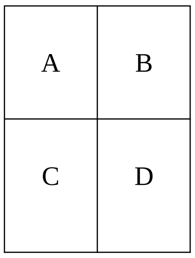
MA72

PACKAGE SPECIFICATION HSL

1 SUMMARY

This collection of subroutines, when used in conjunction with the `MA62` package, **solves symmetric positive-definite finite-element equations using a multiple front algorithm**. It is assumed that the underlying finite-element mesh has been partitioned into (non-overlapping) subdomains. In the multiple front algorithm, a frontal method is applied to each subdomain separately.

For example, a mesh could be divided into 4 subdomains A, B, C, D.



The variables that lie on the boundary between 2 or more subdomains are termed **interface variables** and those that lie within a subdomain are called **internal variables**. When the frontal method is applied to a subdomain, the internal variables can be eliminated but, at the end of the assembly and elimination processes, for each subdomain there remains a frontal matrix $\mathbf{F}_i$ and a corresponding right-hand side vector (or matrix) $\mathbf{c}_i$ that may be assembled to give a system of the form

$$\mathbf{F}\mathbf{y}=\mathbf{c}, \quad \mathbf{F}=\sum_i \mathbf{F}_i, \quad \mathbf{c}=\sum_i \mathbf{c}_i. \tag{1}$$

We call (1) the **interface** problem. By treating each of the matrices $\mathbf{F}_i$ as an element matrix, (1) may be solved using a frontal method (alternatively, any other solver for symmetric positive definite linear systems may be used to solve (1)). Once (1) has been solved, back-substitution on each subdomain completes the solution.

The application of the frontal method to each subdomain may be done in parallel. Using multiple fronts can also have the advantage of requiring less work than applying the frontal method to the whole domain.

`MA72` provides routines for generating lists of interface variables, for preserving the partial factorization of a matrix when the sequence of calls to the frontal solver factorization routine `MA62B/BD` is incomplete, and for performing forward elimination or back-substitution on a subdomain.

`MA72` uses reverse communication.

The use of HSL routine `MC53` to obtain an efficient element ordering in each subdomain is recommended before `MA62` and `MA72` are used.

For unsymmetric or symmetric indefinite problems, `MA52` should be used in conjunction with `MA42`.

For further details of the multiple front approach, see Duff, I. S. and Scott, J. A. (1994). *The use of multiple fronts in Gaussian elimination.* Rutherford Appleton Laboratory, Report RAL-94-040.

ATTRIBUTES — **Version:** 1.0.0. (12 July 2004) **Types:** Real (single, double). **Calls:** `MA62`. **Helpful:** `MC53`. **Language:** Fortran 77. **Original date:** May 1998. **Origin:** J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences.

There are three entries:

(a) The use of MA72A/AD is optional. For each subdomain, MA72A/AD generates a *guard element* that is, a list of interface variables. If MA72A/AD is used, it must be called once for each element in the finite-element mesh and the user must indicate to which subdomain each element belongs. The calls may be made in any order.

(b) If subroutines MA62A/AD and MA62J/JD from the MA62 frontal package are called by the user *k* times, *k* calls to the factorization subroutine MA62B/BD are needed to complete the matrix factorization. If the user makes fewer than *k* calls to MA62B/BD, MA72B/BD may be used to preserve the partial factorization. The frontal matrix and corresponding frontal right-hand side matrix at the point at which the sequence of calls to MA62B/BD was terminated are returned to the user, together with the partial factorization. If direct access files were being used by MA62, the partial factorization is written to the direct access files. The routine may only be used if no errors have been returned from MA62B/BD.

(c) MA72C/CD performs forward elimination or back-substitution on a subdomain of a finite-element mesh. The routine can only be called after an earlier call to MA72B/BD for the subdomain.

In the multiple front algorithm, the frontal solver analyse routine MA62A/AD is called once for each of the *nelt* elements in a subdomain. An extra call to MA62A/AD is then made for the appropriate guard element. Similarly, *nelt*+1 calls are made to the symbolic factorization routine MA62J/JD. *nelt* calls are then made to the factorization subroutine MA62B/BD. Since MA62B/BD expects *nelt*+1 calls (the number of calls made to MA62A/AD), the factorization after *nelt* calls is incomplete. This partial factorization is preserved by calling MA72B/BD. The output from MA72B/BD is in the form of an element matrix $\mathbf{F}_i$. In addition, if MA62B/BD was called with a positive number of right-hand sides ($\texttt{NRHSB} > 0$), the corresponding partial solution is output as an element right-hand side matrix $\mathbf{C}_i$ (or vector $\mathbf{c}_i$ for a single right-hand side). For each subdomain there will be one element matrix and, if $\texttt{NRHSB} > 0$, one element right-hand side matrix.

The interface problem may be solved using the frontal code MA62. Alternatively, the element matrices $\mathbf{F}_i$ may be assembled to give a system of the form (2) and any other suitable package for symmetric positive-definite systems can be used to factor the interface matrix and to solve for the interface variables.

Once the interface problem has been solved, the user should call MA72C/CD for each subdomain to perform the back-substitution needed to solve for the internal variables. MA72C/CD may also be used to solve for further right-hand sides. In this case, MA72C/CD is used to perform forward elimination on each subdomain. Using the partial solution on the subdomains, the user must then solve for the interface variables (using, for example, the frontal solver subroutine MA62C/CD), and then recall MA72C/CD on each subdomain for the final back-substitution for the internal variables.

The calling sequence is illustrated in Section 5.

2.1.1 Generation of guard elements

If the user chooses to use MA72A/AD to obtain the guard elements, a call of the following form must be made for each element in the finite-element mesh. The elements may be presented in any order.

