Warning: Subroutine `MA30` has been superseded by subroutine `MA50` which uses improved algorithms; the use of the latter routine is recommended. The superseded routine may be removed from later releases of the library.

1 SUMMARY

These subroutines perform operations pertinent to the solution of a general sparse $n \times n$ system of linear equations

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i=1, 2,..., n,$$

(i.e. solve $\mathbf{Ax}=\mathbf{b}$). Structurally singular matrices are permitted, including those with rows or columns consisting entirely of zeros (i.e. including rectangular matrices). It is assumed that the nonzero entries a_{ij} do not differ widely in size. If necessary, a prior call of the scaling subroutine `MC19A/AD` may be made.

(a) `MA30A/AD` performs the **LU** decomposition of a user-input matrix **A**. The user can define input permutations $\mathbf{P}_1$ and $\mathbf{Q}_1$ so that $\mathbf{P}_1\mathbf{AQ}_1$ is in block lower triangular form, in which case only the diagonal blocks are factorized and any permutations for pivoting are confined to the diagonal blocks. In this case, in the resulting factorization

$$\mathbf{PAQ} = \mathbf{LU},$$

P and **Q** differ form $\mathbf{P}_1$ and $\mathbf{Q}_1$ only within the diagonal blocks. The input permutations $\mathbf{P}_1$ and $\mathbf{Q}_1$ may be found by calling `MC23A/AD` or the matrix may be treated as a single block by using $\mathbf{P}_1=\mathbf{Q}_1=\mathbf{I}$. The matrix entries are held compactly by rows, although it should be noted that the user can supply the matrix by columns to get the **LU** decomposition of $\mathbf{A}^T$.

(b) `MA30B/BD` performs the **LU** decomposition of the diagonal blocks of a new matrix **PAQ** of the same sparsity pattern and the same **P** and **Q**, using information from a previous call to `MA30A/AD`. The entries of the input matrix must already be in their final positions in the **LU** decomposition structure. This routine typically executes about five times faster than `MA30A/AD`.

(c) `MA30C/CD` uses the factors produced by `MA30A/AD` or `MA30B/BD` to solve $\mathbf{Ax}=\mathbf{b}$ or $\mathbf{A}^T\mathbf{x}=\mathbf{b}$.

If the user requires a more convenient data interface, the `MA28` package should be used. The `MA28` subroutines call the `MA30` subroutines after checking the user's input data and optionally using `MC23A/AD` to permute the matrix to block triangular form.

ATTRIBUTES — **Version:** 1.0.0. **Remark:** Normally called through the `MA28` package. **Types:** `MA30A`, `MA30AD`. **Original date:** April 1977 (modified February 1983). **Origin:** I. S. Duff, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Argument lists

We first describe the argument list for `MA30A/AD`. This description should also be consulted for further information on most of the parameters of `MA30B/BD` and `MA30C/CD`.

2.1.1 Argument list for MA30A/AD

The single precision version

```
        CALL MA30A(N,ICN,A,LICN,LENR,LENRL,IDISP,IP,IQ,IRN,LIRN,
       *           LENC,IFIRST,LASTR,NEXTR,LASTC,NEXTC,
       *           IPTR,IPC,U,IFLAG)
```

The double precision version

```
        CALL MA30AD(N,ICN,A,LICN,LENR,LENRL,IDISP,IP,IQ,IRN,LIRN,
       *           LENC,IFIRST,LASTR,NEXTR,LASTC,NEXTC,
       *           IPTR,IPC,U,IFLAG)
```

`N` is an `INTEGER` variable which must be set by the user to the order of the matrix. `N` is not altered by `MA30A/AD`.

`ICN` is an `INTEGER` array of length `LICN`. Positions `IDISP(2)` to `LICN` must be set by the user to contain the column indices of the entries in the diagonal blocks of $\mathbf{P}_1\mathbf{AQ}_1$. Those belonging to a single row must be contiguous, but the ordering of column indices within each row is unimportant. The entries of row i precede those of row i+1, i=1,..., n–1 and no wasted space is allowed between the rows. On output the column indices of the **LU** decomposition of **PAQ** are held in positions `IDISP(1)` to `IDISP(2)`, the rows are in pivotal order, and the column indices of the **L** part of each row are in pivotal order and precede those of **U**. Again there is no wasted space either within a row or between the rows. `ICN(1)` to `ICN(IDISP(1)-1)` are neither required nor altered. If `MC23A/AD` has been called, these will hold information about the off-diagonal blocks.

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LICN` whose entries `IDISP(2)` to `LICN` must be set by the user to the values of the nonzero entries of the matrix in the order indicated by `ICN`. On output `A` will hold the **LU** factors of the matrix, where again the position in the matrix is determined by the corresponding values in `ICN`. `A(1)` to `A(IDISP(1)-1)` are neither required nor altered.

`LICN` is an `INTEGER` variable which must be set by the user to the length of arrays `ICN` and `A`. It must be big enough for `A` and `ICN` to hold all the entries of **L** and **U** and leave some elbow room. It is possible to calculate a minimum value for `LICN` by a preliminary run of `MA30A/AD` (see `MINICN`, section 2.2.1). The adequacy of the elbow room can be judged by the size of the common block variable `ICNCP` (see section 2.2.1). `LICN` is not altered by `MA30A/AD`.

