1 SUMMARY

To **solve a sparse unsymmetric system of linear equations.** Given a square unsymmetric sparse matrix $\mathbf{A}$ of order n and an n-vector $\mathbf{b}$, this subroutine solves the system $\mathbf{Ax=b}$ or $\mathbf{A}^T\mathbf{x=b}$.

The method used is a direct method based on a sparse multifrontal variant of Gaussian elimination. An initial ordering for the pivotal sequence is chosen using the pattern of the matrix $\mathbf{A} + \mathbf{A}^T$ and is later modified for reasons of numerical stability. Thus this code performs best on matrices whose pattern is symmetric, or nearly so. Other HSL codes might be more appropriate for symmetric sparse matrices (for example, MA57), or for very unsymmetric and very sparse matrices (for example, MA38 or MA48).

ATTRIBUTES — **Version:** 1.2.0. **Types:** Real (single, double). **Calls:** The HSL routines: MC21, MC29, and MC47. The BLAS routines: I_AMAX, _GEMV, _TRSV. **Original date:** May 1995, revised April 2008. **Language:** Fortran 77. **Origin:** P. R. Amestoy (ENSEEIHT-IRIT, Toulouse, France) and I. S. Duff (Rutherford Appleton Laboratory).

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences

There are only two entries:

(a) MA41I/ID sets default values for the components of the arrays that hold control parameters. Normally the user will call MA41I/ID prior to any call to MA41A/AD.. If non-default values for any of the control parameters are required, they should be set immediately after the call to MA41I/ID.

(b) A Fortran driver subroutine MA41A/AD has been designed as a user friendly interface to the multifrontal code. This driver, in addition to providing the normal functionality of a sparse solver, incorporates some pre- and post-processing. This driver enables the user to preprocess the matrix to obtain a maximum transversal so that the permuted matrix has a zero-free diagonal, to perform prescaling of the original matrix (a choice of scaling strategies is provided), to use iterative refinement to improve the solution, and finally to perform error analysis.

The driver routine MA41A/AD offers similar functionalities to other sparse direct solvers, depending on the value of one of its parameters (JOB). These are:

(i) JOB = 1 accepts the pattern of $\mathbf{A}$ and chooses pivots from the diagonal using a selection criterion to preserve sparsity. It uses the pattern of $\mathbf{A}+\mathbf{A}^T$ but ignores numerical values. It subsequently constructs subsidiary information for the actual factorization (a JOB=2 call). An option exists for the user to input the pivot sequence, in which case only the necessary information for a JOB = 2 entry will be generated. We call the JOB=1 entry, the "analysis phase".

(ii) JOB = 2 factorizes a matrix $\mathbf{A}$ using the information from a previous call with JOB = 1. The actual pivot sequence used may differ slightly from that of this earlier call if $\mathbf{A}$ is not diagonally dominant.

(iii) JOB = 3 uses the factors generated by a JOB = 2 call to solve a system of equations $\mathbf{Ax=b}$ or $\mathbf{A}^T\mathbf{x=b}$.

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

Other values for the parameter JOB can invoke combinations of these three basic operations (see Section 2.1.2).

2.1.1 To set default values of controlling parameters

The single precision version

        CALL MA41I(CNTL,ICNTL,KEEP)

The double precision version

        CALL MA41ID(CNTL,ICNTL,KEEP)

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

ICNTL is an INTEGER array of length 20 that need not be set by the user. On return it contains default values. For further information see Section 2.2.

KEEP is an INTEGER array of length 50 that need not be set by the user. MA41I/ID sets default values for some of the components of KEEP. This array must be preserved by the user between calls to MA41A/AD.

