1 SUMMARY

This set of subroutines compute the **solution to an extended system of** $n+m$ **real linear equations in** $n+m$ **unknowns,**

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}, \tag{1}$$

in the case where the n by n matrix $\mathbf{A}$ is nonsingular and solutions to the systems

$$\mathbf{Ax} = \mathbf{b} \text{ and } \mathbf{A}^T\mathbf{y} = \mathbf{c}$$

may be obtained from an external source, such as an existing factorization. The subroutine uses reverse communication to obtain the solution to such smaller systems. The method makes use of the Schur complement matrix

$$\mathbf{S} = \mathbf{D} - \mathbf{C}\,\mathbf{A}^{-1}\mathbf{B}.$$

The Schur complement is stored and factorized as a dense matrix and the subroutine is thus only appropriate if there is sufficient storage for this matrix. Special advantage is taken of symmetry and definiteness in the coefficient matrices. Provision is made for introducing additional rows and columns to, and removing existing rows and columns from, the extended matrix.

ATTRIBUTES — Version: 1.0.0. (12 July 2004) **Types:** Real (single, double). **Original date:** March, 2001. **Origin:** N.I.M. Gould, Rutherford Appleton Laboratory. **Remark:** This is a thread-safe version of `MA39` and supersedes it.

2 HOW TO USE THE PACKAGE

2.1 Calling sequence

To solve the extended system, the user must first make a call to `MA69A/AD` to form and factorize the Schur complement matrix $\mathbf{S}$ and then call `MA69B/BD` to compute the solution to the extended system. The solution of additional extended systems, with the same coefficient matrix but different right-hand sides, may be found by further calls to `MA69B/BD`.

The solution of further-extended systems of the form

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{c}_1 \\ \mathbf{C} & \mathbf{D} & \mathbf{c}_2 \\ \mathbf{r}_1^T & \mathbf{r}_2^T & d \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ x_{n+m+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ b_{n+m+1} \end{pmatrix}$$

may be found by firstly calling `MA69C/CD` to update the existing factorization of $\mathbf{S}$ (obtained from `MA69A/AD`) to give that of the Schur complement of $\mathbf{A}$ in the further-extended coefficient matrix and then by calling `MA69B/BD.` Likewise, the solution of extended systems of the form

$$\begin{pmatrix} \mathbf{A} & \bar{\mathbf{B}} \\ \bar{\mathbf{C}} & \bar{\mathbf{D}} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix},$$

in which a row and column are removed from the coefficient matrix of (1), may be found by firstly calling `MA69D/DD` to update the existing factorization of $\mathbf{S}$ (obtained from `MA69A/AD)` to give that of the Schur complement of $\mathbf{A}$ in the new extended coefficient matrix and then once again by calling `MA69B/BD.`

2.2 The factorization stage

Call `MA69A/AD` to factorize the Schur complement matrix.

The single precision version

```
    CALL MA69A ( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
   *             IPCD, R, LR, Q, LQ, W, VECTOR, ISAVE, INFORM )
```

The double precision version

```
    CALL MA69AD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
   *             IPCD, R, LR, Q, LQ, W, VECTOR, ISAVE, INFORM )
```

`N` is an `INTEGER` variable set by the user to the dimension of the matrix **A**. It is unchanged by the subroutine. **Restriction:** `N`≥ 0.

`M` is an `INTEGER` variable set by the user to the dimension of the matrix **D**. It is unchanged by the subroutine. **Restriction:** `M`≥ 0.

`ICLASS` is an `INTEGER` variable set by the user to indicate the type of matrix used. The permitted values are:

1 the extended matrix is unsymmetric,

2 the extended matrix is symmetric,

3 the extended matrix is symmetric and the Schur complement matrix **S** is known to be positive definite.

4 the extended matrix is symmetric and the Schur complement matrix **S** is known to be negative definite.

It is unchanged by the subroutine. **Restriction:** $1 \leq$ `ICLASS` ≤ 4.

`LBD` is an `INTEGER` variable which must be set by the user to be at least as large as the number of nonzero entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{B} \\ \mathbf{D}_U \end{pmatrix},$$

where $\mathbf{D}_U$ is the upper triangular part of **D**, ie, $(\mathbf{D}_U)_{ij} = (\mathbf{D})_{ij}$ if $i \leq j$ and zero otherwise. It is unchanged by the subroutine.

`BD` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LBD` which must be set by the user to the value of the entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{B} \\ \mathbf{D}_U \end{pmatrix},$$

where $\mathbf{D}_U$ is the upper triangular part of **D**. The entries must be ordered by columns, with the entries in each column contiguous and those of column j preceding those of column j+1 ($j = 1, ..., $`M`). The ordering within each column is unimportant. Any element in `BD` below the diagonal of **D** will be removed by `MA69A/AD.`

`IRNBD` is an `INTEGER` array of length `LBD,` which must be set by the user. It must contain the row indices of the corresponding entries in `BD.` Any element whose index in `IRNBD` is below the diagonal of **D** will be removed by `MA69A/AD.`

`IPBD` is an `INTEGER` array of length `M+1,` which must be set by the user. It must be set so that `IPBD(j)` points to the positions in the arrays `BD` and `IRNBD` of the first entry in column j ($j = 1, ...,$ `M`). `IPBD(M+1)` must be set to the number of entries in

$$\begin{pmatrix} \mathbf{B} \\ \mathbf{D}_U \end{pmatrix}$$

plus one. This argument may be changed by `MA69A/AD` to reflect the removal of any subdiagonal entries of **D** input in `BD` and `IRNBD.`

`LCD` is an `INTEGER` variable which must be set by the user when `ICLASS=1` to be at least as large as the number of nonzero entries in the partitioned matrix ($\mathbf{C}\ \mathbf{D}_L$), where $\mathbf{D}_L$ is the strict lower triangular part of $\mathbf{D}$, i.e., $(\mathbf{D}_L)_{ij} = (\mathbf{D})_{ij}$ if $i > j$ and zero otherwise. It need not be set if `ICLASS` > 1 and is unchanged by the subroutine.

