Science & Technology Facilities Council

HSL_MA48

PACKAGE SPECIFICATION HSL 2013

1 SUMMARY

To **solve a sparse unsymmetric system of linear equations.** Given a sparse matrix $\mathbf{A} = \{a_{ij}\}_{m\times n}$ and a vector $\mathbf{b}$, this subroutine solves the system $\mathbf{Ax} = \mathbf{b}$ or the system $\mathbf{A}^T\mathbf{x} = \mathbf{b}$. The matrix $\mathbf{A}$ can be rectangular. There is an option for iterative refinement and return of error estimates.

The package HSL_MA48 is an update to the package MA48, and offerse several additional features. For example, there is an option to analyse the matrix and generate the factors with a single call. The storage required for the factorization is chosen automatically and, if there is insufficient space for the factorization, more space is allocated and the factorization is continued. It also returns the number of entries in the factors and has facilities for computing the determinant when the matrix is square and for identifying the rows and columns that are treated specially when the matrix is singular or rectangular. In order to treat matrices with more entries than $2^{31}-1$ (around 2.1×10^9), long integers are used for components in some of the derived data types as indicated in the description of each data type.

ATTRIBUTES — Version: 3.2.0 (21 March 2013). **Interfaces:** C, Fortran, MATLAB. **Types:** Real (single, double). **Remark:** HSL_MA48 offers many additional facilities to the Fortran 77 codes MA48 and MA50. **Calls:** MC71, HSL_ZB01, HSL_ZD11, _AXPY, _DOT, _GEMM, _GEMV, _SWAP, _TRSM, _TRSV, _SCAL, I_AMAX. **Language:** Fortran 2003 subset (F95 + TR15581). **Original date:** November 2001. **Origin:** Version 1 and 2: I.S. Duff and J.K. Reid (Rutherford Appleton Laboratory). Version 3: I.S. Duff (Rutherford Appleton Laboratory).

2 HOW TO USE THE PACKAGE

2.1 Calling sequences

Access to the package requires a USE statement of the form

Single precision version

```
USE HSL_MA48_SINGLE
```

Double precision version

```
USE HSL_MA48_DOUBLE
```

If it is required to use more than one module at the same time, the derived types (Section 2.2) must be renamed in one of the USE statements.

There are five principal subroutines for user calls:

The subroutine MA48_INITIALIZE must be called to initialize the structure for the factors. It may also be called to set default values for the components of the control structure. If non-default values are wanted for any of the control components, the corresponding components should be altered after the call to MA48_INITIALIZE.

MA48_ANALYSE accepts the pattern of $\mathbf{A}$ and chooses pivots for Gaussian elimination using a selection criterion to preserve sparsity. It will optionally find an ordering to block triangular form and exploit that structure. An option exists to restrict pivoting to the diagonal, which might reduce fill-in and operations if the matrix has a symmetric structure. It is possible to perform an analysis without generating the factors, in which case data on the costs of a subsequent factorization are returned to the user. The user can also input a desired pivotal sequence. In this case, the block triangular form will not be computed, the user's column ordering will be respected, but the row ordering might be changed for reasons of stability. It is also possible to request that a set of columns are pivoted on last in which case a subsequent factorization can avoid factorization operations on the earlier columns.

MA48_FACTORIZE factorizes a matrix **A** using the information from a previous call to MA48_ANALYSE. The actual pivot sequence used may differ from that of MA48_ANALYSE. An option exists for a fast factorization where the pivot sequence chosen is identical to the previous factorization and data structures from this earlier factorization are used.

MA48_SOLVE uses the factors generated by MA48_FACTORIZE to solve a system of equations $\mathbf{Ax} = \mathbf{b}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{b}$.

MA48_FINALIZE reallocates the arrays held inside the structure for the factors to have size zero. It should be called when all the systems involving its matrix have been solved unless the structure is about to be used for the factors of another matrix.

There are also two auxiliary subroutines for user calls after a successful factorization:

MA48_DETERMINANT computes the determinant and is for use following a call of MA48_FACTORIZE.

MA48_SPECIAL_ROWS_AND_COLS identifies the rows and columns that are treated specially when the matrix is singular or rectangular. It is for use following a call of MA48_FACTORIZE.

