) will be zero on exit from MA48A/AD. The errors and complications that can arise are as follows:

–1 M or N has a value less than 1.

–2 NE has a value less than 1.

–3 LA is too small. INFO(3) gives a minimum value that will permit the code to proceed further. INFO(4) gives a suggested value.

–4 On a call with ICNTL(7) having the value 1, the matrix is structurally rank deficient. The structural rank is given by INFO(10).

–5 Faulty permutation input in KEEP when JOB = 2.

–6 JOB has a value less than 1 or greater than 3.

+1 One or more row or columns indices are out of range or one or more entries are for the same position in the

matrix, or both are true. The first 10 are optionally printed on unit ICNTL(2). (See INFO(11) and INFO(12) in Section 2.2).

+2 The matrix is rank deficient with estimated rank INFO(5).

+3 Combination of warnings +1 and +2.

+4 Not possible to choose all pivots from diagonal (JOB = 3 call).

+5 Combination of warnings +1 and +4.

+6 Combination of warnings +2 and +4.

+7 Combination of warnings +1, +2 and +4.

2.3.2 Error diagnostics for MA48B/BD

If MA48B/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 negative values for INFO(1) are as follows:

–1 M or N has a value less than 1.

–2 NE has a value less than 1.

–3 LA is too small. INFO(4) gives the length necessary for a subsequent successful run unless ICNTL(10)=1.

–6 JOB has a value less than 1 or greater than 3 or JOB equals 2 or 3 after a factorization in which entries were dropped from the factorization (INFO(6) > 0 in Section 2.2).

–7 On a call with JOB = 2, the matrix entries are unsuitable for the pivot sequence chosen on the previous call.

+1 Switched from JOB=2 to JOB=1.

+2 The matrix is rank deficient with estimated rank INFO(5).

+3 Combination of warnings +1 and +2.

2.3.3 Error diagnostics for MA48C/CD

If MA48C/CD performs a successful solution, INFO(1) will have a zero value on exit. Negative values indicate an error. Possible values for INFO(1) are now described.

–1 M or N has a value less than 1.

–6 JOB has a value less than 1 or greater than 4.

–8 Iterative refinement has not converged. This is an indication that the system is very ill-conditioned. The solution may not be accurate although estimates of the error can still be obtained by MA48C/CD.

–9 A problem has occurred in the calculation of matrix norms using MC71A/AD. See the documentation for this routine.

2.4 Rectangular and rank deficient matrices

Rectangular matrices are handled by the code although no attempt is made at prior block triangularization. Rank deficient matrices are also factorized and a warning flag is set (INFO(1) set to +2). If ICNTL(7) is left at its default value of 1, then an error return occurs (INFO(1) = –4) if block triangularization is attempted and the matrix is structurally singular.

2.6 Block upper triangular form

Many large unsymmetric matrices can be permuted to the form

$$\mathbf{PAQ} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & . & . & . & . \\ & \mathbf{A}_{22} & . & . & . & . \\ & & \mathbf{A}_{33} & . & . & . \\ & & & . & . & . \\ & & & & . & . \\ & & & & & \mathbf{A}_{ll} \end{pmatrix}$$

whereupon the system

$$\mathbf{Ax} = \mathbf{b} \qquad (\text{or } \mathbf{A}^T\mathbf{x} = \mathbf{b})$$

can be solved by block back-substitution giving a saving in storage and execution time if the matrices $\mathbf{A}_{ii}$ are much smaller than **A**.

Since it is not very efficient to process a small block (for example a 1×1 block), any block of size less than `ICNTL(6)` other than the final block is merged with its successor.

2.7 Badly-scaled systems

If the user's input matrix has entries differing widely in magnitude, then an inaccurate solution may be obtained. In such cases, the user is advised to first use `MC29A/AD` to obtain scaling factors for the matrix and then explicitly scale it prior to calling `MA48A/AD`. Thereafter, both left and right-hand sides should be scaled as indicated in the code following the example in Section 5.2.