`CD` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LCD` which must be set by the user when `ICLASS=1` to the value of the entries in the partitioned matrix ($\mathbf{C}\ \mathbf{D}_L$), where $\mathbf{D}_L$ is the strict lower triangular part of $\mathbf{D}$. The entries must be ordered by rows, with the entries in each row contiguous and those of row i preceding those of row i+1 ($i = 1, ...,$ `M`). Any element on or to the right of the diagonal of $\mathbf{D}$ will be removed by `MA69A/AD`. The ordering within each row is unimportant. This argument need not be set if `ICLASS` > 1.

`JCNCD` is an `INTEGER` array of length `LCD`, which must be set by the user when `ICLASS=1`. It must then contain the column indices of the corresponding entries in `CD`. Any element whose index is on or to the right of the diagonal of $\mathbf{D}$ will be removed by `MA69A/AD`. This argument need not be set if `ICLASS` > 1.

`IPCD` is an `INTEGER` array of length `M+1`, which must be set by the user when `ICLASS=1`. It must then be set so that `IPCD(i)` points to the positions in the arrays `CD` and `JCNCD` of the first entry in row i ($i = 1, ...,$ `M`). `IPCD(M+1)` must be set to the number of entries in ($\mathbf{C}\ \mathbf{D}_L$) plus one. This argument may be changed by `MA69A/AD` to reflect the removal of entries on or to the right of the diagonal of $\mathbf{D}$ input in `CD` and `JCNCD`. It need not be set if `ICLASS` > 1.

`R` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LR` which need not be set by the user. On a successful exit from `MA69A/AD`, `R` contains the upper triangular matrix associated with a factorization of the Schur complement matrix $\mathbf{S}$. The upper triangle of the relevant matrix is stored by columns. If `ICLASS=1` or `2`, the matrix returned is the matrix $\mathbf{R}$ associated with the $\mathbf{QR}$ factorization $\mathbf{S} = \mathbf{QR}$ of the Schur complement. If `ICLASS=3`, `R` contains the matrix $\mathbf{R}$ associated with the Cholesky factorization $\mathbf{S} = \mathbf{R}^T\mathbf{R}$ of the Schur complement. If `ICLASS=4`, `R` contains the matrix $\mathbf{R}$ associated with the Cholesky factorization $-\mathbf{S} = \mathbf{R}^T\mathbf{R}$ of the negative of the Schur complement.

`LR` is an `INTEGER` variable set by the user to the length of the array `R`. It is unchanged by the subroutine. **Restriction:** `LR` $\geq$ `M*(M+1)/2`.

`Q` is a `REAL` (`DOUBLE PRECISION` in the D version) two dimensional `LQ` by `M` array which need not be set by the user. On a successful exit from `MA69A/AD` when `ICLASS=1` or 2 `Q(i,j)` ($1 \leq i,j \leq$ `M`) contains the i,j–th entry of the matrix $\mathbf{Q}$ in the $\mathbf{QR}$ factorization $\mathbf{S} = \mathbf{QR}$ of the Schur complement. `Q` is not used for other values of `ICLASS`.

`LQ` is an `INTEGER` variable which must be set by the user to actual size of the first dimension of the two dimensional array `Q`. It is unchanged by the subroutine. **Restriction:** `LQ` $\geq$ `M`.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least `M` which is used for workspace and which need not be set by the user.

`VECTOR` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least `N` which need not be set by the user on initial (`INFORM=1`) entry. If `INFORM` is greater than 1 on exit, a re-entry must be made with `VECTOR` set appropriately (see `INFORM`).

`ISAVE` is an `INTEGER` workspace variable that need not be set by the user on initial (`INFORM=1`) entry to `MA69A/AD`, and which must not be changed between this and subsequent calls.

`INFORM` is an `INTEGER` variable which must be set by the user on input to 1. On output, the value of `INFORM` is used to request additional information, to signal an error in the input data or to indicate a successful call to the subroutine. Possible values of `INFORM` and their consequences are as follows:

0 The required factorization of the Schur complement S is given in the arrays `R` and `Q`.

2,3 The user must obtain the solution to the system of equations $\mathbf{Ax} = \mathbf{b}$ and re-enter `MA69A/AD` with `INFORM` unchanged. The particular vector $\mathbf{b}$ is returned in the array `VECTOR`; the user must calculate $\mathbf{x}$ and pass this vector back in `VECTOR`.

4 The user must obtain the solution to the system of equations $\mathbf{A}^T\mathbf{y} = \mathbf{c}$ and re-enter `MA69A/AD` with

INFORM unchanged. The particular vector **c** is returned in the array VECTOR; the user must calculate **y** and pass this vector back in VECTOR.