2.2 The derived data types

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

```
use HSL_MA48_double
...
type (MA48_control) :: control
type (MA48_ainfo) :: info_analyse
type (MA48_finfo) :: info_factorize
type (MA48_sinfo) :: info_solve
...
```

The components of MA48_control, MA48_ainfo, MA48_finfo,and MA48_sinfo are explained in Sections 2.2.4, 2.2.5, 2.2.6, and 2.2.7, respectively.

2.2.1 Integer and real kinds

INTEGER denotes default INTEGER and INTEGER(long) denotes INTEGER(kind=selected_int_kind(18)).
REAL denotes default real in the single precision version and double precision real in the double precision version.

2.2.2 32-bit and 64-bit architectures

On a 32-bit architecture, the maximum size of a rank-1 REAL array that can be allocated is huge(0_short)/4 in the single precision version and huge(0_short)/8 in the double precision version, where huge is the Fortran inquiry function. On a 64-bit architecture, it is huge(0_long)/4 and huge(0_long)/8, respectively

2.2.3 Derived data type for the matrix

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

M is an INTEGER scalar which holds the number of rows m of the matrix **A**. **Restriction:** M $\geq$ 1.

N is an INTEGER scalar which holds the number of columns n of the matrix **A**. **Restriction:** N $\geq$ 1.

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

VAL is a REAL allocatable array of length at least NE, the leading part of which holds the values of the entries. Duplicate entries are summed.

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

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

The other components of the type are not used.

2.2.4 Derived data type for control of the subroutines

The module contains a derived type called MA48_CONTROL with the following components

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

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

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

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

LA is an INTEGER(long) scalar. It is no longer used by HSL_MA48 but is kept in for upward compatibility with Version 2.1.1.

MAXLA is an INTEGER(long) scalar. It is no longer used by HSL_MA48 but is kept in for upward compatibility with Version 2.1.1.

MULTIPLIER is a REAL scalar used by MA48_FACTORIZE when a real or integer array that holds data for the factors is too small. The array is reallocated with its size changed by the factor MULTIPLIER. The default value is 2.0. The value actually used in the code is the minimum of MULTIPLIER and 1.2.

REDUCE is a REAL scalar. It is no longer used by HSL_MA48 but is kept in for upward compatibility with Version 2.1.1.

SWITCH is an REAL scalar used by MA48_ANALYSE to control the switch from sparse to full matrix processing when factorizing 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 SWITCH. A value greater than 1.0 is treated as 1.0. The default value is 0.5.

U is a REAL scalar that is used by MA48_ANALYSE and MA48_FACTORIZE. It holds the threshold parameter for the pivot control. The default value is 0.01. For problems requiring greater than average numerical care a higher value than the default would be advisable. Values greater than 1.0 are treated as 1.0 and less than 0.0 as 0.0.

DROP is a REAL scalar that is used by MA48_ANALYSE and MA48_FACTORIZE. Any entry whose modulus is less than DROP will be dropped from the factorization. The factorization will then require less storage but will be inaccurate. The default value is 0.0.

TOLERANCE is a REAL scalar that is used by MA48_ANALYSE and MA48_FACTORIZE. If it is set to a positive value, any pivot whose modulus is less than TOLERANCE will be treated as zero. The default value is 0.0.

CGCE is a REAL scalar that is used by MA48_SOLVE. It is used to monitor the convergence of the iterative refinement. If successive corrections do not decrease by a factor of at least CGCE, convergence is deemed to be too slow and MA48_SOLVE terminates with SINFO%FLAG set to -8. The default value is 0.5.

PIVOTING is a INTEGER scalar that is used to control numerical pivoting by MA48_ANALYSE. If PIVOTING has a positive value, each pivot search is limited to a maximum of PIVOTING columns. If PIVOTING is set to the value 0, a full Markowitz 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. The default value is 3.

DIAGONAL_PIVOTING is an LOGICAL scalar used by MA48_ANALYSE to limit pivoting to the diagonal. It will do so if DIAGONAL_PIVOTING is set to .TRUE.. Its default value is .FALSE..

FILL_IN is an INTEGER scalar used by MA48_ANALYSE to determine the initial storage allocation for the matrix factors. It will be set to FILL_IN times the value of MATRIX%NE. The default value is 3.