2.8 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 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 **A**. The largest scaled residual (1) is returned in `ERROR(1)` and the largest scaled residual (2) is returned in `ERROR(2)`. If all equations are in category (1), zero is returned in `ERROR(2)`. 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 $\delta\mathbf{A}_{ij} \le \max(\texttt{ERROR(1)}, \texttt{ERROR(2)})|\mathbf{A}|_{ij}$ and $\delta\mathbf{b}_i \le \max(\texttt{ERROR(1)}|\mathbf{b}|_i, \texttt{ERROR(2)}\|\mathbf{A}_i\|_\infty \|\bar{\mathbf{x}}\|_\infty)$. Note that $\delta\mathbf{A}$ respects the sparsity in **A**. For the square case, `ERROR(1)` and `ERROR(2)` can also be used with appropriate condition numbers to obtain an estimate of the relative error in the solution,

$$\frac{\|\mathbf{x}-\bar{\mathbf{x}}\|_\infty}{\|\bar{\mathbf{x}}\|_\infty},$$

which is returned in `ERROR(3)`.

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: MA48D/DD, MC13D, MC21A, MC71A/AD, MA50A/AD, MA50B/BD, and MA50C/CD.

Input/output: Errors and warning messages only. Error messages on unit ICNTL(1), warning messages on unit ICNTL(2). Both have default value 6, and output is suppressed if they are not positive.

Restrictions:

M ≥ 1, N ≥ 1, NE ≥ 1,
1 ≤ JOB ≤ 3 (MA48A/AD and MA48B/BD), 1 ≤ JOB ≤ 4 (MA48C/CD),
LA ≥ 2*NE (MA48A/AD and MA48B/BD).

Major changes: Version 2.0.0: Sizes of CNTL, ICNTL, INFO, and RINFO increased and ICNTL(10) and ICNTL(11) added.

4 METHOD

MA48A/AD reorders the input data to hold the entries by columns and permutes the matrix to block triangular form. It then calls MA50A/AD to factorize each block on the diagonal of the block form. MA50A/AD uses a sparse variant of Gaussian elimination to compute a pivot ordering for the decomposition of **A** into its **LU** factors. It uses pivoting to preserve sparsity in the factors and requires each pivot a_{pj} to satisfy the stability test

$$|a_{pj}| \geq u \max_i |a_{ij}|$$

within the reduced matrix, where u is the threshold held in CNTL(2), with default value 0.1.

MA48B/BD performs the actual factorization by calling MA50B/BD to factorize each diagonal block. The incoming matrix values are rapidly mapped into their correct positions in the permuted form using a mapping array generated in the call to MA48A/AD. The initial call (JOB=1) uses the same stability test as above and generates data structures to allow the possibility of the faster factorization of subsequent matrices. For the full matrix processing, we have included options for the use of level 1, 2 or 3 BLAS. Which is the fastest will depend on the machine and the availability of optimized versions. Where speed is important, each should be tried.

MA48C/CD solves systems by block solution through the block triangular form, using MA50C/CD to effect the solution through forward and back-substitution using entries from the factorization of the diagonal blocks. For the full matrix processing, we have included options for the use of level 1 or 2 BLAS. Where speed is important, each should be tried. Iterative refinement can be performed to improve the accuracy of the solution and to obtain error estimates as discussed in Section 2.8.

A discussion of the design of these subroutines is given by Duff and Reid, *MA48, a Fortran code for direct solution of sparse unsymmetric linear systems of equations*, Report RAL-93-072, Rutherford Appleton Laboratory.