Negative values of INFORM indicate that some input data has been incorrectly assigned. Possible values of INFORM and some potential remedies are:

–1 One of the previously mentioned restrictions on the values of M, N, ICLASS, LQ or LR has been violated. Check these values and re-enter with INFORM=1.

–2 The Schur complement matrix is singular; this has been detected during the **QR** factorization of **S**. Check the matrix entries.

–3 The Schur complement matrix is not positive definite; this has been detected during the Cholesky factorization of **S**. Check the matrix entries.

–4 The Schur complement matrix is not negative definite; this has been detected during the Cholesky factorization of $-\mathbf{S}$. Check the matrix entries.

–5 MA69A/AD has been called with INFORM ≤ 0. Reset INFORM to 1 and re-enter.

2.3 The solution stage

Call MA69B/BD to solve the extended system of equations using the factorization produced by a previous call to MA69A/AD. MA69B/BD should only be used if the call to MA69A/AD terminated successfully (i.e., with INFORM=0).

The single precision version

```
    CALL MA69B ( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
   *             IPCD, R, LR, Q, LQ, RHS, X, VECTOR, INFORM )
```

The double precision version

```
    CALL MA69BD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
   *             IPCD, R, LR, Q, LQ, RHS, X, VECTOR, INFORM )
```

Arguments N, M, ICLASS, LCD, CD, JCNCD, IPCD, LBD, BD, IRNBD, IPBD R, LR, Q, LQ are the same as those of the same name in the call to MA69A/AD and must be unchanged since that call. They are not altered by MA69B/BD.

RHS is a REAL (DOUBLE PRECISION in the D version) array of length N+M which must be set on entry to the values of the right-hand-side vector

$$\begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

of the extended system of equations. RHS is not altered by the subroutine.

X is a REAL (DOUBLE PRECISION in the D version) array of length N+M which need not be set on entry. On final exit, X contains the values of the solution

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}$$

to the extended system of equations.

VECTOR is a REAL (DOUBLE PRECISION in the D version) array of length at least N which need not be set by the user on initial (INFORM=1) entry. If INFORM is greater than 1 on exit, a re-entry must be made with VECTOR set appropriately (see INFORM).

INFORM is an INTEGER variable which must be set by the user on input to 1. On output, the value of INFORM is used to request additional information or to indicate a successful call to the subroutine. Possible values of INFORM and their consequences are as follows:

0 The required solution of the extended system of equations is given in the array X.

2,3 The user must obtain the solution to the system of equations $\mathbf{Ax}=\mathbf{b}$ and re-enter `MA69A/AD` with `INFORM` unchanged. The particular vector **b** is returned in the array `VECTOR;` the user must calculate **x** and pass this vector back in `VECTOR`.

4 The user must obtain the solution to the system of equations $\mathbf{A}^T\mathbf{y}=\mathbf{c}$ and re-enter `MA69A/AD` with `INFORM` unchanged. The particular vector **c** is returned in the array `VECTOR;` the user must calculate **y** and pass this vector back in `VECTOR`.

2.4 The updating stage

Call `MA69C/CD` to extend the factorization of the Schur complement when a new row and column are appended to the extended matrix. Subsequent systems of equations with the larger coefficient matrix may then be solved by calls to `MA69B/BD`. Note in particular that the arrays `CD`, `JCNCD`, `IPCD`, `BD`, `IRNBD` and `IPBD` must be expanded to allow for the incoming row and column.

The single precision version

```
      CALL MA69C ( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, W, VECTOR, RSAVE, INFORM )
```

The double precision version

```
      CALL MA69CD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, W, VECTOR, RSAVE, INFORM )
```

Arguments `N` `M`, and `ICLASS` are the same as those of the same name in the call to `MA69A/AD`. They are not altered by `MA69C/CD`. `ICLASS` must not have been changed since previous calls to `MA69A/AD`.

`LBD` is an `INTEGER` variable which must be set by the user to be at least as large as the number of nonzero entries in the matrix

$$\begin{pmatrix} \mathbf{B} & \mathbf{c}_1 \\ \mathbf{D}_U & \mathbf{c}_2 \\ & d \end{pmatrix},$$

where $\mathbf{D}_U$ is the upper triangular part of **D**. This argument is unchanged by the subroutine.

`BD` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LBD` which must be set by the user to the value of the entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{B} & \mathbf{c}_1 \\ \mathbf{D}_U & \mathbf{c}_2 \\ & d \end{pmatrix},$$

where $\mathbf{D}_U$ is the upper triangular part of **D**. The entries must be ordered by columns, with the entries in each column contiguous and those of column j preceding those of column j+1 ($j=1,....,$`M+1`). Any element below the diagonal of **D** will be ignored. The ordering within each column is unimportant. Any element below the diagonal of **D** will be removed by `MA69C/CD`.

`IRNBD` is an `INTEGER` array of length `LBD,` which must be set by the user. It must contain the row indices of the corresponding entries in `BD`. Any element whose index is below the diagonal of **D** will be removed by `MA69C/CD`.

`IPBD` is an `INTEGER` array of length `M+2,` which must be set by the user. It must be set so that `IPBD(j)` points to the positions in the arrays `BD` and `IRNBD` of the first entry in column j ($j=1,....,$`M+1`). `IPBD(M+2)` must be set to the number of entries in

$$\begin{pmatrix} \mathbf{B} & \mathbf{c}_1 \\ \mathbf{D}_U & \mathbf{c}_2 \\ & d \end{pmatrix}$$

plus one. This argument may be changed by `MA69C/CD` to reflect the removal of any subdiagonal entries of **D** input in `BD` and `IRNBD`.