BTF is an INTEGER scalar used by MA48_ANALYSE to define the minimum size of a block of the block triangular form other than the final block. If block triangularization is not wanted, BTF 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 2.5. The default value is 1.

STRUCT is an LOGICAL scalar used by MA48_ANALYSE. If STRUCT is set to .TRUE., the subroutine will exit immediately structural singularity is detected. The default value is .FALSE..

FACTOR_BLOCKING is an INTEGER scalar used by MA48_FACTORIZE to determine the block size used for the Level 3 BLAS within the full factorization. If it is set to 1, Level 1 BLAS is used, if to 2, Level 2 BLAS is used. The default value is 32.

SOLVE_BLAS is an INTEGER scalar used by MA48_SOLVE to determine whether Level 2 BLAS is used (SOLVE_BLAS $>$ 1) or not (SOLVE_BLAS $\leq$ 1). The default value is 2.

MAXIT is an INTEGER scalar used by MA48_SOLVE to limit the number of refinement iterations. If MAXIT is set to zero then MA48_SOLVE will not perform any error analysis or iterative refinement. The default value is 10.

SWITCH_MODE is an LOGICAL scalar used by MA48_FACTORIZE. If it has the value .TRUE., a switch to slow mode is made when the fast mode is given an unsuitable pivot sequence. The default value is .FALSE..

2.2.5 Derived data type for information from MA48_ANALYSE

The module contains a derived type called MA48_AINFO with the following components

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

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

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

DUP is an INTEGER(long) scalar which is set to the number of duplicate entries.

DROP is an INTEGER(long) scalar which is set to the number of entries dropped from the data structure.

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

OPS is a REAL scalar which is set to the number of floating-point operations required by the factorization.

RANK is an INTEGER scalar that gives an estimate of the rank of the matrix.

STRUC_RANK is an INTEGER scalar that, if BTF is less than or equal to N, holds the structural rank of the matrix. If BTF is greater than N, STRUC_RANK is set to min(M, N).

LENA_ANALYSE is an INTEGER(long) scalar that gives the number of REAL and INTEGER words required for the analysis.

LENJ_ANALYSE is an INTEGER(long) scalar that gives the number of INTEGER words required for an auxiliary array for the analysis.

LEN_ANALYSE is an INTEGER(long) scalar. It has been kept for upward compatibility with Version 2.1.1. It is now redundant but is set equal to the maximum of LENA_ANALYSE and LENJ_ANALYSE.

LENA_FACTORIZE is an INTEGER(long) scalar that gives the number of REAL words required for successful subsequent factorization assuming the same pivot sequence and set of dropped entries can be used.

LENI_FACTORIZE is an INTEGER(long) scalar that gives the number of INTEGER words required for successful subsequent factorization assuming the same pivot sequence and set of dropped entries can be used. In the present version (3.0.0) of the code, LENI_FACTORIZE is equal to LENA_FACTORIZE.

LEN_FACTORIZE is an INTEGER(long) scalar. It has been kept for upward compatibility with Version 2.1.1. It is now redundant but is set equal to the maximum of LENA_FACTORIZE and LENI_FACTORIZE.

NCMPA is an INTEGER scalar that holds the number of compresses of the internal data structure performed by MA48_ANALYSE. If NCMPA is fairly large (say greater than 10), performance may be very poor.

LBLOCK is an INTEGER scalar that holds the order of the largest non-triangular block on the diagonal of the block triangular form. If the matrix is rectangular, LBLOCK will hold the number of rows.

SBLOCK is an INTEGER scalar that holds the sum of the orders of all the non-triangular blocks on the diagonal of the block triangular form. If the matrix is rectangular, SBLOCK will hold the number of columns.

TBLOCK is an INTEGER(long) scalar that holds the total number of entries in all the non-triangular blocks on the diagonal of the block triangular form.

2.2.6 Derived data type for information from MA48_FACTORIZE and, optionally, from MA48_ANALYSE

The module contains a derived type called MA48_FINFO with the following components

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

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

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

OPS is a REAL scalar which is set to the number of floating-point operations required by the factorization.

DROP is an INTEGER(long) scalar which is set to the number of entries dropped from the data structure.

LENA_FACTORIZE is an INTEGER(long) scalar that gives the number of REAL words required for successful subsequent factorization assuming the same pivot sequence and set of dropped entries can be used.