The single precision version

```
      CALL MA72A(ICALL,NVAR,MAXIND,IVAR,TOTELT,NDOMN,IDOMN,NGUARD,LGUARD,
     *           IGUARD,IW,JCNTL,JNFO)
```

The double precision version

```
      CALL MA72AD(ICALL,NVAR,MAXIND,IVAR,TOTELT,NDOMN,IDOMN,NGUARD,LGUARD,
     *           IGUARD,IW,JCNTL,JNFO)
```

`ICALL` is an `INTEGER` variable that must be set by the user to the number of the call to the subroutine. On the first call `ICALL` must be set to `1`, on the second call to `2`, and so on. This argument is not altered by the routine.

`NVAR` is an `INTEGER` variable that must be set by the user to the number of variables in the current element. This argument is not altered by the routine. **Restriction:** $\texttt{NVAR} \geq 1$.

`MAXIND` is an `INTEGER` variable that must be set by the user to be at least as large as the largest integer used to index a variable in the finite-element mesh. This argument must be unchanged between calls to `MA72A/AD` and is not altered by the routine.

`IVAR` is an `INTEGER` array of length at least `NVAR` that must be set by the user to contain the indices of the variables associated with the current element. This argument is not altered by the routine. **Restriction:** $1 \leq \texttt{IVAR(I)} \leq \texttt{MAXIND}$, `I` = `1`, 2,..., `NVAR`.

`TOTELT` is an `INTEGER` variable that must be set by the user to the total number of elements in the finite-element mesh. This argument must be unchanged between calls to `MA72A/AD` and is not altered by the routine. **Restriction:** $\texttt{TOTELT} > 1$.

`NDOMN` is an `INTEGER` variable that must be set by the user to the number of the subdomains that comprise the finite-element mesh. This argument must be unchanged between calls to `MA72A/AD` and is not altered by the routine. **Restriction:** $\texttt{NDOMN} > 1$.

`IDOMN` is an `INTEGER` variable that must be set by the user to the index of the subdomain to which the current element belongs. This argument is not altered by the routine. **Restriction:** $1 \leq \texttt{IDOMN} \leq \texttt{NDOMN}$.

`NGUARD` is an `INTEGER` array of length `NDOMN` that need not be set by the user. On exit from the final call (the call with `ICALL=TOTELT`), `NGUARD(IDOMN)` holds the number of variables in the guard element for subdomain `IDOMN`, that is, the number of variables that lie on the interface for subdomain `IDOMN` (`IDOMN` = `1`, 2,..., `NDOMN`). This argument must be unchanged between calls to `MA72A/AD`.

`LGUARD` is an `INTEGER` variable that must be set by the user to the first dimension of the array `IGUARD`. If the guard element for subdomain `JDOMN` has the most variables, `LGUARD` must be at least `NGUARD(JDOMN)`, but in practice the user needs to choose a value larger than this. This argument must be unchanged between calls to `MA72A/AD` and is not altered by the routine.

`IGUARD` is an `INTEGER` array of dimensions `LGUARD` by `NDOMN` that need not be set by the user. On exit from the final call, `IGUARD(I,IDOMN)`, `I` = `1`, 2,..., `NGUARD(IDOMN)`, is a list of the variables in the guard element for subdomain `IDOMN` (`IDOMN` = `1`, 2,..., `NDOMN`). This argument must be unchanged between calls to `MA72A/AD`.

`IW` is an `INTEGER` array of length `MAXIND` that is used by the routine as workspace. This argument must be unchanged between calls to `MA72A/AD`.

`JCNTL` is an `INTEGER` array of length `2`. `JCNTL(1)` must be set by the user to the stream number for the printing of messages. Printing is suppressed if $\texttt{JCNTL(1)} < 0$. `JCNTL(2)` is used to control the level of printing. `JCNTL(2)` must be set by the user to one of the following values:

`0` No messages are output.

`1` Error messages output.

`2` As for `1`, plus scalar parameters on the first entry to `MA72A/AD`.

This argument is not altered by the routine.

`JNFO` is an `INTEGER` array of length `2` that need not be set by the user. On successful exit, `JNFO(1)` is set to `0`. Negative values of `JNFO(1)` indicate an error. $\texttt{JNFO(1)} = 1$ is associated with a non-terminal error. In the event of an error, `JNFO(2)` is used to hold further information. Possible non-zero values of `JNFO(1)` are:

`+1` The defined first dimension `LGUARD` of the array `IGUARD` is insufficient. If the user continues to call `MA72A/AD`, on exit from the final call a value that is sufficient will be returned in `JNFO(2)` (see error `-6`).

- -1 Value of TOTELT out of range. JNFO(2) holds TOTELT. Immediate return with remaining input parameters unchanged. Note that TOTELT is only checked on the first entry to MA72A/AD.
- -2 Value of NDOMN out of range. JNFO(2) holds NDOMN. Immediate return with remaining input parameters unchanged. Note that NDOMN is only checked on the first entry to MA72A/AD.
- -3 Value of NVAR out of range. JNFO(2) holds NVAR. Immediate return with remaining input parameters unchanged. Note that NVAR is only checked on the first entry to MA72A/AD.
- -4 One or more variable indices in the current element is out of range. JNFO(2) holds the number that are out of range. Immediate return with remaining input parameters unchanged.
- -5 Value of IDOMN out of range. JNFO(2) holds IDOMN.
- -6 Defined first dimension LGUARD of the array IGUARD is insufficient. However, the sequence of calls to MA72A/AD has been completed and a value that is sufficient is given in JNFO(2).