`LCD` is an `INTEGER` variable set by the user when `ICLASS=1` to be at least as large as the number of nonzero entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}_L \\ \mathbf{r}_1 & \mathbf{r}_2 \end{pmatrix},$$

where $\mathbf{D}_L$ is the strict lower triangular part of **D**. It need not be set if `ICLASS` >1 and is unchanged by the subroutine.

`CD` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LCD` which must be set by the user to the value of the entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}_L \\ \mathbf{r}_1 & \mathbf{r}_2 \end{pmatrix},$$

where $\mathbf{D}_L$ is the strict lower triangular part of **D**. The entries must be ordered by rows, with the entries in each row contiguous and those of row i preceding those of row i+1 ($i=1, ...,$ `M+1`). Any element on or to the right of the diagonal of **D** will be removed by `MA69C/CD`. The ordering within each row is unimportant. This argument need not be set if `ICLASS` >1.

`JCNCD` is an `INTEGER` array of length `LCD`, which must be set by the user when `ICLASS=1`. It must then contain the column indices of the corresponding entries in `CD`. Any element whose index is on or to the right of the diagonal of **D** will be removed by `MA69C/CD`. This argument need not be set if `ICLASS` >1.

`IPCD` is an `INTEGER` array of length `M+2`, which must be set by the user when `ICLASS=1`. It must then be set so that `IPCD(i)` points to the positions in the arrays `CD` and `JCNCD` of the first entry in row i ($i=1, ...,$ `M+1`). `IPCD(M+2)` must be set to the number of entries in

$$\begin{pmatrix} \mathbf{C} & \mathbf{D}_L \\ \mathbf{r}_1 & \mathbf{r}_2 \end{pmatrix}$$

plus one. This argument may be changed by `MA69C/CD` to reflect the removal of entries on or to the right of the diagonal of **D** input in `CD` and `JCNCD`. It need not be set if `ICLASS` >1.

`R` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LR` which serves the same purpose as that of the same name in `MA69A/AD`. On entry to `MA69C/CD`, the array must be unchanged from that output from the most recent call to `MA69A/AD`, `MA69C/CD` or `MA69D/DD` excepting that, if `M=0`, `R` need not have been set. On exit, `R` contains the relevant factor of the Schur complement of **A** in the further-extended matrix.

`LR` is an `INTEGER` variable set by the user to the length of the array `R`. It is unchanged by the subroutine. **Restriction:** `LR` $\geq$ `(M+1)*(M+2)/2`.

`Q` is a `REAL` (`DOUBLE PRECISION` in the D version) two dimensional `LQ` by `M+1` array which serves the same purpose as that of the same name in `MA69A/AD`. On entry to `MA69C/CD`, the array must be unchanged from that output from the most recent call to `MA69A/AD`, `MA69C/CD` or `MA69D/DD` excepting that, if `M=0`, `Q` need not have been set. On exit, `Q` contains the relevant factor of the Schur complement of **A** in the further-extended matrix. `Q` is only used for `ICLASS=1,2`.

`LQ` is an `INTEGER` variable which must be set by the user to actual size of the first dimension of the two dimensional array `Q`. It is unchanged by the subroutine. **Restriction:** `LQ` $\geq$ `M+1`.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least `M+1` which is used for workspace and which need not be set by the user. The contents of `W` must not be changed between the initial and subsequent calls to `MA69C/CD` or `MA69D/DD`.

`VECTOR` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least `N` which need not be set by the user on initial (`INFORM=1`) entry. If `INFORM` is greater than 1 on exit, a re-entry must be made with `VECTOR` set appropriately (see `INFORM`).

RSAVE is an REAL (DOUBLE PRECISION in the D version) workspace variable that need not be set by the user on initial (INFORM=1) entry to MA69C/CD, and which must not be changed between this and subsequent calls.

INFORM is an INTEGER variable which must be set by the user on input to 1. On output, the value of INFORM is used to request additional information, to signal an error in the input data or to indicate a successful call to the subroutine. Possible values of INFORM and their consequences are as follows:

0 The required factorization of the Schur complement **S** is given in the arrays R and Q.

2,3 The user must obtain the solution to the system of equations $\mathbf{Ax}=\mathbf{b}$ and re-enter MA69C/CD with INFORM unchanged. The particular vector **b** is returned in the array VECTOR; the user must calculate **x** and pass this vector back in VECTOR.

4 The user must obtain the solution to the system of equations $\mathbf{A}^T\mathbf{y}=\mathbf{c}$ and re-enter MA69C/CD with INFORM unchanged. The particular vector **c** is returned in the array VECTOR; the user must calculate **y** and pass this vector back in VECTOR.

Negative values of INFORM indicate that some input data has been incorrectly assigned. Possible values of INFORM and some potential remedies are:

–1 One of the previously mentioned restrictions on the values of M, N, ICLASS, LQ or LR has been violated. Check these values and re-enter with INFORM=1.

–2 The Schur complement matrix is singular; this has been detected during the QR factorization of the further-extended Schur complement. Check the matrix entries.

–3 The Schur complement matrix is not positive definite; this has been detected during the Cholesky factorization of the further-extended Schur complement. Check the matrix entries.

–4 The Schur complement matrix is not negative definite; this has been detected during the Cholesky factorization of the negative of the further-extended Schur complement. Check the matrix entries.