2.1.2 To perform matrix factorization and solution

The single precision version

        CALL MA41A (JOB, N, NE, IRN, JCN, ASPK, RHS, COLSCA, ROWSCA,
       *   KEEP, IS, MAXIS, S, MAXS, CNTL, ICNTL, INFO, RINFO)

The double precision version

        CALL MA41AD (JOB, N, NE, IRN, JCN, ASPK, RHS, COLSCA, ROWSCA,
       *   KEEP, IS, MAXIS, S, MAXS, CNTL, ICNTL, INFO, RINFO)

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

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

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

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

IRN and JCN are INTEGER arrays of length NE. IRN(K) and JCN(K), K=1, ... NE must be set by the user to hold the row and column indices, respectively, for the matrix entries. They are not altered by the subroutine except when a permutation for a zero-free diagonal is requested (ICNTL(6) ≠ 0).

ASPK is a REAL (DOUBLE PRECISION in the D version) array of length NE. The user must set ASPK(K) to the value of the entry in row IRN(K) and column JCN(K) of the matrix. It is not altered by the subroutine.

RHS is a REAL (DOUBLE PRECISION in the D version) array of length N that is only accessed when JOB = 3, 5, or 6. On entry, RHS(I) must hold the I th component of the right-hand side of the equations being solved. On exit, RHS(I) will hold the I th component of the solution vector. For other values of JOB, RHS is not accessed and can be declared to have size one.

COLSCA, ROWSCA are REAL (DOUBLE PRECISION in the D version) arrays of length N that are used to hold scaling

factors for the columns and rows of the original matrix, respectively. These arrays need to be set by the user only if `ICNTL(8) = -1`. If `ICNTL(8)=0`, `COLSCA` and `ROWSCA` are not accessed and so can be declared to have size one. For any other values of `ICNTL(8)`, the scaling arrays are computed before numerical factorization. The factors of the scaled matrix diag(`ROWSCA(i)`) **A** diag(`COLSCA(i)`) are computed.

`KEEP` is an `INTEGER` array of length `50` that need not be set by the user. It must not be changed since the call to `MA41I/ID` and must be preserved between calls to `MA41A/AD` for the same matrix.

`IS` is an `INTEGER` array of length `MAXIS`. It need not be set by the user unless the ordering is provided by the user (`ICNTL(7)=1`). In this case, `IS(I)`, `I=1, ... N` holds the position of variable `I` in the pivot order. Note that, even when the ordering is provided by the user, the analysis must be performed before numerical factorization. `IS` must be preserved between calls to `MA41A/AD`.

`MAXIS` is an `INTEGER` variable that must be set by the user to the size of the array `IS`. If the ordering is provided by the user (`ICNTL(7)=1`), then value of `MAXIS` to run the analysis phase is `NE+12*N+1`. If the ordering is determined by the analysis phase, then the minimum value of `MAXIS` to run the analysis phase is `2*NE+11*N+1`. For numerical factorization, `MAXIS` must be greater than `INFO(7)` as output from the analysis phase on the same matrix. It is not altered by the subroutine.

`S` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `MAXS`. It need not be set by the user but must be preserved between successive calls to `MA41A/AD`.

`MAXS` is an `INTEGER` variable that must be set by the user to the size of the working array `S`. It is not accessed in the analysis phase (`JOB=1`). After the analysis phase, an indication of the minimum space required to run the numerical factorization is available in `INFO(8)`. Increasing the value of `MAXS` can improve the performance of the factorization phase in a multiprocessor environment. It is not altered by the subroutine.

`CNTL` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `10` that contains control parameters and must be set by the user. Default values for the components may be set by a call to `MA41I/ID`. Details of the control parameters are given in Section 2.2.

`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 `MA41I/ID`. Details of the control parameters are given in Section 2.2.

`INFO` is an `INTEGER` array of length `20` that need not be set by the user. On return from `MA41A/AD`, a non-negative value for `INFO(1)` indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3. For the meaning of the value of other components of `INFO`, see Section 2.2.

`RINFO` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `20` that need not be set by the user. This array supplies information on the execution of `MA41A/AD`. For the meaning of the value of the components of `RINFO`, see Section 2.2.

2.2 Arrays for control and information

The components of the arrays `CNTL` and `ICNTL` control the action of `MA41A/AD`. Default values are set by `MA41I/ID`. Only the first two components of `CNTL` and the first `11` components of `ICNTL` are presently used by `MA41A/AD`. The remaining components are set to zero by `MA41I/ID` and are not accessed in `MA41A/AD`. The components of the arrays `RINFO` and `INFO` provide information on the action of `MA41A/AD`.