5 EXAMPLE OF USE

5.1 Solving sparse equations with iterative refinement

In the example code shown below, we first decompose a matrix and use information from this decomposition to solve a square set of linear equations. Then we factorize a matrix of a similar sparsity pattern and solve another set of equations with iterative refinement and error estimation in this case.

```
      INTEGER  LA, MAXN, MAXNE
      PARAMETER (LA=200, MAXN=20, MAXNE=100)
      DOUBLE PRECISION A(LA),CNTL(10),RINFO(10),ERROR(3),RHS(MAXN),
     *         W(4*MAXN),X(MAXN)
      INTEGER  I,ICNTL(20),INFO(20),IRN(LA),IW(9*MAXN),JCN(LA),JOB,
     *         KEEP(7*MAXN),N,NE
      LOGICAL TRANS

C     Read in input matrix.
      READ(5,*) N,NE
      IF (N.GT.MAXN .OR. NE.GT.MAXNE) THEN
        WRITE(6,'(A,2I10)') ' Immediate STOP N or NE too large = ',
     *                    N, NE
        STOP
      END IF

      READ (5,FMT=*) (IRN(I),JCN(I),A(I),I=1,NE)

C     Set default controls
      CALL MA48ID(CNTL,ICNTL)

C     Find pivot order, etc.
      JOB = 1
      CALL MA48AD(N,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,IW,INFO,RINFO)
      IF (INFO(1).LT.0) THEN
        WRITE (6,'(A,I3)') 'Error STOP from MA48A/AD with INFO(1) ='
     +    ,INFO(1)
        STOP
      END IF

C     Factorize matrix
      CALL MA48BD(N,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,W,IW,INFO,
     *            RINFO)
      IF (INFO(1).NE.0) THEN
        WRITE (6,FMT='(A,I3/A)') 'STOP from MA48B/BD with INFO(1) =',
     +    INFO(1),'Solution not possible'
        STOP
      END IF

C     Read in right-hand side.
      READ(5,*) (RHS(I),I=1,N)
C     Solve linear system without iterative refinement.
      TRANS = .FALSE.
      CALL MA48CD(N,N,TRANS,JOB,LA,A,IRN,KEEP,CNTL,ICNTL,
     *            RHS,X,ERROR,W,IW,INFO)
C     Print out solution vector.
      WRITE(6,'(/A/(5D13.5))')'The solution vector is',(X(I),I=1,N)

C     Read in and solve system of similar equations.
      READ(5,* ) (A(I),I=1,NE)

C     Use fast factorize option but solve using iterative refinement.
      JOB = 2
```

```
       CALL MA48BD(N,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,W,IW,INFO,
      *            RINFO)

C      Read in right-hand side.
       READ(5,*) (RHS(I),I=1,N)

C Solve using iterative refinement and error estimation.
       JOB = 4
       CALL MA48CD(N,N,TRANS,JOB,LA,A,IRN,KEEP,CNTL,ICNTL,
      *            RHS,X,ERROR,W,IW,INFO)
       WRITE(6,'(/A/(5D13.5))')'The solution vector is',(X(I),I=1,N)
       WRITE(6,'(/A/(3D13.5))')'The ERROR vector is',ERROR
       END
```

Thus if, in this example, we wish to solve:

$$\begin{pmatrix} 3.14 & 7.5 & \\ 4.1 & 3.2 & 0.3 \\ & 1.0 & 4.1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1.0 \\ 2.0 \\ 3.0 \end{pmatrix}$$

followed by the system:

$$\begin{pmatrix} 4.7 & 6.2 & \\ 3.2 & 0.0 & 0.31 \\ & 3.1 & 0.0 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1.1 \\ 2.1 \\ 3.1 \end{pmatrix}$$

we have as input

```
3     7
1     1     3.14
2     3     0.30
3     3     4.1
2     1     4.1
1     2     7.5
3     2     1.0
2     2     3.2
1.0       2.0          3.0
4.7       0.31         0.0          3.2          6.2
3.1       0.0
1.1       2.1          3.1
```

and the output would be

```
 The solution vector is
   0.48858D+00 -0.71219D-01  0.74908D+00

 The solution vector is
  -0.10851D+01  0.10000D+01  0.17975D+02

 The ERROR vector is
   0.79695D-16  0.00000D+00  0.40088D-15
```