–5 MA69C/CD has been called with INFORM ≤ 0. Reset INFORM to 1 and re-enter.

2.5 The deletion stage

Call MA69D/DD to extend the factorization of the Schur complement when a row and column of the existing extended matrix are to be removed. Subsequent systems of equations with the smaller coefficient matrix may then be solved by calls to MA69B/BD. The data structures for holding **B**, **C** and **D** may be automatically updated if required.

The single precision version

```
    CALL MA69D ( N, M, ICLASS, NEWDS, LBD, BD, IRNBD, IPBD, JCDROP,
   *             LCD, CD, JCNCD, IPCD, IRDROP, R, LR, Q, LQ,
   *             W, VECTOR, INFORM )
```

The double precision version

```
    CALL MA69DD( N, M, ICLASS, NEWDS, LBD, BD, IRNBD, IPBD, JCDROP,
   *             LCD, CD, JCNCD, IPCD, IRDROP, R, LR, Q, LQ,
   *             W, VECTOR, INFORM )
```

Arguments N M, and ICLASS are the same as those of the same name in the call to MA69A/AD. They are not altered by MA69C/CD. ICLASS must not have been changed since previous calls to MA69A/AD.

NEWDS is a LOGICAL variable which must be set on entry. If the user wishes the subroutine to update the data structures required to store the extended matrix (i.e., the arrays BD, IRNBD, IPBD and, if ICLASS = 1, CD, JCNCD and IPCD) to account for the impending row and column removal NEWDS must be set to .TRUE. If the call to MA69D/DD is to be followed by further calls to any of the MA69 subroutines, NEWDS should be set .TRUE. If the user is unconcerned about the data structures, NEWDS should be set .FALSE. We recommend that NEWDS is always set .TRUE. NEWDS is unchanged by the subroutine.

`LBD` is an `INTEGER` variable which must be set by the user to be at least as large as the number of nonzero entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{B} \\ \mathbf{D}_U \end{pmatrix}$$

plus `M`, where $\mathbf{D}_U$ is the upper triangular part of **D**, ie, $(\mathbf{D}_U)_{ij} = (\mathbf{D})_{ij}$ if $i \le j$ and zero otherwise. It is unchanged by the subroutine.

`BD` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LBD` which must be set by the user to the value of the entries in the partitioned matrix

$$\begin{pmatrix} \mathbf{B} \\ \mathbf{D}_U \end{pmatrix},$$

as described in §2.2. The required entries of `BD` will have been set in a previous call to `MA69A/AD`, `MA69C/CD` or `MA69DD` and must not have been altered since. If `NEWDS` is `.TRUE.`, `BD` will be altered by `MA69D/DD` to contain the entries of the partitioned matrix after the required row and column have been removed. If `NEWDS` is `.FALSE.`, `BD` will not be altered by the subroutine.

`IRNBD` is an `INTEGER` array of length `LBD`, which must be set by the user. It must contain the row indices of the corresponding entries in `BD` as described in §2.2 The required values of `IRNBD` will have been set in a previous call to `MA69A/AD`, `MA69C/CD` or `MA69DD` and must not have been altered since. If `NEWDS` is `.TRUE.`, `IRNBD` will be altered by `MA69D/DD` to contain the row indices of the entries of the partitioned matrix after the required row and column have been removed. If `NEWDS` is `.FALSE.`, `IRNBD` will not be altered by the subroutine.

`IPBD` is an `INTEGER` array of length `M+1`, which must be set by the user on input as described in §2.2. The required values of `IPBD` will have been set in a previous call to `MA69A/AD`, `MA69C/CD` or `MA69DD` and must not have been altered since. If `NEWDS` is `.TRUE.`, `IPBD` will be altered by `MA69D/DD` to contain the pointers to the start of each column of the partitioned matrix after the required row and column have been removed. If `NEWDS` is `.FALSE.`, `IPBD` will not be altered by the subroutine.

`JCDROP` is an `INTEGER` variable which must be set by the user to the index of the column of

$$\begin{pmatrix} \mathbf{B} \\ \mathbf{D} \end{pmatrix}$$

which is to be removed. It is unchanged by the subroutine. **Restriction:** $1 \le$ `JCDROP` $\le$ `M`.

`LCD` is an `INTEGER` variable which must be set by the user when `ICLASS=1` to be at least as large as the number of nonzero entries in the partitioned matrix $(\mathbf{C}\ \mathbf{D}_L)$ *plus* `M`, where $\mathbf{D}_L$ is the strict lower triangular part of **D**, i.e., $(\mathbf{D}_L)_{ij} = (\mathbf{D})_{ij}$ if $i > j$ and zero otherwise. It need not be set if `ICLASS` > 1 and is unchanged by the subroutine.

`CD` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LCD` which must be set by the user when `ICLASS=1` to the value of the entries in the partitioned matrix $(\mathbf{C}\ \mathbf{D}_L)$, as described in §2.2. The required entries of `CD` will then have been set in a previous call to `MA69A/AD`, `MA69C/CD` or `MA69DD` and must not have been altered since. If `NEWDS` is `.TRUE.` and `ICLASS=1`, `BD` will be altered by `MA69D/DD` to contain the entries of the partitioned matrix after the required row and column have been removed. If `NEWDS` is `.FALSE.`, `CD` will not be altered by the subroutine. This argument need not be set if `ICLASS` > 1 and is unchanged by the subroutine.