`CNTL(1)` has default value 0.01 and is used for threshold pivoting. If `CNTL(1)` is nonzero, numerical pivoting will be performed. In general, a larger value of `CNTL(1)` leads to greater fill-in but a more accurate factorization. If `CNTL(1)` is zero, no pivoting will be performed and the subroutine will fail if a zero pivot is encountered. If the matrix **A** is diagonally dominant, then setting `CNTL(1)` to zero will decrease the factorization time while still providing a stable decomposition. Values greater than 1.0 are treated as 1.0, and less than zero as zero.

`CNTL(2)` must be set to the tolerance for the convergence of iterative refinement. The default value for `MA41A` is 10^{-5} and for `MA41AD` is 10^{-10}.

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

`ICNTL(2)` has default value 0. It is the output stream for diagnostic printing and for warning messages. If it is negative, these messages are suppressed.

`ICNTL(3)` has default value 0. It is the output stream for the printing of statistics. If it is negative, the statistics are not printed.

`ICNTL(4)` is used by `MA41A/AD` 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).

        4   Errors and warnings and all information on input and output parameters printed.

`ICNTL(5)` has default value 1 and is equal to the number of processors. On a uniprocessor machine, the user is strongly encouraged not to change the value of `ICNTL(5)` from its default value since the storage management is quite different for the multiprocessor version and could be very inefficient on a uniprocessor.

`ICNTL(6)` has default value 0 and is only accessed and operational on a call that includes an analysis phase (JOB = 1, 4, or 6). In these cases, if `ICNTL(6)=1`, a maximum transversal algorithm is applied to the original matrix. Column permutations are then applied to the original matrix to get a zero-free diagonal. If the permutation obtained is not the identity matrix then the minimum value of `MAXIS` to run the analysis is `2*NE+12*N+1`. The user is advised to set `ICNTL(6)=1` only when the matrix is very unsymmetric. If the ordering is provided by the user (`ICNTL(7)=1`), then the value of `ICNTL(6)` is ignored.

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

`ICNTL(8)` has default value 0 and is used to describe the scaling strategy. Note that scaling is performed only when the numerical factorization step is performed (JOB = 2, 4, 5, or 6). If `ICNTL(8)` is not equal to any of the values listed below then `ICNTL(8)` is treated as if it had its default value of 0 (no scaling). If the matrix is known to be very badly scaled, our experience has been that option 6 is the most robust but the best scaling is very problem dependent. Possible values of `ICNTL(8)` are:

       –1 the user must provide the scaling in arrays `COLSCA` and `ROWSCA`

       0 no scaling

       1 Diagonal scaling

       2 Scaling based on HSL Subroutine MC29 and further discussed in Curtis and Reid *J. Inst. Maths. Applics.* **10** 118-124 (1972)

       3 Column scaling

       4 Row and column scaling

       5 Scaling based on MC29 followed by column scaling

       6 Scaling based on MC29 followed by row and column scaling.

    If `ICNTL(8)=0`, `COLSCA` and `ROWSCA` are not accessed.

`ICNTL(9)` has default value 1. If `ICNTL(9)=1` the system of equations $\mathbf{A}\mathbf{x}=\mathbf{b}$ is solved. For other values the system $\mathbf{A}^T\mathbf{x}=\mathbf{b}$ is solved.

`ICNTL(10)` has default value 0. It corresponds to the maximum number of steps of iterative refinement. If `ICNTL(10)=0`, iterative refinement is not performed. Values of `ICNTL(10) < 0` are treated as zero.

`ICNTL(11)` has default value 0. A positive value will return the infinite norm of the input matrix, the computed solution, and the scaled residual in `RINFO(4)` to `RINFO(6)`, respectively, a backward error estimate in `RINFO(7)` and `RINFO(8)`, and an estimate of the forward error in `RINFO(9)`. If `ICNTL(11)` is negative or zero no estimates are returned.