5.2 Scaling

In the example code shown below we scale the rectangular matrix and right-hand vector (see section 2.7) prior to decomposition and solution of the consistent set of linear equations.

```
C Specification sheet example.  Illustrating rectangular solution and
C     use of scaling routine MC29.
      INTEGER LA,MAXN,MAXNE
      PARAMETER (LA=200,MAXN=20,MAXNE=100)
```

```
      DOUBLE PRECISION CNTL(10),RINFO(10),ERROR(3),A(LA),W(4*MAXN),
     *                 RHS(MAXN),X(MAXN),R(MAXN),C(MAXN)
      INTEGER JCN(LA),IRN(LA),IW(9*MAXN),KEEP(7*MAXN),ICNTL(20),
     *        INFO(20),JOB,M,N,NE,I,II,J,LP,IFAIL
      LOGICAL TRANS

C     Read in input matrix.
      READ (5,*) M,N,NE
      IF (M.GT.MAXN .OR. N.GT.MAXN .OR. NE.GT.MAXNE) THEN
        WRITE (6,'(A)') ' Error in input data for N and/or NE'
        STOP
      END IF
      READ (5,*) (IRN(I),JCN(I),A(I),I=1,NE)

C     Scale input matrix
      LP = 6
      CALL MC29AD(M,N,NE,A,IRN,JCN,R,C,A(NE+1),LP,IFAIL)
      IF (IFAIL.LT.0) THEN
        WRITE (6,'(A)') ' Error in scaling routine'
        STOP
      END IF
      DO 10 I = 1,M
        R(I) = EXP(R(I))
   10 CONTINUE
      DO 20 I = 1,N
        C(I) = EXP(C(I))
   20 CONTINUE
      DO 30 II = 1,NE
        I = IRN(II)
        J = JCN(II)
        A(II) = A(II)*R(I)*C(J)
   30 CONTINUE

C     Set default controls
      CALL MA48ID(CNTL,ICNTL)

C     Find pivot order, etc.
      JOB = 1
      CALL MA48AD(M,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,IW,INFO,RINFO)
      IF (INFO(1).LT.0) THEN
        WRITE (6,'(A,I3)') 'Error return from MA48AD with INFO(1) =',
     *                     INFO(1)
        STOP
      END IF

C     Factorize matrix
      CALL MA48BD(M,N,NE,JOB,LA,A,IRN,JCN,KEEP,CNTL,ICNTL,W,IW,INFO,
     *            RINFO)
      IF (INFO(1).NE.0) THEN
        WRITE (6,'(A,I3/A)') 'Return from MA48BD with INFO(1) =',
     *                       INFO(1),'Solution not possible'
        STOP
      END IF
C     Read in right hand side.
      READ (5,*) (RHS(I),I=1,M)
C     Scale the right hand side vector by row weights

      DO 40 I = 1,M
        RHS(I) = RHS(I)*R(I)
   40 CONTINUE
```

```
C     Solve linear system.
      TRANS = .FALSE.
      CALL MA48CD(M,N,TRANS,JOB,LA,A,IRN,KEEP,CNTL,ICNTL,RHS,X,ERROR,W,
     *            IW,INFO)

C     Scale the right-hand side vector by column weights

      DO 50 I = 1,N
        X(I) = X(I)*C(I)
   50 CONTINUE

C     Print out solution vector.
      WRITE (6,'(/A/(5D13.5))') ' The solution vector is:',
     *                          (X(I),I=1,N)
      END
```

Thus if, in this example we wish to solve:

$$\begin{pmatrix} 3.14E5 & 7.5E1 & \\ 4.1 & 3.2E{-}3 & 0.3 \\ & 1.0 & 4.1E2 \\ 1.0 & 1.0 & 1.0 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 3.1415E5 \\ 5.0064E0 \\ 1.232E3 \\ 6.0E0 \end{pmatrix}$$

we have as input

```
      4     3       10
      1     1       3.14000D+05
      2     3       0.30
      3     3       4.10000D+02
      2     1       4.1
      1     2       7.50000D+01
      3     2       1.00000D+00
      2     2       3.20000D-03
      4     1       1.00000D+00
      4     2       1.00000D+00
      4     3       1.00000D+00
  0.31415D+06  0.50064D+01  0.12320D+04 0.60000D+01
```

and the output would be

```
 The solution vector is:
  0.10000D+01  0.20000D+01  0.30000D+01
```