LENI_FACTORIZE is an INTEGER(long) scalar that gives the number of INTEGER words required for successful subsequent factorization assuming the same pivot sequence and set of dropped entries can be used. In the present version (3.0.0) of the code LENI_FACTORIZE is equal to LENA_FACTORIZE.

LEN_FACTORIZE is an INTEGER(long) scalar. It has been kept for upward compatibility with Version 2.1.1. It is now redundant but is set equal to the maximum of LENA_FACTORIZE and LENI_FACTORIZE.

SIZE_FACTOR is an INTEGER(long) scalar that gives the number of entries in the matrix factors.

RANK is an INTEGER scalar that gives an estimate of the rank of the matrix.

2.2.7 Derived data type for information from MA48_SOLVE

The module contains a derived type called MA48_SINFO with the following components

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

MORE is an INTEGER scalar that provides further information in the case of an error, see Section 2.4.3.

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

2.2.8 Derived data type for the factors of a matrix

The module contains a derived type called MA48_FACTORS. The components of MA48_FACTORS are used to pass data between the subroutines of the package and must not be altered by the user. Some of the components of MA48_FACTORS are long integers.

2.3 Argument lists

We use square brackets [] to indicate optional arguments.

2.3.1 The initialization subroutine

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

```
CALL MA48_INITIALIZE([FACTORS,CONTROL])
```

FACTORS is optional, scalar, of INTENT(OUT) and of type MA48_FACTORS. On exit, some components of FACTORS will be set.

CONTROL is optional, scalar, of INTENT(OUT) and of type MA48_CONTROL. On exit, its components will have been given the default values specified in Section 2.2.4.

2.3.2 To analyse the sparsity pattern

```
CALL MA48_ANALYSE(MATRIX,FACTORS,CONTROL,AINFO[,FINFO,PERM,ENDCOL])
```

MATRIX is scalar, of INTENT(IN) and of type ZD11_TYPE. The user must set the components M, N, NE, ROW, COL, and VAL. **Restrictions:** MATRIX%M $\geq$ 1, MATRIX%N $\geq$ 1, and MATRIX%NE $\geq$ 0.

FACTORS is scalar, of INTENT(INOUT) and of type MA48_FACTORS. It must have been initialized by a call to MA48_INITIALIZE or have been used for a previous calculation. In the latter case, the previous data will be lost but the allocatable arrays will not be reallocated unless they are found to be too small.

CONTROL is scalar, of INTENT(IN) and of type MA48_CONTROL. Its components control the action, as explained in Section 2.2.4.

AINFO is scalar, of INTENT(OUT) and of type MA48_AINFO. Its components provide information about the execution, as explained in Section 2.2.5.

FINFO is scalar, optional, of INTENT(OUT) and of type MA48_FINFO. If present, the call to MA48_ANALYSE will compute and store the factorization of the matrix. Its components provide information about the execution of the factorization, as explained in Section 2.2.6.

PERM is an array of shape $(m+n)$, optional, of INTENT(IN) and of type INTEGER. If present, PERM(i), i = 1, 2, ..., m, should be set to the position of row i in the permuted matrix, and PERM($m+j$), j = 1, 2, ..., n, should be set to the index of the column that is in position j in the permuted matrix. In this case, the block triangular form will not be computed. The routine will try to use this input pivotal sequence but may change the row ordering if necessary for stability reasons.

ENDCOL is an array of shape (n), optional, of INTENT(IN) and of type INTEGER. MA48_ANALYSE will place each column j for which ENDCOL(j)=0 at the end of the pivot sequence within its block. A subsequent call to MA48_FACTORIZE can save work by assuming that only these columns are changed since the previous call.

2.3.3 To perform a factorization

```
CALL MA48_FACTORIZE(MATRIX,FACTORS,CONTROL,FINFO[,FAST,PARTIAL])
```

MATRIX is scalar, of INTENT(IN) and of type ZD11_TYPE. The components M, N and NE must be unaltered since the call to MA48_ANALYSE. The user must set the component VAL to hold the real values of the entries.

FACTORS is scalar, of INTENT(INOUT) and of type MA48_FACTORS. It must be unaltered since the call to MA48_ANALYSE or a subsequent call to MA48_FACTORIZE.