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

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

Statistics produced after analysis phase

`INFO(3)` Estimated real space needed for factors.

`INFO(4)` Estimated integer space needed for factors.

`INFO(5)` Estimated maximum frontal size.

`INFO(6)` Number of nodes in the tree.

`INFO(7)` Minimum value of `MAXIS` estimated by the analysis phase to run the numerical factorization.

`INFO(8)` Minimum value of `MAXS` estimated by the analysis phase to run the numerical factorization.

Statistics produced after factorization

`INFO(9)` Size of the real space used to store the LU factors.

`INFO(10)` Size of the integer space used to store the LU factors.

`INFO(11)` Order of largest frontal matrix.

`INFO(12)` Number of off-diagonal pivots.

`INFO(13)` Number of uneliminated variables sent to the father.

`INFO(14)` Number of memory compresses.

Statistics produced after solution

`INFO(15)` Number of solution steps in iterative refinement.

`INFO(16)` to `INFO(20)` are not currently used by `MA41A/AD`.

Statistics produced after analysis phase

`RINFO(1)` The estimated number of floating-point operations to perform the factorization.

Statistics produced after factorization

`RINFO(2)` Number of floating-point operations for the assembly process.

`RINFO(3)` Number of floating-point operations for the elimination process.

Statistics produced after solve with error analysis

`RINFO(4)` Infinite norm of the input matrix, where

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

`RINFO(5)` Infinite norm of the computed solution, where

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

`RINFO(6)` Norm of scaled residuals

$$\frac{\max_{i=1,n} \left| \sum_{j=1}^{n} a_{ij} x_j - b_i \right|}{\| \mathbf{A} \|_{\infty} \ \| \mathbf{x} \|_{\infty}} \ .$$

`RINFO(7)`, `RINFO(8)` and `RINFO(9)` are used to hold information on the backward error. See Section 2.4.

`RINFO(10)` to `RINFO(20)` are not currently used by `MA41A/AD`.

2.3 Error diagnostics

A successful return from `MA41A/AD` is indicated by a value of `INFO(1)` equal to zero.

A negative value of `INFO(1)` is associated with an error message which will be output on unit `ICNTL(1)`.

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

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

–3 `JOB` has a wrong value or analysis was not performed prior to the factorization. Value given is held in `INFO(2)`.

–4 Error in permutation array when `ICNTL(7)=1`.

–5 Not enough space to preprocess the input matrix. `MAXS` must be increased to at least `INFO(2)`.

–6 The matrix is structurally singular. The estimated structural rank of the matrix is given in `INFO(2)`.

–7 Error return from analysis. `MAXIS` is too small to run the analysis and must be increased. `INFO(2)` holds the minimum size of the corresponding working array to perform analysis.

–8 Error return from numerical factorization. `MAXIS` is too small and must be increased to at least `INFO(2)`.

–9 Error return from numerical factorization. `MAXS` is too small and must be increased to at least `INFO(2)`.

–10 The matrix is numerically singular. `INFO(2)` holds an estimate of the rank of the matrix.

–11 Error return from solution phase. `MAXS` is too small and must be increased to at least `INFO(2)`.

–12 Not enough space to postprocess the solution (iterative refinement and/or error analysis phase). `MAXS` is too small and must be increased to at least `INFO(2)`. This failure occurs after a solution has been calculated so it is possible that the solution returned in `RHS` is satisfactory.

A positive value of `INFO(1)` is associated with a warning message which will be output on unit `ICNTL(2)`.

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

+2 During error analysis the max-norm of the computed solution was found to be zero.

+4 Real working array used during numerical factorization might be increased to at least `INFO(2)` to improve performance.

+8 Warning return from the iterative refinement routine. More than `ICNTL(10)` iterations are required.

+x x=3,5,6,7,9,10,11,12,13,14,15 are combinations of the above warnings corresponding to summing the constituent warnings.