`LENR` is an `INTEGER` array of length `N`. On input, `LENR(i)` should equal the number of entries in row `i`, `i`=1,..., `N` of the diagonal blocks of $\mathbf{P}_1\mathbf{AQ}_1$. On output, `LENR(i)` will equal the total number of entries in row `i` of **L** and row `i` of **U** (excluding the diagonal in the case of **L**).

`LENRL` is an `INTEGER` array of length `N` which need not be set by the user. On output from `MA30A/AD`, `LENRL(i)` will hold the number of entries in row `i` of **L**, excluding the diagonal.

`IDISP` is an `INTEGER` array of length 2. The user should set `IDISP(1)` to be the first available position in the array pair `A/ICN` for the **LU** decomposition and `IDISP(2)` to the position in `A/ICN` of the first entry in the diagonal blocks of $\mathbf{P}_1\mathbf{AQ}_1$. On output, `IDISP(1)` will be unaltered while `IDISP(2)` will be set to the position in `A/ICN` of the last entry of the **LU** decomposition.

`IP` is an `INTEGER` array of length `N`. On input to `MA30A/AD`, the absolute value of `IP(i)` must be set to the row of **A** which is row `i` of $\mathbf{P}_1\mathbf{AQ}_1$. A negative value for `IP(i)` indicates that row `i` is at the end of a diagonal block. On output from `MA30A/AD`, `IP(i)` indicates the row of **A** that is row `i` of **PAQ**. `IP(i)` will still be negative for the last row of each block (except the last).

IQ is an INTEGER array of length N. On input to MA30A/AD, IQ(j) must be set to the column of **A** which is column j of $\mathbf{P}_1\mathbf{A}\mathbf{Q}_1$. On output from MA30A/AD, the absolute value of IQ(j) indicates the column of **A** that is column j in **PAQ**. For a row, i say, in which a zero pivot is detected, IQ(i) is negated (see section 2.6.1).

IRN is an INTEGER array of length LIRN used as workspace by MA30A/AD.

LIRN is an INTEGER variable which must be set to the length of IRN and is not altered. It must be greater than the largest number of entries in a diagonal block of $\mathbf{P}_1\mathbf{A}\mathbf{Q}_1$ but need not be as large as LICN. It should be large enough to hold the active part of any block, plus some elbowroom, the *a posteriori* adequacy of which can be estimated by examining the size of common block variable IRNCP (see section 2.2.1).

LENC,IFIRST,LASTR,NEXTR,LASTC,NEXTC are all INTEGER arrays of length N which are used as workspace by MA30A/AD. If NSRCH (see section 2.2.1) is set to a value less than or equal to N, arrays LASTC and NEXTC are not referenced by MA30A/AD and so can be dummied in the call to MA30A/AD.

IPTR,IPC are INTEGER arrays of length N which are used as workspace by MA30A/AD.

U is a REAL (DOUBLE PRECISION in the D version) variable which should be set by the user to a value between 0. and 1.0. If less than zero it is reset to zero and if its value is 1.0 or greater it is reset to 0.9999 (0.999999999 in D version). It determines the balance between pivoting for sparsity and for stability, values near zero emphasizing sparsity and values near one emphasizing stability. We recommend U=0.1 as a possible first trial value. The stability can be judged by a later call to MC24A/AD or by setting LBIG (see section 2.2) to .TRUE..

IFLAG is an INTEGER variable. It will have a non-negative value if MA30A/AD is successful. Negative values indicate error conditions while positive values indicate that the matrix has been successfully decomposed but is singular (see section 2.3.1).

2.1.2 Argument list for MA30B/BD

The single precision version

```
      CALL MA30B(N,ICN,A,LICN,LENR,LENRL,IDISP,IP,IQ,W,IW,IFLAG)
```

The double precision version

```
      CALL MA30BD(N,ICN,A,LICN,LENR,LENRL,IDISP,IP,IQ,W,IW,IFLAG)
```

N is an INTEGER variable set to the order of the matrix. It is not altered.

ICN is an INTEGER array of length LICN. It should be unchanged since the last call to MA30A/AD. It is not altered by MA30B/BD.

A is a REAL (DOUBLE PRECISION in the D version) array of length LICN. The user must set entries IDISP(1) to IDISP(2) to contain the entries in the diagonal blocks of the matrix **PAQ** whose column numbers are held in ICN, using corresponding positions. Note that some zeros may need to be held explicitly. On output, entries IDISP(1) to IDISP(2) of array A contain the **LU** decomposition of the diagonal blocks of **PAQ**. Entries A(1) to A(IDISP(1)-1) are neither required nor altered by MA30B/BD.

LICN is an INTEGER variable which must be set by the user to the length of arrays A and ICN. It is not altered by MA30B/BD.

LENR,LENRL are INTEGER arrays of length N. They should be unchanged since the last call to MA30A/AD. They are not altered by MA30B/BD.