2.1.2 Preservation of the partial factorization.

MA72B/BD may be called to preserve the partial factorization of a matrix when the sequence of calls to the frontal solver factorization routine MA62B/BD has not been completed (provided no errors have been issued by MA62B/BD). For the multiple front algorithm, if there are *nelt* elements in a subdomain, MA72B/BD should be called once after the *nelt* calls to MA62B/BD. MA72B/BD must be called for each subdomain.

The single precision version

```
CALL MA72B(NRHSB,NDF,NFVAR,LAST,LX,X,LIW,IW,LW,W,ISAVE,JCNTL,JNFO)
```

The double precision version

```
CALL MA72BD(NRHSB,NDF,NFVAR,LAST,LX,X,LIW,IW,LW,W,ISAVE,JCNTL,JNFO)
```

NRHSB and NDF are INTEGER variables that must be unchanged since the calls to the frontal solver factorization routine MA62B/BD. These arguments are not altered by the routine.

NFVAR is an INTEGER that need not be set by the user. On exit, NFVAR holds the number of variables remaining in the front after the last call to the frontal solver factorization routine MA62B/BD.

LAST is an INTEGER array of length NDF that must be unchanged since the last call to the frontal solver factorization routine MA62B/BD. On exit, LAST is restored to the values it contained on exit from the final call to the frontal solver analyse routine MA62A/AD.

LX is an INTEGER variable that must be set by the user to the first dimension of array X. This argument is not altered by the routine. **Restriction:** LX $\geq$ NDF.

X is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX by NRHSB that need not be set by the user. This argument is changed by the routine. X is not accessed if NRHSB = 0.

LIW is an INTEGER variable that must be unchanged since the last call to the frontal solver factorization routine MA62B/BD. This argument is not altered by the routine.

IW is an INTEGER array of length LIW that must be unchanged since the last call to the frontal solver factorization routine MA62B/BD. On exit, IW(ISAVE(20)+I-1), I = 1, 2,..., NFVAR, is the index of the Ith variable remaining in the front after the last call to MA62B/BD.

LW is an INTEGER variable that must be unchanged since the last call to the frontal solver factorization routine MA62B/BD. This argument is not altered by the routine.

W is a REAL (DOUBLE PRECISION in the D version) array of length LW that must be unchanged since the last call to the frontal solver factorization routine MA62B/BD. On exit, W contains the upper triangular part of the frontal matrix and the corresponding frontal right-hand side matrix remaining after the last call to MA62B/BD. Specifically, if NFRONT = ISAVE(27), W(ISAVE(18)+I-1+(J-1)*NFRONT) contains the value of the (I, J)th entry in the upper triangular part of the frontal matrix (I = 1, 2,..., J, J = 1, 2,..., NFVAR). In addition, W(ISAVE(19)+I-1+(J-1)*NFRONT) contains the value of the Ith entry in the J-th frontal right-hand side matrix (I = 1, 2,..., NFVAR, J = 1, 2,..., NRHSB).

ISAVE is an INTEGER array of length 50 that must be unchanged since the last call to the frontal solver factorization routine MA62B/BD. Only ISAVE(50) is altered by MA72B/BD.

JCNTL is an INTEGER array of length 2. JCNTL(1) must be set by the user to the stream number for the printing of messages. Printing is suppressed if $JCNTL(1) < 0$. JCNTL(1) should not be equal to the stream number of either of the direct access files used by MA62B/BD (but no check is made for this). JCNTL(2) is used to control the level of printing. JCNTL(2) must be set by the user to one of the following values:

0 No messages are output.

1 Error messages output.

2 As for 1, plus scalar parameters on entry to MA72B/BD. This argument is not altered by the routine.

JNFO is an INTEGER array of length 2 that need not be set by the user. On successful exit, JNFO(1) is set to 0. Negative values of JNFO(1) indicate an error. In the event of an error, JNFO(2) is used to hold further information. Possible non-zero values of JNFO(1) are:

-1 MA72B/AD has been called after an error was issued by the frontal solver factorization routine MA62B/BD. JNFO(2) holds the error flag issued by MA62B/BD. Immediate return with remaining input parameters unchanged.

-2 The sequence of calls to the frontal solver factorization routine MA62B/BD was completed before MA72B/AD was called. Immediate return with input parameters unchanged.

-3 NDF has been changed since the calls to the frontal solver factorization routine MA62B/BD. JNFO(2) holds the value of NDF used in the calls to MA62B/BD. Immediate return with remaining input parameters unchanged.

-4 NRHSB has been changed since the calls to the frontal solver factorization routine MA62B/BD. JNFO(2) holds the value of NRHSB used in the calls to MA62B/BD. Immediate return with remaining input parameters unchanged.

-5 LX is out of range. JNFO(2) is set to the minimum possible value for LX. Immediate return with remaining input parameters unchanged.

-6 Error detected when writing to a direct access file. The iostat parameter is returned in JNFO(2).

2.1.3 Forward elimination and back-substitution on a subdomain

To perform forward elimination or back-substitution on a subdomain, a call of the following form must be made.

The single precision version

```
CALL MA72C(JOB,NRHSC,LX,X,LW,W,LIW,IW,ISAVE,JCNTL,JNFO)
```

The double precision version

```
CALL MA72CD(JOB,NRHSC,LX,X,LW,W,LIW,IW,ISAVE,JCNTL,JNFO)
```

JOB is an INTEGER variable that must be set by the user. Possible values are JOB = 1, 2, and 3. If the user was using the frontal solver factorization routine MA62B/BD to solve for a number of right-hand sides at the same time as factorizing the matrix (that is, MA62B/BD was called with $NRHSB > 0$), JOB should be set to 1 to perform back-substitution on the subdomain. JOB = 2 and 3 are used when solving for further right-hand sides. If the user wishes to perform forward elimination, JOB should be set to 2. If the user wishes to perform back-substitution having already computed the solution to the interface problem, JOB should be set to 3. A call with JOB = 3 must be preceded by one with JOB = 2. **Restriction:** $1 \le JOB \le 3$.