CONTROL is scalar, of INTENT(IN) and of type MA48_CONTROL. Its components control the action, as explained in Section 2.2.4.

FINFO is scalar, of INTENT(OUT) and of type MA48_FINFO. Its components provide information about the execution, as explained in Section 2.2.6.

FAST is scalar, optional, of INTENT(IN) and of type INTEGER. This option is only available after there has been at least one successful factorization on a previous matrix. If present, the factorization will use the same pivot sequence as the previous factorization. It will also utilize data structures from the earlier factorization to effect a more rapid factorization. If, however, entries were dropped from the previous analysis or factorization, this option will be inoperative and the matrix will be factorized using the same pivot sequence but will regenerate the data structures during factorization.

PARTIAL is scalar, optional, of INTENT(IN) and of type INTEGER. If present, the factorization will be performed on only the last columns of the matrix that were flagged by the parameter ENDCOL during the call to MA48_ANALYSE. If, however, entries were dropped from the previous analysis or factorization, this option will be inoperative and the matrix will be factorized using the same pivot sequence but will regenerate the data structures during factorization.

2.3.4 To solve a set of equations

```
CALL MA48_SOLVE(MATRIX,FACTORS,RHS,X,CONTROL,SINFO[,TRANS,RESID,ERROR])
```

MATRIX is scalar, of INTENT(IN) and of type ZD11_TYPE. It must be unaltered since the call to MA48_FACTORIZE and is not altered by the subroutine.

FACTORS is scalar, of INTENT(IN) and of type MA48_FACTORS. It must be unaltered since the call to MA48_FACTORIZE and is not altered by the subroutine.

RHS is an array of shape (n) of INTENT(IN) and of type REAL. It must be set by the user to the vector $\mathbf{b}$.

X is an array of shape (n) of INTENT(OUT) and of type REAL. On return it holds the solution $\mathbf{x}$.

CONTROL is scalar, of INTENT(IN) and of type MA48_CONTROL. Its components control the action, as explained in Section 2.2.4.

SINFO is scalar, of INTENT(OUT) and of type MA48_SINFO. Its components provide information about the execution, as explained in Section 2.2.7.

TRANS is scalar, optional, of INTENT(IN) and of type INTEGER. If present $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is solved, otherwise the solution is obtained for $\mathbf{Ax} = \mathbf{b}$.

RESID is an array of shape 2, optional, of INTENT(OUT) and of type REAL. If present and CONTROL%MAXIT $\geq$ 1, the scaled residual for the two categories of equations (see Section 2.8) will be held in RESID(1) and RESID(2), respectively.

ERROR is an scalar, optional, of INTENT(OUT) and of type REAL. If present and CONTROL%MAXIT $\geq$ 1, an estimate of the error in solving the equations (see Section 2.8) will be held in ERROR.

2.3.5 The finalization subroutine

```
CALL MA48_FINALIZE(FACTORS,CONTROL,INFO)
```

FACTORS is scalar, of INTENT(INOUT) and of type MA48_FACTORS. On exit, its allocatable array components will have been deallocated. Without such finalization, the storage occupied is unavailable for other purposes.

CONTROL is scalar, of INTENT(IN) and of type MA48_CONTROL. Its components control the action, as explained in Section 2.2.4.

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

2.3.6 To compute the determinant following a successful factorization

```
CALL MA48_DETERMINANT(FACTORS,SGNDET,LOGDET,INFO)
```

FACTORS is scalar, of INTENT(IN) and of type MA48_FACTORS. It must be unaltered since the call to MA48_FACTORIZE and is not altered by the subroutine.

SGNDET is an INTEGER variable, of INTENT(OUT) that need not be set by the user. On return, it has the value 1 if the determinant is positive, -1 if the determinant is negative, or 0 if the determinant is zero or the matrix is not square.

LOGDET is a REAL (DOUBLE PRECISION in the D version) variable, of INTENT(OUT) that need not be set by the user. On return, it holds the logarithm of the absolute value of the determinant, or zero if the determinant is zero or the matrix is not square.

INFO is an INTEGER variable, of INTENT(OUT) that need not be set by the user. On return, its value is 0 if the call was successful and is -1 if the allocation of a temporary array failed.