IDISP is an INTEGER array of length 2. It should be unchanged since the last call to MA30A/AD. It is not altered by MA30B/BD.

IP,IQ are INTEGER arrays of length N. They should be unchanged since the last call to MA30A/AD. They are not altered by MA30B/BD.

W is a REAL (DOUBLE PRECISION in the D version) array of length N which is used as workspace by MA30B/BD.

IW is an INTEGER array of length N which is used as workspace by MA30B/BD.

IFLAG is an INTEGER variable. On output from MA30B/BD, IFLAG has the value zero if the factorization was successful, has the value I if pivot I was very small (see section 2.3.2) and has the value -I if an unexpected singularity was detected at stage I of the decomposition (see section 2.3.2).

2.1.3 Argument list for MA30C/CD

The single precision version

```
      CALL MA30C(N,ICN,A,LICN,LENR,LENRL,
     *           LENOFF,IDISP,IP,IQ,X,W,MTYPE)
```

The double precision version

```
      CALL MA30CD(N,ICN,A,LICN,LENR,LENRL,
     *            LENOFF,IDISP,IP,IQ,X,W,MTYPE)
```

N is an INTEGER variable set to the order of the matrix. It is not altered by the subroutine.

ICN is an INTEGER array of length LICN. Entries IDISP(1) to IDISP(2) should be unchanged since the last call to MA30A/AD. If the matrix has more than one diagonal block, then column indices corresponding to entries in subdiagonal blocks of **PAQ** must appear in positions 1 to IDISP(1)-1. For each row these entries must be contiguous, with those in row i preceding those in row i+1 (i=1,..., N-1) and no wasted space between rows. Entries may be in any order within each row. ICN is not altered by MA30C/CD.

A is a REAL (DOUBLE PRECISION in the D version) array of length LICN. Entries IDISP(1) to IDISP(2) should be unchanged since the last call to MA30A/AD or MA30B/BD. If the matrix has more than one diagonal block, then the values of the entries in subdiagonal blocks must be in positions 1 to IDISP(1)-1 in the order given by ICN. It is not altered by MA30C/CD.

LICN is an INTEGER variable set to the size of arrays ICN and A. It is not altered by MA30C/CD.

LENR,LENRL are INTEGER arrays of length N which should be unchanged since the last call to MA30A/AD. They are not altered by MA30C/CD.

LENOFF is an INTEGER array of length N. If the matrix **PAQ** (or $\mathbf{P}_1\mathbf{AQ}_1$) has more than one diagonal block, then LENOFF(i), i=1,..., N should be set to the number of entries in row i of the matrix **PAQ** which are in subdiagonal blocks. If there is only one diagonal block then LENOFF(1) may be set to –1, in which case the other entries of LENOFF are never accessed. LENOFF is not altered by MA30C/CD.

IDISP is an INTEGER array of length 2 which should be unchanged since the last call to MA30A/AD. It is not altered by MA30C/CD.

IP,IQ are INTEGER arrays of length N which should be unchanged since the last call to MA30A/AD. They are not altered by MA30C/CD.

X is a REAL (DOUBLE PRECISION in the D version) array of length N. It must be set by the user to the values of the right-hand side vector **b** for the equations being solved. On exit from MA30C/CD it will be equal to the solution **x**.

W is a REAL (DOUBLE PRECISION in the D version) array of length N which is used as workspace by MA30C/CD.

MTYPE is an INTEGER variable which must be set by the user. If MTYPE=1, the solution to the system **Ax**=**b** is returned; any other value for MTYPE will return the solution to the system $\mathbf{A}^T\mathbf{x}=\mathbf{b}$. MTYPE is not altered by MA30C/CD.

2.2 Common blocks

Some of the common block variables have default values. These are set by BLOCK DATA MA30J/JD.

2.2.1 Common blocks used by MA30A/AD

The variables in common block MA30E/ED and MA30I/ID (used also by MA30B/BD) are used as controlling parameters. The variables in common block MA30F/FD are used to provide the user with information on the decomposition.

The single precision version

```
        COMMON/MA30E/LP,ABORT1,ABORT2,ABORT3
        COMMON/MA30F/IRNCP,ICNCP,IRANK,MINIRN,MINICN
        COMMON/MA30I/TOL,BIG,NDROP,NSRCH,LBIG
```

The double precision version

```
        COMMON/MA30ED/LP,ABORT1,ABORT2,ABORT3
        COMMON/MA30FD/IRNCP,ICNCP,IRANK,MINIRN,MINICN
        COMMON/MA30ID/TOL,BIG,NDROP,NSRCH,LBIG
```

LP is an INTEGER variable holding the unit number to which the error messages (described in section 2.3) are sent. LP has a default value of 6. This default value can be reset by the user, if desired. A value of 0 suppresses all messages.

ABORT1,ABORT2,ABORT3 are LOGICAL variables that are used to control the conditions under which MA30A/AD will terminate. If ABORT1 is .TRUE., the subroutine will exit immediately on detecting structural singularity. If ABORT2 is .TRUE., the subroutine will exit immediately on detecting numerical singularity. If ABORT3 is .TRUE., the subroutine will exit immediately the available space in the array pair A/ICN is filled up by the previously decomposed, active, and undecomposed parts of the matrix. The default values for ABORT1,ABORT2,ABORT3 are set to .TRUE., .TRUE. and .FALSE., respectively.