NRHSC is an INTEGER variable. If JOB = 1, NRHSC must be equal to NRHSB, the number of right-hand sides on the calls to the frontal solver factorization routine MA62B/BD. Otherwise, NRHSC must be set by the user to the number of right-hand sides, and must have the same value on the call with JOB = 3 as on the call with JOB = 2. This argument is not altered by the routine. **Restriction:** NRHSC > 0.

LX is an INTEGER variable that must be set by the user to the first dimension of X. LX must be at least as large as the largest integer used to index a variable (that is, as large as the value of NDF on exit from the final call to the frontal solver analyse routine MA62A/AD for the subdomain). This argument is not altered by the routine.

X is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX, NRHSC.

If JOB = 1, if L is used to index an interface variable, X(L,J) must hold the solution for variable L for system J (J = 1, 2,..., NRHSC). On exit, X(L,J) is unchanged and if K is used to index an internal variable, X(K,J) holds the solution for variable K to system J (J = 1, 2,..., NRHSC).

If JOB = 2, on entry X must be set by the user so that if K has been used to index a variable in the subdomain, X(K,J) is the corresponding component of the right-hand side for the J-th system (J = 1, 2,..., NRHSC). On exit, if K is used to index an internal variable, X(K,J) holds the partial solution for variable K for system J.

If JOB = 3, on entry X must be set by the user so that if K is an internal variable, X(K,J) is the corresponding component of the partial solution for the J-th system and if L is used to index an interface variable, X(L,J) must hold the solution for variable L for the J-th system (J = 1, 2,..., NRHSC). On exit, X(L,J) is unchanged and X(K,J) holds the solution for variable K to system J (J = 1, 2,..., NRHSC).

LW is an INTEGER variable that must be set by the user to the dimension of the array W. A sufficient value for LW is L1+L2, where L1 = NRHSC*ISAVE(27) (ISAVE(27) holds the maximum frontsize on the subdomain). If direct access files were not used by MA62, L2 = 3 + ISAVE(5) (ISAVE(5) holds the length of the real buffer), otherwise L2 = ISAVE(5) + ISAVE(17)*(ISAVE(27) + NRHSB) (ISAVE(17) holds the maximum pivot block size and NRHSB is the nuber of right-hand sides on the calls to MA42B/BD). This argument is not altered by the routine. **Restriction:** LW ≥ L1 + L2.

W is a REAL (DOUBLE PRECISION in the D version) array of length LW. If direct access files were not used by MA62, the first ISAVE(5) entries of W must be unchanged since the call to MA72B/BD and these entries are unchanged by MA72C/CD, otherwise W is used by MA72C/CD as workspace.

LIW is an INTEGER variable that must be set by the user to the dimension of the array IW. If direct access files were not used by MA62, LIW must be at least L3 = ISAVE(6) (ISAVE(6) holds the length of the integer buffer). Otherwise, LIW must be at least L3 = 4 + ISAVE(6) + ISAVE(27). This argument is not altered by the routine. **Restriction:** LIW ≥ L3.

IW is an INTEGER array of length LW. If direct access files were not used by MA62, the first ISAVE(6) entries of IW must be unchanged since the call to MA72B/BD and these entries are unchanged by MA72B/BD. Otherwise, IW is used by MA72C/CD as workspace.

ISAVE is an INTEGER array of dimension 50 that must be unchanged since the call to MA72B/BD. This argument is not altered by the routine.

JCNTL is an INTEGER array of length 2. JCNTL(1) must be set by the user to the stream number for the printing of messages. Printing is suppressed if JCNTL(1) < 0. JCNTL(1) should not be equal to the stream number of any of the direct access files used by MA62B/BD (but no check is made for this). JCNTL(2) is used to control the level of printing. JCNTL(2) must be set by the user to one of the following values:

0 No messages are output.

1 Error messages output.

2 As for 1, plus scalar parameters on entry to MA72C/CD.

This argument is not altered by the routine.

JNFO is an INTEGER array of length 2 that need not be set by the user. On successful exit, JNFO(1) is set to 0. Negative values of JNFO(1) indicate an error. In the event of an error, JNFO(2) is used to hold further information. Possible non-zero values of JNFO(1) are:

-1 The call to MA72C/CD does not follow an earlier call to MA72B/BD. Immediate return with input parameters unchanged.

-2 First dimension LX of the array X too small. JNFO(2) holds a sufficient value. Immediate return with remaining input parameters unchanged.

-3 Defined length LW of the array W violates the restrictions on LW. JNFO(2) holds a sufficient value. Immediate return with remaining input parameters unchanged.

-4 Defined length LIW of the array IW violates the restrictions on LIW. JNFO(2) holds a sufficient value. Immediate return with remaining input parameters unchanged.

-5 Either the number of right-hand sides has been changed since the call to MA72B/BD (JOB = 1) or NRHSC ≤ 0 (JOB = 2 or 3). If JOB = 1, JNFO(2) holds the number of right-hand sides on the call to MA72B/BD. Otherwise, JNFO(2) holds NRHSC. Immediate return with remaining input parameters unchanged.

-6 Error detected when reading from a direct access file. The iostat parameter is returned in JNFO(2).

-7 Value of JOB out of range. JNFO(2) holds JOB. Immediate return with remaining input parameters unchanged.