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 $\delta\mathbf{A}_{ij} \leq \max(\texttt{RINFO(7)},\texttt{RINFO(8)})|\mathbf{A}|_{ij}$ and $\delta\mathbf{b}_i \leq \max(\texttt{RINFO(7)}|\mathbf{b}|_i, \texttt{RINFO(8)}\|\mathbf{A}_i\|_\infty \|\bar{\mathbf{x}}\|_\infty)$. Note that $\delta\mathbf{A}$ respects the sparsity in $\mathbf{A}$. An upper bound for the forward error is returned in `RINFO(9)`.

Reference

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

3 GENERAL INFORMATION

Use of common: None.

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

Other routines called directly:

BLAS **routines:** XERBLA, LSAME, SAXPY/DAXPY, SSWAP/DSWAP, ISAMAX/IDAMAX, SGER/DGER, STRSV/DTRSV, SGEMV/DGEMV, SGEMM/DGEMM, STRSM/DTRSM (Fortran version of the BLAS routines can be provided).

HSL routines: MC21, MC29, and MC47.

All the other subroutines called by the principal subroutines are in the MA41 package. They are:
MA41B/BD, MA41C/CD, MA41D/DD, MA41E/ED, MA41F/FD, MA41G/GD, MA41H/HD, MA41J/JD, MA41K/KD, MA41L/LD, MA41M/MD, MA41N/ND, MA41O/OD, MA41P/PD, MA41Q/QD, MA41R/RD, MA41S/SD, MA41T/TD, MA41U/UD, MA41V/VD, MA41W/WD, MA41X/XD, MA41Y/YD. MA41Z/ZD, MC51A/AD, MC51B/BD, MC51C/CD, MC51D/DD, MC51E/ED, MC51F/FD, MC51G/GD, MC51H/HD, MC51I/ID, MC51J/JD, MC51K/KD, MC51L/LD, MC51M/MD, MC51N/ND, MC51O/OD, MC51P/PD, MC51Q/QD, MC51R/RD, MC51S/SD, MC51T/TD, MC51U/UD, MC51V/VD, MC51W/WD, MC51X/XD, MC51Y/YD, MC51Z/ZD. At present, MA41Z/ZD is a dummy routine that might be used in a later version of the code.

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

Restrictions: N $\geq$ 1, NE $\geq$ 1.

4 METHOD

A multifrontal version of sparse Gaussian elimination is used.

The `JOB=1` entry entry chooses pivots from the diagonal using the Approximate Minimum Degree algorithm of Amestoy, Davis and Duff (An approximate minimum degree ordering algorithm, Technical Report TR-94-039, Computer and Information Sciences Department, University of Florida, Gainesville, FL, USA, 1994) and employing a generalized element model of the elimination thus avoiding the need to store the filled-in pattern explicitly. The elimination is represented as an assembly and elimination tree.

The `JOB=2` entry factorizes the matrix by using the assembly and elimination tree generated by a `JOB=1` call. At each stage in the multifrontal approach, pivoting and elimination are performed on full submatrices and partial pivoting with a threshold criterion is used to control the growth in the factors.

The `JOB=3` entry uses the factors from the `JOB=2` entry to solve systems of equations either by loading the appropriate parts of the vectors into an array of the current frontsize and using full matrix code or by indirect addressing at each stage, whichever performs better.

In addition, `MA41A/AD` offers pre-processing and post-processing facilities. Permutations for a zero-free diagonal (Duff, ACM Trans. Math. Softw. **7** 387-330, 1981) can be applied to very unsymmetric matrices and can help reduce fill-in. Prescaling of the input matrix can help reduce fill-in during factorization can improve the numerical accuracy. A range of classical scalings are provided for the user and can be automatically performed before numerical factorization. Iterative refinement can be optionally performed after the solution step. Arioli, Demmel, and Duff, *SIAM J. Matrix Anal. and Applics.* **10**, 165-190 (1989) have shown that with only two to three steps of iterative refinements the solution can often be significantly improved. Finally `MA41A/AD` also enables the user to perform classical error analysis based on the residuals.