IRNCP,ICNCP are INTEGER variables used to monitor the adequacy of the allocated space in arrays IRN and the array pair A/ICN respectively, by counting the number of data management compresses performed on these arrays. If IRNCP or ICNCP is fairly large (say greater than N/10), it may be advantageous to increase the size of the corresponding array(s). IRNCP and ICNCP are initialized to zero on entry to MA30A/AD and one of them is incremented each time the compressing routine MA30D/DD (see section 3) is entered.

IRANK is an INTEGER variable which gives an estimate (actually an upper bound) of the rank of the matrix (see section 2.6.1). On an exit with IFLAG equal to 0, this will be equal to N.

MINIRN is an INTEGER variable which, after a successful call to MA30A/AD, indicates the minimum length to which IRN could be reduced while still permitting a successful decomposition of the given matrix. If, however, the user were to decrease the length of IRN to that size, the number of compresses (IRNCP) may be very high and quite costly. If LIRN is not large enough to begin the decomposition on a diagonal block, MINIRN will be equal to the value required to continue the decomposition and IFLAG will be set to –3 or –6 (see section 2.3.1). A value of LIRN slightly greater than this (say about N/2) will usually provide enough space to complete the decomposition on that block. In the event of any other failure MINIRN gives the minimum size of IRN required for a successful decomposition up to the failure.

MINICN is an INTEGER variable which after a successful call to MA30A/AD, indicates the minimum size of LICN required to enable a successful decomposition. In the event of failure with IFLAG= –5 (see section 2.3.1), MINICN will, if ABORT3 is left set to .FALSE. (see beginning of this section), indicate the minimum length that would be sufficient to prevent this error in a subsequent run on an identical matrix. Again the user may prefer to use a value of ICN slightly greater than MINICN for subsequent runs to avoid too many compresses (ICNCP). In the event of failure with IFLAG equal to any negative value except –4, MINICN will give the minimum length to which LICN could be reduced to enable a successful decomposition to the point at which failure occurred. Notice that, on a successful entry, IDISP(2) gives the amount of space in A/ICN required for the decomposition while MINICN will usually be slightly greater because of the need for elbow room. If the user is very unsure how large to make LICN, the variable MINICN can be used to provide that information. A preliminary run should be performed with ABORT3 left set to .FALSE. and LICN about 1½ times as big as the number of entries in the original matrix. Unless the initial problem is very sparse (when the run will be successful) or fills in extremely badly (giving an error return with IFLAG equal to –4), an error return with IFLAG equal to –5 should result and MINICN will give the amount of space required for a successful decomposition.

TOL is a REAL (DOUBLE PRECISION in the D version) variable. If it is set to a positive value, MA30A/AD will drop from the factors any entry whose modulus is less than TOL. The factorization will then require less storage but will be inaccurate. After a run of MA30A/AD where entries have been dropped, MA30B/BD should not be called. The default value for TOL is 0.0.

BIG is a REAL (DOUBLE PRECISION in the D version) variable. If LBIG has been set to .TRUE., BIG will be set to the largest entry encountered during the factorization. Since BIG is initialized at compile time only to a default value of zero, it should be reset between different runs of MA30A/AD within the same program. Note that, if MA30 is used through the interface MA28, then BIG is initialized in MA28 to the largest entry in the input matrix.

NDROP is an INTEGER variable. If TOL has been set positive, on exit from MA30A/AD, NDROP will hold the number of entries dropped from the data structure.

NSRCH is an INTEGER variable. If NSRCH is set to a value less than or equal to N, the pivot search is limited to NSRCH rows. This may result in different fill-in and execution time for MA30A/AD. If NSRCH is less than or equal to N, the workspace arrays LASTC and NEXTC (see section 2.1) are not referenced by MA30A/AD. The default value for NSRCH is 32768.

LBIG is a LOGICAL variable. If LBIG is set to .TRUE., the value of the largest entry encountered in the factorization by MA30A/AD is returned in BIG. Setting LBIG to .TRUE. will marginally increase the factorization time. The default value for LBIG is .FALSE..

2.2.2 Common blocks used by MA30B/BD

The single precision version

```
COMMON/MA30E/LP,ABORT1,ABORT2,ABORT3
COMMON/MA30G/EPS,RMIN
COMMON/MA30I/TOL,BIG,NDROP,NSRCH,LBIG
```

The double precision version

```
COMMON/MA30ED/LP,ABORT1,ABORT2,ABORT3
COMMON/MA30GD/EPS,RMIN
COMMON/MA30ID/TOL,BIG,NDROP,NSRCH,LBIG
```

LP has the same meaning as in MA30A/AD.

ABORT1,ABORT2,ABORT3 are LOGICAL variables that are not referenced by MA30B/BD.