3 GENERAL INFORMATION

3.1 Summary of information.

Other routines called directly: MA72A/AD calls no other routines. MA72B/BD and MA72C/CD use internal routines for the MA62 package. MA72B/BD calls the internal subroutines MA72D/DD and MA72E/ED, as well as MA62L/LD from the MA62 package. MA72C/CD calls routines MA62D/DD, MA62E/ED, and MA62L/LD from the MA62 package.

Input/output: In the event of errors, diagnostic messages are printed on unit JCNTL(1).

Restrictions:

Restrictions for MA72A/AD**:**
TOTELT ≥ 2.
NDOMN ≥ 2.
NVAR ≥ 1.
1 ≤ IVAR(I) ≤ MAXIND, I = 1, 2,..., NVAR.
1 ≤ IDOMN ≤ NDOMN.

Restrictions for MA72B/BD**:** LX ≥ NDF and, in addition, the routine requires that the arguments NRHSB, NDF, LAST, LW, W, LIW, IW, and ISAVE are unchanged since the last call to the frontal solver factorization routine MA62B/BD. The routine checks that NRHSB and NDF are unchanged.

Restrictions for MA72C/CD**:**
NRHSC ≥ 1.
1 ≤ JOB ≤ 3.
If direct access files are being used,
LW ≥ NRHSC* ISAVE(27) + ISAVE(5) + ISAVE(17)*(ISAVE(27) + NRHSB).
LIW ≥ 4 + ISAVE(6) + ISAVE(27).

Otherwise,

$\texttt{LW} \geq \texttt{NRHSC* ISAVE(27)} + 3 + \texttt{ISAVE(5)}$.

$\texttt{LIW} \geq \texttt{ISAVE(6)}$.

4 METHOD

MA72A/AD

On the initial call to MA72A/AD the input data is checked for errors. If no errors are found, the arrays NGUARD and IW are initialised to zero. When an element is entered, each of its variables J is considered in turn. If IW(J) is equal to 0, variable J is being encountered for the first time and IW(J) is set to IDOMN, the index of the subdomain to which the current element belongs. If IW(J) is nonzero and not equal to IDOMN, say IW(J) = JDOMN, then variable J has already been entered in subdomain JDOMN and so must lie on the interface between subdomains IDOMN and JDOMN. In this case, NGUARD(IDOMN) and NGUARD(JDOMN) are incremented by one, and J is added to the lists of interface variables held in columns IDOMN and JDOMN of the matrix IGUARD. Once all the elements have been entered, any duplicated entries in the interface lists are removed.

MA72B/BD

MA72B/BD first checks that the sequence of calls to the frontal solver factorization routine MA62B/BD has not been completed and that an error has not been issued by MA62B/BD. This is done using information held in the array ISAVE. The input parameters are then checked, again using the array ISAVE. If direct access files are being used (this information is held in ISAVE), the code then calls MA62L/LD to write out the contents of the buffers to the direct access files.

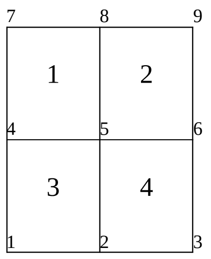
The array LAST is set to hold the values it would have contained if the sequence of calls to MA62B/BD had been completed and, if the number of right-hand sides NRHSB is nonzero, the first NRHSB columns of the array X are initialised to zero.

MA72C/CD

A check is first made that MA72B/BD was previously called for the subdomain. This information is held in ISAVE. The input parameters are then checked. Again, this is done using ISAVE. The workspace is partitioned according to whether or not direct access files are being used and the value of the input parameter JOB. If JOB = 1 or 3, MA62D/DD from the MA62 package is called to perform back-substitution, and if JOB = 2, MA62E/ED performs forward elimination.

5 EXAMPLE OF USE

We wish to solve the following simple finite-element problem in which the finite-element mesh comprises four 4-noded quadrilateral elements with one freedom at each node i, $1 \leq i \leq 6$ (the nodes 7, 8, and 9 are assumed constrained). The mesh is divided into 2 subdomains in which elements 1 and 2 comprise subdomain 1 and elements 3 and 4 comprise subdomain 2.



The four element matrices $\mathbf{A}^{(k)}$ $(1 \leq k \leq 4)$ are

$$\begin{array}{c}4\\5\end{array}\begin{pmatrix}2. & 1.\\1. & 7.\end{pmatrix}\qquad \begin{array}{c}5\\6\end{array}\begin{pmatrix}3. & 2.\\2. & 8.\end{pmatrix}\qquad \begin{array}{c}4\\5\\1\\2\end{array}\begin{pmatrix}4. & 3. & 2. & 3.\\3. & 1. & 3. & 2.\\2. & 3. & 6. & 1.\\3. & 2. & 1. & 5.\end{pmatrix}\qquad \begin{array}{c}5\\6\\2\\3\end{array}\begin{pmatrix}2. & 1. & 8. & 3.\\1. & 3. & 2. & 2.\\8. & 2. & 2. & 5.\\3. & 2. & 5. & 4.\end{pmatrix},$$

where the variable indices are indicated by the integers before each matrix (columns are identified symmetrically to rows). The corresponding element right-hand side vectors $\mathbf{b}^{(k)}$ $(1 \le k \le 4)$ are

$$\begin{pmatrix}3.\\8.\end{pmatrix}\qquad \begin{pmatrix}5.\\10.\end{pmatrix}\qquad \begin{pmatrix}12.\\9.\\12.\\11.\end{pmatrix}\qquad \begin{pmatrix}14.\\8.\\17.\\14.\end{pmatrix}.$$