2.3.7 To identify the rows and columns that are treated specially following a successful factorization

```
CALL MA48_SPECIAL_ROWS_AND_COLS(FACTORS,RANK,ROWS,COLS,INFO)
```

FACTORS is scalar, of intent(in) and of type MA48_FACTORS. It must be unaltered since the call to MA48_FACTORIZE and is not altered by the subroutine.

RANK is an INTEGER variable, of INTENT(OUT) that need not be set by the user. On return, it holds the calculated rank of the matrix (it is the rank of the matrix actually factorized).

ROWS is an INTEGER array of length M, of INTENT(OUT) that need not be set by the user. On return, it holds a permutation. The indices of the rows that are taken into account when solving $\mathbf{Ax} = \mathbf{b}$ are ROWS(i), $i \leq$ RANK.

COLS is an INTEGER array of length N, of INTENT(OUT) that need not be set by the user. On return, it holds a permutation. The indices of the columns that are taken into account when solving $\mathbf{Ax} = \mathbf{b}$ are COLS(j), $j \leq$ RANK.

INFO is an INTEGER variable, of INTENT(OUT) that need not be set by the user. On return, its value is 0 if the call was successful and is -1 if the allocation of a temporary array failed.

2.4 Error diagnostics

2.4.1 When performing the analysis.

A successful return from MA48_ANALYSE is indicated by AINFO%FLAG having the value zero. A negative value is associated with an error message that will be output on unit CONTROL%LP. Possible negative values are:

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

-2 Value of MATRIX%N out of range. MATRIX%N $<$ 1. AINFO%MORE is set to the value of MATRIX%N.

-3 Value of MATRIX%NE out of range. MATRIX%NE $<$ 0. AINFO%MORE is set to the value of MATRIX%NE.

-4 Failure of an allocate or deallocate statement. AINFO%STAT is set to the STAT value.

-5 On a call with STRUCT having the value .TRUE., the matrix is structurally rank deficient. The structural rank is given by STRUC_RANK.

-6 The array PERM does not hold valid permutations. AINFO%MORE holds the first component at which an error was detected.

-7 An error occurred in a call to MA48_FACTORIZE (when the optional parameter finfo was in the call). The only reason why this can happen is because of an allocation error in MA48_FACTORIZE.

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

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

+2 Duplicate indices. Action taken by subroutine is to sum corresponding reals. AINFO%DUP is set to the number of duplicate entries. Details of the first ten are optionally printed on unit CONTROL%MP.

+3 Combination of a +1 and a +2 warning.

+4 The matrix is rank deficient with estimated rank AINFO%RANK.

+5 Combination of a +1 and a +4 warning.

+6 Combination of a +2 and a +4 warning.

+7 Combination of a +1, a +2, and a +4 warning.

+8 Not possible to choose all pivots from diagonal (call with CONTROL%DIAGONAL_PIVOTING equal to .TRUE.).

+9 to +15 Combination of warnings that sum to this total.

2.4.2 When factorizing the matrix

A successful return from MA48_FACTORIZE is indicated by FINFO%FLAG having the value zero. A negative value is associated with an error message that will be output on unit CONTROL%LP. In this case, no solution will have been calculated. Possible negative values are:

-1 Value of MATRIX%M differs from the MA48_ANALYSE value. FINFO%MORE holds value of MATRIX%M.

-2 Value of MATRIX%N differs from the MA48_ANALYSE value. FINFO%MORE holds value of MATRIX%N.

-3 Value of MATRIX%NE out of range. MATRIX%NE $<$ 0. FINFO%MORE holds value of MATRIX%NE.

-4 Failure of an allocate statement. FINFO%STAT is set to the STAT value.

-10 MA48_FACTORIZE has been called without a prior call to MA48_ANALYSE.

-11 MA48_FACTORIZE has been called with FAST present, but the matrix entries are unsuitable for this.

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

+4 Matrix is rank deficient. In this case, FINFO%RANK will be set to the rank of the factorization. In the subsequent solution, all columns in the singular block will have the corresponding component in the solution vector set to zero.

2.4.3 When using factors to solve equations

A successful return from MA48_SOLVE is indicated by SINFO%FLAG having the value zero. A negative value is associated with an error message that will be output on unit CONTROL%LP. In this case, the solution will not have been completed. Possible negative values are:

-1 Value of MATRIX%M differs from the MA48_ANALYSE value. SINFO%MORE holds the value of MATRIX%M.