EPS is a REAL (DOUBLE PRECISION in the D version) variable. It is used to test for small pivots. Its default value is 1.0E-4 (1.0D-4 in the D version). If the user sets EPS to any value greater than 1.0, no check is made on the size of the pivots. Although the absence of such a check would fail to warn the user of bad instability, its absence will enable MA30B/BD to run slightly faster. An *a posteriori* check on the stability of the factorization can be obtained from MC24A/AD.

RMIN is a REAL (DOUBLE PRECISION in the D version) variable which gives the user some information about the stability of the decomposition. At each stage, *i* say, of the **LU** decomposition, the magnitude of the pivot $APIV_i$ is compared with the largest off-diagonal entry ($ROWMAX_i$) currently in its row (row *i* of **U**). If the smallest ratio $APIV_i/ROWMAX_i$ is less than EPS, RMIN is set to this value and IFLAG is returned with the value I (see section 2.3), which is the index of the row in which this occurs. If the user sets EPS greater than one, this test is not performed. In this case, and when there are no small pivots, `RMIN` will be set equal to `EPS`.

`TOL` is a `REAL` (`DOUBLE PRECISION` in the D version) variable not referenced by `MA30B/BD`.

`BIG` is a `REAL` (`DOUBLE PRECISION` in the D version) variable. If `LBIG` has been set to `.TRUE.`, `BIG` will be set to the largest entry encountered during the factorization by `MA30B/BD`. Since `BIG` is initialized at compile time only to a default value of zero, it should be reset between different runs of `MA30B/BD` (or between runs of `MA30A/AD` and `MA30B/BD`) within the same program. Note that, if `MA30` is used through the interface `MA28`, then `BIG` is initialized in `MA28` to the largest entry in the input matrix.

`NDROP` is an `INTEGER` variable not referenced by `MA30B/BD`.

`NSRCH` is an `INTEGER` variable not referenced by `MA30B/BD`.

`LBIG` is a `LOGICAL` variable. If `LBIG` is set to `.TRUE.`, the value of the largest entry encountered in the factorization by `MA30B/BD` is returned in `BIG`. Setting `LBIG` to `.TRUE.` will increase the factorization time for `MA30B/BD` by about 20 percent. The default value for `LBIG` is `.FALSE.`.

2.2.3 Common blocks used by MA30C/CD

The single precision version

```
COMMON/MA30H/RESID
```

The double precision version

```
COMMON/MA30HD/RESID
```

`RESID` is a `REAL` (`DOUBLE PRECISION` in the D version) variable. In the case of singular or rectangular matrices, its final value will be equal to the maximum residual $|b_i - \sum_j a_{ij} x_j|$ for the unsatisfied equations; otherwise its value will be set to zero.

2.3 Error diagnostics

2.3.1 Error diagnostics for MA30A/AD

If the subroutine performs the **LU** decomposition without any complications or errors, the value of `IFLAG` will be non-negative on exit from `MA30A/AD`. The complications which can arise are given below. Some are more disastrous than others and the user must decide in each instance what further action to take. In some cases messages are output on stream `LP` (unless `LP=0` see section 2.2.1). Possible negative values for `IFLAG` are as follows:

–1 The matrix is structurally singular with rank given by `IRANK` in `COMMON` block `MA30F/FD` (see section 2.2.1). The message: `ERROR RETURN FROM MA30A/AD BECAUSE MATRIX IS STRUCTURALLY SINGULAR` will be output. If, however, the user wants the **LU** decomposition of a structurally singular matrix (see section 2.6) and sets the common block variable `ABORT1` to `.FALSE.`, then, in the event of singularity and a successful decomposition, `IFLAG` is returned with the value +1 and no message is output.

–2 The matrix is numerically singular (it may also be structurally singular) with estimated rank given by `IRANK` in `COMMON` block `MA30F/FD` (see section 2.2.1). The message: `ERROR RETURN FROM MA30A/AD BECAUSE MATRIX IS NUMERICALLY SINGULAR` will be output. The user can choose to continue the decomposition even when a zero pivot is encountered by setting common block variable `ABORT2` to `.FALSE.`. If a singularity is encountered, `IFLAG` will then return with a value of +2, and no message is output if the decomposition has been completed successfully.

–3 `LIRN` has not been large enough to continue with the decomposition. Should this happen, the message: `ERROR RETURN FROM MA30A/AD BECAUSE LIRN NOT BIG ENOUGH AT STAGE ... IN BLOCK ... WITH FIRST ROW ... AND LAST ROW ...` is output. If the stage was zero, then common block variable `MINIRN` (see section 2.2.1) gives the length sufficient to start the decomposition on this block and the message: `TO CONTINUE SET LIRN TO AT LEAST ...` is output. For a successful decomposition on this block the user should make `LIRN` slightly (say about `N/2`) greater than this value, which is that of `MINIRN`.

–4 LICN has not been large enough to continue with the decomposition. Should this happen, the message: ERROR RETURN FROM MA30A/AD BECAUSE LICN NOT BIG ENOUGH AT STAGE ... IN BLOCK ... WITH FIRST ROW ... AND LAST ROW ... is output.