`JCNCD` is an `INTEGER` array of length `LCD`, which must be set by the user when `ICLASS=1`. It must then contain the column indices of the corresponding entries in `CD` as described in §2.2 The required values of `JCNCD` will have been set in a previous call to `MA69A/AD`, `MA69C/CD` or `MA69DD` and must not have been altered since. If `NEWDS` is `.TRUE.` and `ICLASS=1`, `JCNCD` will be altered by `MA69D/DD` to contain the column indices of the entries of the partitioned matrix after the required row and column have been removed. If `NEWDS` is `.FALSE.`, `JCNCD` will not be altered by the subroutine. This argument need not be set if `ICLASS` > 1 and is unchanged by the subroutine.

`IPCD` is an `INTEGER` array of length `M+1`, which must be set by the user on input when `ICLASS=1` as described in §2.2. The required values of `IPCD` will then have been set in a previous call to `MA69A/AD`, `MA69C/CD` or

MA69DD and must not have been altered since. If NEWDS is .TRUE. and ICLASS=1, IPCD will be altered by MA69D/DD to contain the pointers to the start of each row of the partitioned matrix after the required row and column have been removed. If NEWDS is .FALSE., IPCD will not be altered by the subroutine. This argument need not be set if ICLASS>1 and is unchanged by the subroutine.

IRDROP is an INTEGER variable which must be set by the user if ICLASS=1 to the index of the row of (**C** **D**) which is to be removed. It is unchanged by the subroutine. This argument need not be set if ICLASS >1. **Restriction:** $1\le$ IRDROP $\le$ M.

R is a REAL (DOUBLE PRECISION in the D version) array of length LR which serves the same purpose as that of the same name in MA69A/AD. On entry to MA69D/DD, the array must be unchanged from that output from the most recent call to MA69A/AD, MA69C/CD or MA69D/DD. On exit, R contains the relevant factor of the Schur complement of **A** in the new extended matrix.

LR is an INTEGER variable set by the user to the length of the array R. It is unchanged by the subroutine. **Restriction:** LR $\ge$ M*(M+1)/2.

Q is a REAL (DOUBLE PRECISION in the D version) two dimensional M by M array which serves the same purpose as that of the same name in MA69A/AD. On entry to MA69D/DD, the array must be unchanged from that output from the most recent call to MA69A/AD, MA69C/CD or MA69D/DD. On exit, Q contains the relevant factor of the Schur complement of **A** in the new extended matrix. Q is only used for ICLASS=1,2.

LQ is an INTEGER variable which must be set by the user to actual size of the first dimension of the two dimensional array Q. It is unchanged by the subroutine. **Restriction:** LQ $\ge$ M.

W is a REAL (DOUBLE PRECISION in the D version) array of length at least M which is used for workspace and which need not be set by the user. The contents of W must not be changed between the initial and subsequent calls to MA69D/DD or MA69C/CD.

VECTOR is a REAL (DOUBLE PRECISION in the D version) array of length at least N which is again used for workspace and which need not be set by the user.

INFORM is an INTEGER variable which must be set by the user on input to 1. On output, the value of INFORM is used to signal an error in the input data or to indicate a successful call to the subroutine. Possible values of INFORM and their consequences are as follows:

0 The required factorization of the Schur complement **S** is given in the arrays R and Q.

Negative values of INFORM indicate that some input data has been incorrectly assigned. Possible values of INFORM and some potential remedies are:

–1 One of the previously mentioned restrictions on the values of M, N, ICLASS, JCDROP, IRDROP, LQ or LR has been violated. Check these values and re-enter with INFORM=1.

–2 The Schur complement matrix is singular; this has been detected during the QR factorization of the further-extended Schur complement. Check the matrix entries.

–5 MA69D/DD has been called with INFORM ≤ 0. Reset INFORM to 1 and re-enter.

2.6 Error returns

If some error in the input data is detected by the subroutines, they return immediately to the calling program with the error flag INFORM having a negative value. See §2.2-2.5 for details.

2.7 Multiple updates

Once MA69C/CD or MA69D/DD (with NEWDS=.TRUE.) has been used to update the factorization of the Schur complement matrix, it is as if the enlarged or reduced matrix were that originally factorized by MA69A/AD. Consequently *sequences* of row and column additions and removals may be performed so long as sufficient storage is available.

3 GENERAL INFORMATION

Use of common: none.

Workspace: provided by the user through the argument `W` (`REAL` (`DOUBLE PRECISION` in the D version) array of length `M`).

Other routines called directly: internal subroutines `MA69E/ED` and `MA69F/FD` are used by `MA69A/AD`, `MA69B/BD` and `MA69C/CD` and `MA69D/DD`. The basic linear algebra subprograms (BLAS) `SAXPY/DAXPY` `SCOPY/DCOPY` `SDOT/DDOT` `SROT/DROT` and `SROTG/DROTG` are also called.

Input/output: none.

Restrictions: `N` ≥ 0, `M` $\geq$ `0` and `1` $\leq$ `ICLASS` ≤ 4. Also `LR` $\geq$ `M*(M+1)/2` and `LQ` $\geq$ `M` for `MA69A/AD`, `MA69B/BD` and `MA69D/DD`, `LR` $\geq$ `(M+1)*(M+2)/2` and `LQ` $\geq$ `M+1` for `MA69C/CD` and `1` $\leq$ `IRDROP` $\leq$ `M` and `1` $\leq$ `JCDROP` $\leq$ `M` for `MA69D/DD`

4 METHOD

The subroutine `MA69A/AD` forms the Schur complement $\mathbf{S}=\mathbf{D}-\mathbf{C}\,\mathbf{A}^{-1}\mathbf{B}$ of $\mathbf{A}$ in the extended matrix by repeated reverse communication to obtain the columns of $\mathbf{A}^{-1}\mathbf{B}$. The Schur complement or its negative is then factorized into its **QR** or, if possible, Cholesky factors.