-2 Value of MATRIX%N differs from the MA48_ANALYSE value. SINFO%MORE holds the value of MATRIX%N.

-3 Value of MATRIX%NE out of range. MATRIX%NE $<$ 0. SINFO%MORE holds the value of MATRIX%NE.

-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 MA48_SOLVE.

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

-10 MA48_SOLVE has been called without a prior call to MA48_FACTORIZE.

2.5 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 (AINFO%FLAG or FINFO%FLAG set to +4). If CONTROL%STRUCT is set to .TRUE, then an error return occurs (AINFO%FLAG = -5) if block triangularization is attempted and the matrix is structurally singular.

The package identifies a square submatrix of $\mathbf{A}$ that it considers to be nonsingular. When solving $\mathbf{Ax} = \mathbf{b}$, equations outside this submatrix are ignored and solution components that correspond to columns outside the submatrix are set to zero. MA48_SPECIAL_ROWS_AMD_COLS identifies the rows and columns of this submatrix from stored integer data.

It should be emphasized that the primary purpose of the package is to solve square nonsingular sets of equations. The rank is determined from the number of pivots that are not small or zero. There are more reliable (but much more expensive) ways of determining numerical rank.

2.6 Block upper triangular form

Many large unsymmetric square matrices can be permuted to the form

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

whereupon the system

$\mathbf{Ax} = \mathbf{b}$ (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 $\mathbf{A}$.

Since it is not very efficient to process a small block (for example a 1×1 block), any block of size less than CONTROL%BTF 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 a scaling routine (for example, MC29A/AD, MC64A/AD, or MC77A/AD) to obtain scaling factors for the matrix and then explicitly scale it prior to calling this package.

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

$$\frac{|\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{b}|+|\mathbf{A}||\bar{\mathbf{x}}|)_i} \tag{2.1}$$

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

$$\frac{|\mathbf{b}-\mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{A}||\bar{\mathbf{x}}|)_i+||\mathbf{A}_i||_\infty||\bar{\mathbf{x}}||_\infty} \tag{2.2}$$

is used instead, where $\mathbf{A}_i$ is row i of $\mathbf{A}$. The largest scaled residual (2.1) is returned in RESID(1) and the largest scaled residual (2.2) is returned in RESID(2). If all equations are in category (2.1), zero is returned in RESID(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 $\delta\mathbf{A}_{ij} \leq \max(\texttt{RESID(1)},\texttt{RESID(2)})|\mathbf{A}|_{ij}$ and $\delta\mathbf{b}_i \leq \max(\texttt{RESID(1)}|\mathbf{b}|_i,\texttt{RESID(2)}||\mathbf{A}_i||_\infty||\bar{\mathbf{x}}||_\infty)$. Note that $\delta\mathbf{A}$ respects the sparsity in $\mathbf{A}$. For the square case, RESID(1) and RESID(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.

Reference [1] 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.

2.9 Determinant

For the determinant, the package computes the parity of the permutations that have been applied and the product of the diagonal entries of the triangular factors.

3 GENERAL INFORMATION

Other routines called directly: The auxiliary routines that are revised Fortran 95 versions of routines from the packages MA48, MA50, MA51, MC21, and MC13 are internal to the package. Packages HSL_ZD11 and HSL_ZB01 are also used.

Input/output: Error, warning and diagnostic messages only. Error messages on unit CONTROL%LP and warning and diagnostic messages on unit CONTROL%WP and CONTROL%MP, respectively. These have default value 6, and printing of these messages is suppressed if the relevant unit number is set negative. These messages are also suppressed if MA48_CONTROL%LDIAG is less than 1.

Changes in Version 2.0.0 HSL_ZD11 used in place of HSL_ZD01 so that allocatable arrays are used instead of pointer arrays for dynamic memory.

Changes in Version 3.0.0 The most significant changes involve the use of long integers, the ability to reallocate arrays and continue the factorization if the arrays were too small, and some changes to decouple the real and integer array sizes. Extra parameters have been added to MA48_CONTROL, MA48_AINFO, and MA48_FINFO. HSL_ZB01 is now called to increase array sizes if necessary.

Restrictions: MATRIX%M ≥ 1, MATRIX%N ≥ 1, MATRIX%NE ≥ 0.