–5 The decomposition has been completed but some of the **LU** factors have been discarded to create enough room in A/ICN. The variable MINICN in COMMON block MA30F/FD (see section 2.2.1) then gives the size that LICN should be to enable the factorization to be successful. Each time any of the **LU** factors are destroyed, the message: LU DECOMPOSITION DESTROYED TO CREATE MORE SPACE is output. If the user sets common block variable ABORT3 to .TRUE., the subroutine will exit immediately instead of destroying any factors and continuing.

–6 Both LICN and LIRN are too small. Termination has been caused by lack of space in IRN (see error IFLAG= –3), but already some of the **LU** factors in A/ICN have been lost (see error IFLAG= –5). MINICN gives the minimum amount of space required in A/ICN for decomposition up to this point and the message: ERROR RETURN FROM MA30A/AD LIRN AND LICN TOO SMALL is output.

2.3.2 Error diagnostics for MA30B/BD

If MA30B/BD performs a successful decomposition, IFLAG will have a non-negative value on exit. Positive values indicate a warning; negative values indicate an error. Possible values for IFLAG are now described.

-I The routine has terminated at stage I for one of the following reasons.

(i) A zero pivot was found at stage I, where MA30A/AD found a nonzero pivot.

(ii) An entry was found in a part of row I which lay in a submatrix which was entirely zero after decomposition using MA30A/AD (see section 2.6.1). With this error return the message: ERROR RETURN FROM MA30B/BD SINGULARITY DETECTED IN ROW ... is output.

+I Although the decomposition has been successful, this return indicates that there is a small pivot in row I with possible consequent stability problems. The common block variable RMIN gives an idea of how severe this is. See the description of common block MA30G/GD (section 2.2.2) for further details on this return.

2.3.3 Error diagnostics for MA30C/CD

There are no error returns from this subroutine.

2.4 Parameter usage summary

2.4.1 MA30A/AD

Input by user: N, ICN, A(IDISP(1):LICN), LICN, LENR(N), IDISP(2), IP(N), LIRN, U.

Unchanged by MA30A/AD: N, LICN, LIRN.

Work arrays: IRN(LIRN), LENC(N), IFIRST(N), LASTR(N), NEXTR(N), LASTC(N), NEXTC(N), IPTR(N), IPC(N).

Output from MA30A/AD: ICN, A, LENR, LENRL(N), IDISP, IP, IQ, IFLAG.

2.4.2 MA30B/BD

Input by user: A(IDISP(1):IDISP(2)).

Input (from MA30A/AD): N, ICN(IDISP(1):IDISP(2)), LICN, LENR(N), LENRL(N), IDISP(2).

Unchanged by MA30B/BD: N, ICN, LICN, LENR, LENRL, IDISP.

Work arrays: W(N), IW(N).

Output from MA30B/BD: A, IFLAG.

2.4.3 MA30C/CD

Input by user: LENOFF(N), X(N), MTYPE, A(IDISP(1)–1), ICN(IDISP(1)–1).

Input (from MA30A/AD or MA30B/BD): N, A(IDISP(1):IDISP(2)), ICN(IDISP(1):IDISP(2)), LICN, IDISP(2), LENR(N), LENRL(N), IP(N), IQ(N).

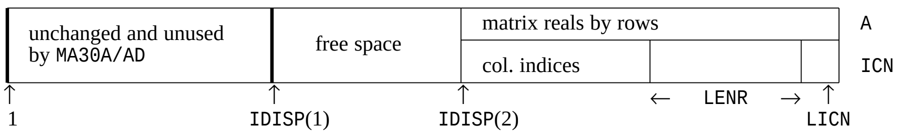
Unchanged by MA30C/CD: N, A, ICN, LICN, IDISP, LENR, LENRL, LENOFF, IP, IQ, MTYPE.

Work array: W(N).

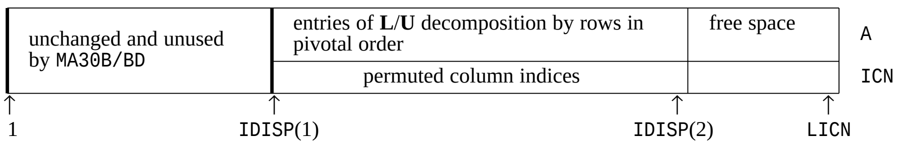
Output from MA30C/CD: X.

2.5 Data structures summary

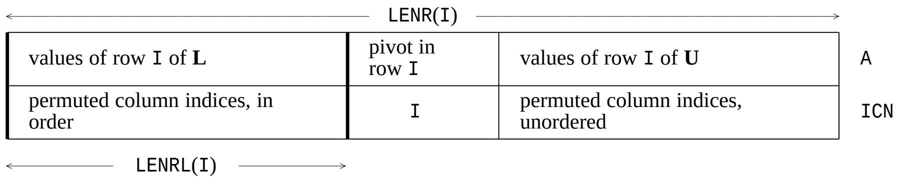
2.5.1 Input to MA30A/AD



2.5.2 Output from MA30A/AD and input to MA30B/BD (A is overwritten)

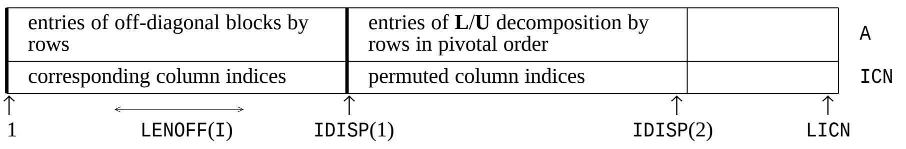

unchanged and unused by MA30B/BD
entries of L/U decomposition by rows in pivotal order
free space
permuted column indices
A
ICN
1
IDISP(1)
IDISP(2)
LICN

and for each row (row I, say)



N.B. Fill-ins can occur anywhere within the row.