The multifrontal approach is discussed in detail by Duff and Reid, *SIAM J. Sci. Stat. Comput.* **5** (1984), 633-641 and Duff, *Parallel Computing* **3** 193-204 (1986). The multiprocessor version of the code described here is based on work by Duff, *Parallel Computing* **3** 193-204 (1986), Duff, *J. Comput. Appl. Math.* **27** 229-239 (1989), Amestoy and Duff, *Int. J. Supercomputer Applics.* **3** (3) 41-59 (1989), Amestoy, CERFACS Report TH/PA/91/2 (1990), and Amestoy and Duff, *Int. J. Supercomputer Applics.* **7** (1) (1993).

5 EXAMPLES OF USE

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

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

The driver `MA41A/AD` is used with `JOB=6` and the solution and error estimates are printed. Note that this example does not illustrate all the facilities.

Program

```
C
C Simple example of use of MA41 Package
C
      PROGRAM MA41ST

C Set array dimensions
C NMAX   = Maximum order of matrix
C NZMAX  = Maximum number of entries in matrix
```

```
C MAXSP  = Maximum size of the REAL working space
C MAXIW  = Maximum size of the INTEGER working space
      INTEGER NMAX, NZMAX, MAXSP, MAXIW
      PARAMETER (NMAX=10,NZMAX=20,MAXSP=800,MAXIW=800)

C Declare MA41 parameters
      INTEGER JOB, N, NZ, IRN(NZMAX), JCN(NZMAX)
      DOUBLE PRECISION ASPK(NZMAX), RHS(NMAX), COLSCA(NMAX),ROWSCA(NMAX)
      INTEGER KEEP(50), IW(MAXIW)
      DOUBLE PRECISION S(MAXSP), CNTL(10)
      INTEGER ICNTL(20), INFO(20)
      DOUBLE PRECISION RINFO(20)

C Local variables
      INTEGER I, INFILE, LP

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

C Read matrix and right-hand side
      READ (INFILE,*) N,NZ
      READ (INFILE,*) (IRN(I),I=1,NZ)
      READ (INFILE,*) (JCN(I),I=1,NZ)
      READ (INFILE,*) (ASPK(I),I=1,NZ)
      READ (INFILE,*) (RHS(I),I=1,N)

C------------------------------------
C Initialization of control parameters
C------------------------------------
      CALL MA41ID (CNTL, ICNTL, KEEP)
C Request error analysis and iterative refinement
      ICNTL(10) = 10
      ICNTL(11) = 1

C---------------------------------------------------------
C Analyse sparsity pattern using minimum degree ordering
C Scale (column scaling) and factorize matrix
C Solve the equations, and perform error analysis
C---------------------------------------------------------
      JOB = 6
      CALL MA41AD (JOB, N, NZ, IRN, JCN, ASPK, RHS,
     *             COLSCA, ROWSCA,KEEP, IW, MAXIW, S, MAXSP,
     *             CNTL, ICNTL, INFO, RINFO)

C Print error and residual
      WRITE(LP,'(A/(10F8.4))') 'Solution is:',(RHS(I),I=1,N)
      WRITE(LP,'(A,F8.4)') 'Backward error is (class 1)',RINFO(7)
      WRITE(LP,'(A,F8.4)') 'Backward error is (class 2)',RINFO(8)
      WRITE(LP,'(A,F8.4)') 'Forward error is           ',RINFO(9)

C Error if INFO(1) is negative
      IF (INFO(1).LT.0) WRITE(LP,'(A)') ' Error in MA41AD'

      STOP
      END
```

Data

```
  5   12
```

```
   1    2    4    5    2    1    5    3    2    3    1    3
   2    3    3    5    1    1    2    4    5    2    3    3
 3.0 -3.0  2.0  1.0  3.0  2.0  4.0  2.0  6.0 -1.0  4.0  1.0
20.0 24.0  9.0  6.0 13.0
```

Output
```
Solution is:
  1.0000  2.0000  3.0000  4.0000  5.0000
Backward error is (class 1)  0.0000
Backward error is (class 2)  0.0000
Forward error is             0.0000
```