The subroutine `MA69B/BD` solves the extended system using the following well-known scheme:

(i) Compute the solution to $\mathbf{Au}=\mathbf{b}_1$;

(ii) Compute $\mathbf{x}_2$ from $\mathbf{Sx}_2=\mathbf{b}_2-\mathbf{Cu}$;

(iii) Compute the solution to $\mathbf{Av}=\mathbf{Bx}_2$; and

(iv) Compute $\mathbf{x}_1=\mathbf{u}-\mathbf{v}$.

The subroutines `MA69C/CD` and `MA69D/DD` compute the factorization of the Schur complement after a row and column have been appended to, and removed from, the extended matrix, respectively. The existing factorization is updated to obtain the new one; this is normally more efficient than forming the factorization from scratch.

5 EXAMPLE OF USE

As a simple example, suppose we wish to solve the system of equations

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 5 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 0 & 1 & 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 7 \\ 8 \\ 10 \end{pmatrix}.$$

Notice that the leading 5 by 5 coefficient matrix is diagonal and hence easily invertible. So we might choose $n=5$, $m=2$ and use `MA69AD/MA69BD` to find the required solution. As the matrix is unsymmetric, we must set `ICLASS=1`.

Now suppose that we have solved this system as described and are now confronted with the further system

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\ 1 & 0 & 1 & 0 & 1 & 3 & 4 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \\ 4 \\ 5 \\ 7 \\ 12 \\ 12 \\ 4 \end{pmatrix}.$$

Rather than applying `MA69AD/MA69BD` with $n=5$, $m=3$, we note that the new coefficient matrix differs from the old one merely in having an extra row and column. Thus, we can use `MA69CD` with $n=5$, $m=2$ to update the existing factorization and then call `MA69BD` with $n=5$, $m=3$ to calculate the desired solution.

Finally, suppose that we have solved this second system and now wish to solve

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 5 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 3 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 4 \\ 5 \\ 6 \\ 6 \\ 2 \end{pmatrix}.$$

Again, rather than applying `MA69AD/MA69BD` with $n=5$, $m=2$, we note that this third coefficient matrix is the second one with its $n+1$–st row and $n+2$–nd column removed. Thus, we can use `MA69DD` with $n=5$, $m=3$ to update the existing factorization and then call `MA69BD` with $n=5$, $m=2$ to calculate the desired solution.