2.5.3 Input to MA30C/CD


entries of off-diagonal blocks by rows
entries of L/U decomposition by rows in pivotal order
A
corresponding column indices
permuted column indices
ICN
1
LENOFF(I)
IDISP(1)
IDISP(2)
LICN

2.6 Singular or rectangular matrices

2.6.1 MA30A/AD

MA30A/AD can perform a decomposition on matrices which are singular or rectangular. This facility is controlled by the common block variables ABORT1 and ABORT2 (see section 2.2.1).

In any singular block of the matrix, MA30A/AD will permute a null or zero block to the end, viz:

$$\mathbf{P}_k \mathbf{A}_{kk} \mathbf{Q}_k = \begin{pmatrix} \mathbf{A}_1 & \mathbf{C} \\ \mathbf{D} & \mathbf{0} \end{pmatrix},$$

performing an **LU** decomposition on the first r_k rows and columns, where the $r_k \times r_k$ matrix $\mathbf{A}_1$ is nonsingular. The final rank (IRANK) equals $\sum_k r_k$ where the sum is over the diagonal blocks. For each row of **A** in which a singularity is detected the corresponding entry in IQ is made negative.

2.6.2 MA30B/BD

MA30B/BD will produce the **LU** decomposition of a singular or rectangular matrix provided that a previous matrix was successfully decomposed by MA30A/AD and the same singularities are found by both the MA30B/BD and the MA30A/AD runs. Thus the zero block of the above figure must be zero after MA30B/BD and the block corresponding to the first r_k rows and columns must be nonsingular.

2.6.3 MA30C/CD

If MA30A/AD or MA30B/BD have been used to decompose a singular or rectangular matrix, then MA30C/CD will still perform the solution process using the information in array IQ which has a negative value for any row in which a singularity occurs. Because of the mode of operation of these earlier codes, these singularities will always occur at the ends of blocks. The maximum residual for the unsatisfied equations is given by the common block variable RESID (see section 2.2.3).

3 GENERAL INFORMATION

Use of common: The common blocks MA30E/ED, MA30F/FD, MA30G/GD, MA30H/HD and MA30I/ID are used by the subroutines. Default values are set by BLOCK DATA MA30J/JD.

Workspace:

MA30A/AD uses the INTEGER work arrays IRN of length LIRN (> NZ), LENC, IFIRST, LASTR, NEXTR, LASTC, NEXTC all of length N, and the INTEGER work arrays IPTR and IPC both of length N. Note that arrays LASTC and NEXTC are not referenced if NSRCH (see section 2.2) has been set to a value less than or equal to N.

MA30B/BD uses an INTEGER array IW of length N and a REAL (DOUBLE PRECISION in the D version) array W also of length N.

MA30C/CD uses a REAL (DOUBLE PRECISION in the D version) array W of length N.

Other routines called directly: MA30D/DD, MA30J/JD (block data).

MA30D/DD need never be called directly by the user.

There are no subroutines called by MA30B/BD or MA30C/CD.

Input/output: Error messages are output on Fortran stream number LP. The default value for LP is 6 and these messages can be suppressed by setting LP to zero (see section 2.2).

4 METHOD

MA30A/AD uses a sparse variant of Gaussian elimination to decompose each diagonal block into its **LU** factors.

MA30B/BD utilizes knowledge of the pivotal sequence and the sparsity structure of the **LU** factors from a previous call to MA30A/AD to factorize the new matrix into its **LU** factors by row Gauss elimination.

MA30C/CD performs simple forward elimination and back-substitution on each block in turn performing back-substitution on the off-diagonal parts when required. The code for the solution of the direct problem and its transpose are entirely separate.

A discussion of the design of these subroutines is given by Duff and Reid (ACM Trans Math Software **5** pp. 18-35, 1979) while fuller details of the implementation are given in Duff (Harwell Report R 8730, 1977). The additional pivoting option in MA30A/AD and the use of drop tolerances (see common block MA30I/ID in section 2.2) were added to the package after joint work by Duff, Reid, Schaumburg, Wasniewski and Zlatev.