The following program is used to solve this problem. In this program, we read the element data into arrays ELTPTR (location of first entry of element), ELTVAR (variable indices), VALPTR (location of first numerical value for element), VALUE (numerical values), and RHSVAL (right-hand sides). This method of storing the element data is used here for **illustrative purposes only**; the user may prefer, for example, to read in the element data from a direct access file. In practice, the user should reorder the elements in each subdomain before calling MA62. This may be done using routine MC53.

```
C Code to run MA62 on a finite element mesh composed of
C two subdomains.

C     .. Parameters ..
      INTEGER LRHS,LWMX,LIWMX,LAVAR,LFVAR,LGUARD,MAXIND,MAXVL,MAXRVL,
     +        MELT,NDOMN,NSUB,NZMAX
      PARAMETER (LRHS=1,LWMX=1200,LIWMX=1200,LAVAR=4,LFVAR=9,LGUARD=3,
     +          MAXIND=9,MAXVL=30,MAXRVL=15,MELT=4,NDOMN=2,NSUB=4,
     +          NZMAX=30)
C     ..
C     .. Local Scalars ..
      INTEGER I,IDOMN,J,JFILE,NDFTOT,NFRONT,NFVAR,NRHSB,NRHSC,NZ,TOTELT
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION AVAR(LAVAR,LAVAR),
     +                 FRHS(LFVAR,LRHS),FVAR(LFVAR,LFVAR),
     +                 RHSVAL(MAXRVL),RINFO(20),RKNFO(20),
     +                 VALUE(MAXVL),W(LWMX),X(MAXIND,LRHS)
      INTEGER ELTPTR(MELT+1),ELTVAR(NZMAX),IFVAR(LFVAR),
     +        IGUARD(LGUARD,NDOMN),INFO(20),ISAVE(50,NDOMN),
     +        IW(LIWMX),JCNTL(2),JNFO(2),KCNTL(15),
     +        KNFO(20),KSAVE(50),LAST(MAXIND),NELT(NDOMN),
     +        NGUARD(NDOMN),NLIST(NSUB,NDOMN),VALPTR(MELT)
C     ..
C     .. External Subroutines ..
      EXTERNAL GUARD,INTERF,MA62CD,MA72CD,READIN,SUBDOM
C     ..
C Stream for error messages.
      JCNTL(1) = 6
      JCNTL(2) = 1
C Read in matrix data
      CALL READIN(NDFTOT,TOTELT,NDOMN,NSUB,NELT,NLIST,MELT,ELTPTR,NZMAX,
     +            ELTVAR,NZ,VALPTR,MAXVL,VALUE,MAXRVL,RHSVAL)

C Generate the guard elements.
      CALL GUARD(NDOMN,NSUB,NELT,NLIST,TOTELT,ELTPTR,NZMAX,ELTVAR,
     +          NGUARD,LGUARD,IGUARD,MAXIND,IW,JCNTL,JNFO)
      IF (JNFO(1).LT.0) GO TO 50

C Run MA62 on each subdomain. This can be done in parallel.
      NRHSB = 1
      DO 10 IDOMN = 1,NDOMN
         CALL SUBDOM(IDOMN,NELT(IDOMN),NLIST(1,IDOMN),TOTELT,ELTPTR,NZ,
     +               ELTVAR,VALPTR,MAXVL,VALUE,MAXRVL,RHSVAL,MAXIND,
     +              LAST,NGUARD(IDOMN),IGUARD(1,IDOMN),LGUARD,LRHS,
     +              NRHSB,LWMX,W,LIWMX,IW,MAXIND,X,LAVAR,AVAR,
     +              ISAVE(1,IDOMN),INFO,RINFO,NFVAR,JCNTL,JNFO)
         IF (INFO(1).LT.0) GO TO 50
         IF (JNFO(1).LT.0) GO TO 50

C Write the interface element matrix and element right hand side
C vector out to a file.
         JFILE = 8
         NFRONT = ISAVE(27,IDOMN)
         WRITE (JFILE) NFVAR, (IW(ISAVE(20,IDOMN)+I-1), I=1,NFVAR),
     +     ((W(ISAVE(19,IDOMN)+I-1+(J-1)*NFRONT),
     +       I=1,NFVAR), J=1,NRHSB),
     +     ((W(ISAVE(18,IDOMN)+I-1+(J-1)*NFRONT), I=1,J), J=1,NFVAR)
   10 CONTINUE
      REWIND (JFILE)

C Solve interface problem using MA62.
      CALL INTERF(JFILE,NDOMN,NFVAR,LFVAR,IFVAR,FVAR,FRHS,MAXIND,LAST,
     +           LRHS,NRHSB,LWMX,W,LIWMX,IW,MAXIND,X,KCNTL,KSAVE,KNFO,
     +           RKNFO)
      IF (KNFO(1).LT.0) GO TO 50

C Now perform backsubstitution on each subdomain
      DO 20 IDOMN = 1,NDOMN
         CALL MA72CD(1,NRHSB,MAXIND,X,LWMX,W,LIWMX,IW,ISAVE(1,IDOMN),
     +              JCNTL,JNFO)
         IF (JNFO(1).LT.0) GO TO 50
   20 CONTINUE

C  Solution is in first NDFTOT locations of X
      WRITE (*,FMT='(/A)') ' The solution is:'
      WRITE (*,FMT='(/6(1X,F6.3))') (X(I,1),I=1,NDFTOT)

C Now solve for a further right-hand side.
      WRITE (*,FMT='(/A)') ' Now solving for a further right-hand side.'
      NRHSC = 1
C Read in (assembled) right-hand side
      READ (*,FMT=*) (X(I,1),I=1,NDFTOT)

C Forward substitution on subdomains
      DO 30 IDOMN = 1,NDOMN
         CALL MA72CD(2,NRHSC,MAXIND,X,LWMX,W,LIWMX,IW,ISAVE(1,IDOMN),