To carry out these calculations, we might use the following piece of code. Notice how the extra row and column for the second problem are simply introduced at the end of the existing data structures.

```
      PROGRAM MAIN
      INTEGER I, ICLASS, INFORM, IRDROP, JCDROP, LBD, LCD, LQ, LR
      INTEGER M, MPN, MPNP1, MP1, MP2, N, ISAVE
      DOUBLE PRECISION RSAVE
      PARAMETER ( N = 5, M = 2, ICLASS = 1 )
      PARAMETER ( LBD = 15, LCD = 13 )
      PARAMETER ( MP1 = M + 1, MP2 = MP1 + 1, MPN = M + N  )
      PARAMETER ( MPNP1 = MP1 + N, LQ = MP1, LR = MP1 * MP2 / 2 )
      INTEGER IRNBD( LBD ), JCNCD( LCD ), IPBD( MP2 ), IPCD( MP2 )
      DOUBLE PRECISION BD( LBD ), CD( LCD ), W( MP1 ),
     *                 R( LR ), Q( LQ, MP1 ), VECTOR( N ), X1( MPN ),
     *                 RHS1( MPN ), X2( MPNP1 ), RHS2( MPNP1 ),
     *                 X3( MPN ), RHS3( MPN )
      EXTERNAL MA69AD, MA69BD, MA69CD, MA69DD
      INTRINSIC DBLE
      DATA ( IPBD( I ), I = 1, 3 ) / 1, 7, 10 /
      DATA ( IPCD( I ), I = 1, 3 ) / 1, 6, 10 /
      DATA ( IRNBD( I ), I = 1, 9 ) / 1, 2, 3, 4, 5, 6, 5, 6, 7 /
      DATA ( BD( I ), I = 1, 9 ) / 1.0D+0, 1.0D+0, 1.0D+0, 1.0D+0,
     *          1.0D+0, 1.0D+0, 1.0D+0, 2.0D+0, 4.0D+0 /
      DATA ( JCNCD( I ), I = 1, 9 ) / 1, 2, 3, 4, 5, 1, 3, 5, 6 /
      DATA ( CD( I ), I = 1, 9 ) / 1.0D+0, 1.0D+0, 1.0D+0, 1.0D+0,
     *          1.0D+0, 1.0D+0, 1.0D+0, 1.0D+0, 3.0D+0 /
      DATA RHS1/ 2.0D+0, 3.0D+0, 4.0D+0, 5.0D+0, 7.0D+0, 8.0D+0, 1.0D+1/
      DATA RHS2/ 5.0D+0, 5.0D+0, 4.0D+0, 5.0D+0, 7.0D+0, 1.2D+1,
     *           1.2D+1, 4.0D+0 /
      DATA RHS3/ 3.0D+0, 5.0D+0, 4.0D+0, 5.0D+0, 6.0D+0, 6.0D+0, 2.0D+0/
C
```

```
C  FIRST PROBLEM.
C
      INFORM = 1
   10 CONTINUE
      CALL MA69AD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, W, VECTOR, ISAVE, INFORM )
      IF ( INFORM .GT. 0 ) THEN
         DO 20 I        = 1, N
            VECTOR( I ) = VECTOR( I ) / DBLE( I )
   20    CONTINUE
         GO TO 10
      END IF
      WRITE( 6, 2000 ) INFORM
      IF ( INFORM .LT. 0 ) STOP
      INFORM = 1
   30 CONTINUE
      CALL MA69BD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, RHS1, X1, VECTOR, INFORM )
      IF ( INFORM .GT. 0 ) THEN
         DO 40 I        = 1, N
            VECTOR( I ) = VECTOR( I ) / DBLE( I )
   40    CONTINUE
         GO TO 30
      END IF
      WRITE( 6, 2010 ) INFORM
      IF ( INFORM .LT. 0 ) STOP
      WRITE( 6, 2020 ) ( X1( I ), I = 1, MPN )
C
C  SECOND PROBLEM.
C
      IRNBD( 10 ) = 1
      BD( 10 )    = 1.0D+0
      IRNBD( 11 ) = 6
      BD( 11 )    = 1.0D+0
      IRNBD( 12 ) = 8
      BD( 12 )    = 1.0D+0
      IPBD( 4 )   = 13
      JCNCD( 10 ) = 1
      CD( 10 )    = 1.0D+0
      IPCD( 4 )   = 11
      INFORM      = 1
  110 CONTINUE
      CALL MA69CD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, W, VECTOR, RSAVE, INFORM )
      IF ( INFORM .GT. 0 ) THEN
         DO 120 I       = 1, N
            VECTOR( I ) = VECTOR( I ) / DBLE( I )
  120    CONTINUE
         GO TO 110
      END IF
      WRITE( 6, 2030 ) INFORM
      IF ( INFORM .LT. 0 ) STOP
      INFORM = 1
  130 CONTINUE
      CALL MA69BD( N, MP1, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, RHS2, X2, VECTOR, INFORM )
      IF ( INFORM .GT. 0 ) THEN
         DO 140 I       = 1, N
            VECTOR( I ) = VECTOR( I ) / DBLE( I )
  140    CONTINUE
         GO TO 130
      END IF
      WRITE( 6, 2010 ) INFORM
      IF ( INFORM .LT. 0 ) STOP
      WRITE( 6, 2040 ) ( X2( I ), I = 1, MPNP1 )
C
C  THIRD PROBLEM.
C
      IRDROP = 1
      JCDROP = 2
      INFORM = 1
      CALL MA69DD( N, MP1, ICLASS, .TRUE., LBD, BD, IRNBD, IPBD, JCDROP,
     *             LCD, CD, JCNCD, IPCD, IRDROP, R, LR, Q, LQ,
     *             W, VECTOR, INFORM )
      WRITE( 6, 2050 ) INFORM
      IF ( INFORM .LT. 0 ) STOP
      INFORM = 1
  210 CONTINUE
      CALL MA69BD( N, M, ICLASS, LBD, BD, IRNBD, IPBD, LCD, CD, JCNCD,
     *             IPCD, R, LR, Q, LQ, RHS3, X3, VECTOR, INFORM )
      IF ( INFORM .GT. 0 ) THEN
         DO 220 I       = 1, N
            VECTOR( I ) = VECTOR( I ) / DBLE( I )
  220    CONTINUE
         GO TO 210
      END IF
      WRITE( 6, 2010 ) INFORM
      IF ( INFORM .LT. 0 ) STOP
      WRITE( 6, 2060 ) ( X3( I ), I = 1, MPN )
      STOP
 2000 FORMAT( /, ' ON EXIT FROM MA69A, INFORM = ', I3 )
 2010 FORMAT(    ' ON EXIT FROM MA69B, INFORM = ', I3 )
 2020 FORMAT( /, ' SOLUTION (FIRST EQUATIONS)', /, ( 1P, 8D9.2 ) )
 2030 FORMAT( /, ' ON EXIT FROM MA69C, INFORM = ', I3 )
 2040 FORMAT( /, ' SOLUTION (SECOND EQUATIONS)', /, ( 1P, 8D9.2 ) )
 2050 FORMAT( /, ' ON EXIT FROM MA69D, INFORM = ', I3 )
 2060 FORMAT( /, ' SOLUTION (THIRD EQUATIONS)', /, ( 1P, 8D9.2 ) )
      END
```

This produces the following output:

```
 ON EXIT FROM MA69A, INFORM =   0
 ON EXIT FROM MA69B, INFORM =   0

 SOLUTION (FIRST EQUATIONS)
 1.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00

 ON EXIT FROM MA69C, INFORM =   0
 ON EXIT FROM MA69B, INFORM =   0

 SOLUTION (SECOND EQUATIONS)
 3.00D+00 2.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00

 ON EXIT FROM MA69D, INFORM =   0
 ON EXIT FROM MA69B, INFORM =   0

 SOLUTION (THIRD EQUATIONS)
 1.00D+00 2.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00 1.00D+00
```