5 EXAMPLE OF USE

In the example code shown below, we read in the entries of a sparse matrix, which is then sorted into row order. We should point out that MA30A/AD expects that the matrix entries are in row order, though the columns within each row need not be in any particular order. We then decompose the linear equations and finally solve for the unknowns.

```
      PARAMETER( MAXN = 50 , MAXNZ = MAXN*MAXN ,
     +           LICN = 500 , LIRN = 500 )
      DOUBLE PRECISION A(LICN), W(MAXN), X(MAXN), U
      INTEGER ICN(LICN), IRN(LIRN), IPTR(MAXN), IPC(MAXN), LENC(MAXN),
     +        LENR(MAXN), LENRL(MAXN), IFIRST(MAXN), LASTR(MAXN),
     +        NEXTR(MAXN), LASTC(MAXN), NEXTC(MAXN), IP(MAXN),
     +        IQ(MAXN), IDISP(2), LENOFF(MAXN), IFLAG,
     +        IROW(MAXNZ), ICOL(MAXNZ), JPTR(MAXN), N, NZ, I
      READ(5,*) N , NZ
      IF ( (N .GT. MAXN) .OR. (NZ .GT. MAXNZ) ) THEN
         WRITE(6,550)
         STOP
      END IF
C     READ IN DATA IN ANY ORDER INTO THE TEMPORY ARRAY'S IROW AND ICOL
      READ(5, * ) ( IROW(I) , ICOL(I) , A(I) , I=1,NZ )
      JDISP = 0
C     SORT THE INPUT MATRIX INTO ROW ORDER (THIS IS HOW
C                    MA30A/AD MUST HAVE IT'S INPUT DATA)
      CALL MC20AD( N, NZ, A, ICOL, JPTR, IROW, JDISP)
C     JPTR(I) POINTS TO THE START OF THE I'TH ROW OF THE MATRIX STORED
C     IN THE ARRAY A().
C     NOW SETUP THE INPUT DATA ARRAYS FOR MA30A/AD
      DO 100 I=1,NZ
        ICN(I) = ICOL(I)
 100  CONTINUE
      ISTART   = LICN - NZ  + 1
      IDISP(2) = ISTART
      IDISP(1) = NZ + 1
C     LENR(I)  HOLDS THE NUMBER OF ENTRIES IN THE I'TH ROW
      DO 150 I=1,N-1
        LENR(I) = JPTR(I+1) - JPTR(I)
 150  CONTINUE
      LENR(N) = NZ - JPTR(N) + 1
C     MA30AD EXPECTS THE BLOCK THAT IT IS TO REDUCE TO LU FORM
C     TO BE CONTAINED IN THE ENDS OF THE ARRAYS A() AND ICN()
C     SO MOVE THE MATRIX WE WISH TO HAVE REDUCED TO THE END
      DO 155 I = 1 , NZ
        A(ISTART+I-1) = A(I)
        ICN(ISTART+I-1) = ICN(I)
 155  CONTINUE
C     WE WISH TO USE MA30A/AD SUCH THAT WE TREAT THE INPUT MATRIX
C     A() AS A SINGLE BLOCK, SET THE PERMUTATION ARRAYS IP() AND IQ()
C     TO REFLECT THIS FACT.
      DO 200 I=1,N
        IP(I) = I
        IQ(I) = I
 200  CONTINUE
```

```
      U = 0.1D0
      CALL MA30AD( N, ICN, A, LICN, LENR, LENRL, IDISP, IP, IQ, IRN,
     +             LIRN, LENC, IFIRST, LASTR, NEXTR, LASTC,
     +             NEXTC, IPTR, IPC, U, IFLAG )
      IF (IFLAG .LT. 0) THEN
         WRITE(6,650) IFLAG
         STOP
      END IF
C     READ IN THE RIGHT-HAND SIDE VECTOR
      READ(5, * ) ( X(I) , I=1,N )
      MTYPE = 1
      LENOFF(1) = -1
      CALL MA30CD( N, ICN, A, LICN, LENR, LENRL,
     +             LENOFF, IDISP, IP, IQ, X, W, MTYPE )
      WRITE(6,400)
      WRITE(6,450) ( I, X(I) ,I=1,N)
 400  FORMAT( / ' THE SOLUTION VECTOR IS:'/)
 450  FORMAT( '  X(',I2,')= ', D13.5 )
 550  FORMAT( / ' ERROR IN THE INPUT DATA FOR N AND/OR NZ'//)
 650  FORMAT( ' ERROR RETURN FROM MA30AD WITH IFLAG =',I2 )
      STOP
      END
```

If we wish to solve the following problem:

$$\begin{pmatrix} 1. & & & 4. \\ & 6. & 7. & 8. \\ 9. & & 11. & 12. \\ & 14. & & 16. \end{pmatrix} \mathbf{x} = \begin{pmatrix} 5. \\ 21. \\ 32. \\ 30. \end{pmatrix}$$

we might have as input

```
        4      10
        4       4           16.
        3       3           11.
        2       2            6.
        1       1            1.
        4       2           14.
        2       3            7.
        3       1            9.
        2       4            8.
        3       4           12.
        1       4            4.
     5.
    21.
    32.
    30.
```

and we would get the following output

```
  THE SOLUTION VECTOR IS:

   X( 1)=   0.10000D+01
   X( 2)=   0.10000D+01
   X( 3)=   0.10000D+01
   X( 4)=   0.10000D+01
```