     +              JCNTL,JNFO)
         IF (JNFO(1).LT.0) GO TO 50
   30 CONTINUE

C Call MA62C/CD for interface problem
      CALL MA62CD(NRHSC,MAXIND,X,LWMX,W,LIWMX,IW,KCNTL,KSAVE,KNFO)
      IF (KNFO(1).LT.0) GO TO 50

C Back substitution on subdomains
      DO 40 IDOMN = 1,NDOMN
         CALL MA72CD(3,NRHSC,MAXIND,X,LWMX,W,LIWMX,IW,ISAVE(1,IDOMN),
     +              JCNTL,JNFO)
         IF (JNFO(1).LT.0) GO TO 50
   40 CONTINUE
      WRITE (*,FMT='(/A)') ' The solution is:'
      WRITE (*,FMT='(/6(1X,F6.3))') (X(I,1),I=1,NDFTOT)
   50 CONTINUE

      STOP
      END

C*******************************************************************
      SUBROUTINE READIN(NDFTOT,TOTELT,NDOMN,NSUB,NELT,NLIST,MELT,ELTPTR,
     +                 NZMAX,ELTVAR,NZ,VALPTR,MAXVL,VALUE,MAXRVL,
     +                 RHSVAL)
```

```
C Subroutine to read in element data

C     .. Scalar Arguments ..
      INTEGER MAXRVL,MAXVL,MELT,NDFTOT,NDOMN,NSUB,NZMAX,NZ,TOTELT
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION RHSVAL(MAXRVL),VALUE(MAXVL)
      INTEGER ELTPTR(MELT+1),ELTVAR(NZMAX),NELT(NDOMN),
     +        NLIST(NSUB,NDOMN),VALPTR(MELT)
C     ..
C     .. Local Scalars ..
      INTEGER I,IDOMN,RHSCRD,VALCRD
C     ..
C Read in the number of variables and elements in the problem.
      READ (*,FMT=*) NDFTOT,TOTELT
C Read in elements. NELT(IDOMN) is the number of elements in subdomain
C IDOMN and NLIST is used to hold lists of the elements in each
C subdomain.
      READ (*,FMT=*) (NELT(IDOMN),IDOMN=1,NDOMN)
      DO 10 IDOMN = 1,NDOMN
         READ (*,FMT=*) (NLIST(I,IDOMN),I=1,NELT(IDOMN))
   10 CONTINUE

C ELTVAR contains lists of the variables belonging to the
C elements, with those for element 1 preceding those for element
C 2, and so on. ELTPTR(I) points to the position in ELTVAR
C of the first variable in element I. NZ is the total number
C of entries in the element lists.

      READ (*,FMT=*) (ELTPTR(I),I=1,TOTELT+1)
      NZ = ELTPTR(TOTELT+1) - 1
      READ (*,FMT=*) (ELTVAR(I),I=1,NZ)

C VALCRD is the number of numerical values to be input.
C VALUE contains lists of the numerical values in the elemental
C matrices, with element 1 preceding element 2, and so on.
C Since the elemental matrices are symmetric only the upper
C triangular part is needed. VALPTR(I) points to the position in
C VALUE of the first value for element I.

      READ (*,FMT=*) (VALPTR(I),I=1,TOTELT)
      READ (*,FMT=*) VALCRD
      READ (*,FMT=*) (VALUE(I),I=1,VALCRD)

C RHSCRD is the number of right-hand side numerical values to
C be input. RHSVAL contains lists of the right-hand side
C numerical values corresponding to each of the elements in order.

      READ (*,FMT=*) RHSCRD
      READ (*,FMT=*) (RHSVAL(I),I=1,RHSCRD)
      RETURN
      END

C*******************************************************************
      SUBROUTINE GUARD(NDOMN,NSUB,NELT,NLIST,TOTELT,ELTPTR,NZMAX,ELTVAR,
     +                 NGUARD,LGUARD,IGUARD,MAXIND,IW,JCNTL,JNFO)

C Subroutine to generate the guard elements using MA72A/AD.

C     .. Scalar Arguments ..
      INTEGER LGUARD,MAXIND,NDOMN,NSUB,NZMAX,TOTELT
C     ..
C     .. Array Arguments ..
      INTEGER ELTPTR(TOTELT+1),ELTVAR(NZMAX),IGUARD(LGUARD,NDOMN),
     +        IW(MAXIND),JCNTL(2),JNFO(2),NELT(NDOMN),NGUARD(NDOMN),
     +        NLIST(NSUB,NDOMN)
C     ..
```

```
C     .. Local Scalars ..
      INTEGER I,ICALL,IDOMN,IELT,JSTRT,NVAR
C     ..
C     .. External Subroutines ..
      EXTERNAL MA72AD
C     ..
      ICALL = 0
      DO 20 IDOMN = 1,NDOMN
         DO 10 I = 1,NELT(IDOMN)
            IELT = NLIST(I,IDOMN)
            NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
            JSTRT = ELTPTR(IELT)
            ICALL = ICALL + 1
            CALL MA72AD(ICALL,NVAR,MAXIND,ELTVAR(JSTRT),TOTELT,NDOMN,
     +                  IDOMN,NGUARD,LGUARD,IGUARD,IW,JCNTL,JNFO)
            IF (JNFO(1).LT.0) RETURN
   10    CONTINUE
   20 CONTINUE
      RETURN
      END

C*******************************************************************