4 METHOD

A version of sparse Gaussian elimination is used.

The MA48_ANALYSE entry 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 CONTROL%U, with default value 0.01. It then optionally computes the numerical factors.

The MA48_FACTORIZE entry factorizes the matrix by using information generated by MA48_ANALYSE. The initial call uses the same stability test as above and generates data structures to allow for the possibility of the faster factorization of subsequent matrices, using the optional parameter, FAST.

The MA48_SOLVE entry uses the factors from MA48_FACTORIZE to solve systems of equations. 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 the MA48 routines called by this package 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 (obtainable from the Web page http://www.numerical.rl.ac.uk/reports/reports.html) and in ACM Trans Math Softw **22**, 1996, 187-226.

5 EXAMPLE OF USE

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.

Program

```
! Simple example of use of HSL_MA48
PROGRAM MAIN
   USE HSL_MA48_DOUBLE
   IMPLICIT NONE
   TYPE(ZD11_TYPE) MATRIX
   TYPE(MA48_CONTROL) CONTROL
   TYPE(MA48_AINFO) AINFO
   TYPE(MA48_FINFO) FINFO
   TYPE(MA48_SINFO) SINFO
   TYPE(MA48_FACTORS) FACTORS

   DOUBLE PRECISION, ALLOCATABLE :: B(:),X(:)
```

```
   DOUBLE PRECISION RES(2),ERR
   INTEGER I,INFO,FAST,M,N,NE

! Read matrix order and number of entries

      READ (5,*) M,N,NE
      MATRIX%M = M
      MATRIX%N = N
      MATRIX%NE = NE

! Allocate arrays of appropriate sizes
      ALLOCATE(MATRIX%VAL(NE), MATRIX%ROW(NE), MATRIX%COL(NE))
      ALLOCATE(B(N),X(N))

! Read matrix and right-hand side
      READ (5,*) (MATRIX%ROW(I),MATRIX%COL(I),MATRIX%VAL(I),I=1,NE)
      READ (5,*) B

! Initialize the structures
      CALL MA48_INITIALIZE(FACTORS,CONTROL)

! Analyse and factorize

      CALL MA48_ANALYSE(MATRIX,FACTORS,CONTROL,AINFO,FINFO)
      IF(AINFO%FLAG .ne. 0) THEN
         WRITE(6,'(A,I2)') &
            ' Failure of MA48_ANALYSE with AINFO%FLAG=', AINFO%FLAG
         STOP
      END IF

! Solve without iterative refinement
      CALL MA48_SOLVE(MATRIX,FACTORS,B,X,CONTROL,SINFO)
      IF(SINFO%FLAG == 0) WRITE(6,'(A,/,(6ES11.3))')  &
         'Solution of first set of equations without refinement is',X

! Read new matrix and right-hand side
      READ (5,*) (MATRIX%VAL(I),I=1,NE)
      READ (5,*) B

! Fast factorize
      CALL MA48_FACTORIZE(MATRIX,FACTORS,CONTROL,FINFO,FAST)
      IF(FINFO%FLAG .ne. 0) THEN
         WRITE(6,'(A,I2)') &
            ' Failure of MA48_FACTORIZE with FINFO%FLAG=', FINFO%FLAG
         STOP
      END IF

! Solve with iterative refinement
      CALL MA48_SOLVE(MATRIX,FACTORS,B,X,CONTROL,SINFO,RESID=RES,ERROR=ERR)
```

```
      IF(SINFO%FLAG == 0)THEN
         WRITE(6,'(//A,/,(6ES11.3))')       &
              'Solution of second system with refinement is',X
         WRITE(6,'(A/(6ES11.3))') 'Scaled residual is',RES
         WRITE(6,'(A/(6ES11.3))') 'Estimated error is',ERR
      ENDIF

! Clean up
      DEALLOCATE(MATRIX%VAL, MATRIX%ROW, MATRIX%COL)
      CALL MA48_FINALIZE(FACTORS,CONTROL,INFO)

END PROGRAM MAIN
```

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     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

```
Solution of first set of equations without refinement is
  4.886E-01 -7.122E-02  7.491E-01


Solution of second system with refinement is
 -1.085E+00  1.000E+00  1.798E+01
Scaled residual is
  7.163E-17  0.000E+00
Estimated error is
  3.603E-16
```