      SUBROUTINE SUBDOM(IDOMN,NELT,NLIST,TOTELT,ELTPTR,NZ,ELTVAR,VALPTR,
     +                  MAXVL,VALUE,MAXRVL,RHSVAL,MAXIND,LAST,NGUARD,
     +                  IGUARD,LGUARD,LRHS,NRHS,LWMX,W,LIWMX,IW,LX,X,
     +                  LAVAR,AVAR,ISAVE,INFO,RINFO,NFVAR,JCNTL,JNFO)

C Subroutine to run MA62 on a subdomain and preserve partial
C factorization using MA72B/BD

C     .. Scalar Arguments ..
      INTEGER IDOMN,LAVAR,LGUARD,LIWMX,LRHS,LWMX,LX,MAXIND,MAXRVL,
     +        MAXVL,NELT,NFVAR,NGUARD,NRHS,NZ,TOTELT
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION AVAR(LAVAR,LAVAR),
     +                 RHSVAL(MAXRVL),RINFO(20),
     +                 VALUE(MAXVL),W(LIWMX),X(LX,LRHS)
      INTEGER ELTPTR(TOTELT+1),ELTVAR(NZ),IGUARD(LGUARD),
     +        INFO(20),ISAVE(50),IW(LIWMX),JCNTL(2),JNFO(2),
     +        LAST(MAXIND),NLIST(NELT),VALPTR(TOTELT)
C     ..
C     .. Local Scalars ..
      INTEGER I,IELT,J,JSTRT,K,KSTRT,LIW,LW,NDF,NFRONT,NVAR
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION CNTL(5)
      INTEGER ICNTL(15),ISTRM(2),LENBUF(2)
      CHARACTER FILNAM(2)*128
C     ..
C     .. External Subroutines ..
      EXTERNAL MA62AD,MA62BD,MA62ID,MA62JD,MA62PD,MA72BD
C     ..
      CALL MA62ID(ICNTL,CNTL,ISAVE)
C Loop over the elements in the subdomain calling MA62A/AD
      DO 10 I = 1,NELT
         IELT = NLIST(I)
         NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
         JSTRT = ELTPTR(IELT)
         CALL MA62AD(NVAR,ELTVAR(JSTRT),NDF,LAST,MAXIND,ICNTL,ISAVE,
     +               INFO)
         IF (INFO(1).LT.0) RETURN
   10 CONTINUE
C Call to MA62A/AD for guard element
      CALL MA62AD(NGUARD,IGUARD,NDF,LAST,MAXIND,ICNTL,ISAVE,INFO)
      IF (INFO(1).LT.0) RETURN

C Loop over the elements in the subdomain calling MA62J/JD
```

```
      DO 20 I = 1,NELT
         IELT = NLIST(I)
         NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
         JSTRT = ELTPTR(IELT)
         CALL MA62JD(NVAR,ELTVAR(JSTRT),NDF,LAST,ICNTL,ISAVE,INFO,RINFO)
         IF (INFO(1).LT.0) RETURN
   20 CONTINUE
C Call to MA62J/JD for guard element
      CALL MA62JD(NGUARD,IGUARD,NDF,LAST,ICNTL,ISAVE,INFO,RINFO)
      IF (INFO(1).LT.0) RETURN
C
C Prepare to call MA62P/PD to set up direct access files.
C Make sure different stream numbers are used on each subdomain.
      DO 30 I = 1,2
         ISTRM(I) = 10 + (IDOMN-1)*2 + I
         LENBUF(I) = 512
   30 CONTINUE
      CALL MA62PD(ISTRM,FILNAM,LENBUF,ICNTL,ISAVE,INFO)
      IF (INFO(1).LT.0) RETURN
C
C Prepare to call MA62B/BD
      NFRONT = INFO(6)
      LW = 3 + LENBUF(1) + NFRONT* (NFRONT+NRHS)
      LIW = LENBUF(2) + 3*NFRONT
      IF (LW.GT.LWMX) INFO(1) = -99
      IF (LIW.GT.LIWMX) INFO(1) = -98
      IF (INFO(1).LT.0) RETURN
      DO 60 I = 1,NELT
         IELT = NLIST(I)
         NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
         JSTRT = ELTPTR(IELT)
         KSTRT = VALPTR(IELT)
         DO 50 K = 1,NVAR
            DO 40 J = 1,K
               AVAR(J,K) = VALUE(KSTRT)
               KSTRT = KSTRT + 1
   40       CONTINUE
   50    CONTINUE
         CALL MA62BD(NVAR,ELTVAR(JSTRT),NDF,LAST,LAVAR,AVAR,NRHS,
     +               RHSVAL(JSTRT),LX,X,LENBUF,LW,W,LIW,IW,ICNTL,CNTL,
     +               ISAVE,INFO,RINFO)
         IF (INFO(1).LT.0) RETURN
   60 CONTINUE
C Call MA72B/BD to preserve partial factorization
      CALL MA72BD(NRHS,NDF,NFVAR,LAST,LX,X,LIW,IW,LW,W,ISAVE,JCNTL,JNFO)
      RETURN
      END

C*******************************************************************
      SUBROUTINE INTERF(JFILE,NDOMN,NFVAR,LFVAR,IFVAR,FVAR,FRHS,MAXIND,
     +                  LAST,LRHS,NRHS,LWMX,W,LIWMX,IW,LX,X,ICNTL,ISAVE,
     +                  INFO,RINFO)

C Subroutine to use MA62 o solve interface problem

C     .. Scalar Arguments ..
      INTEGER JFILE,LFVAR,LIWMX,LRHS,LWMX,LX,MAXIND,NDOMN,NFVAR,NRHS
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION FRHS(LFVAR,LRHS),FVAR(LFVAR,LFVAR),RINFO(20),
     +                 W(LWMX),X(LX,LRHS)
      INTEGER ICNTL(15),IFVAR(LFVAR),INFO(20),ISAVE(50),IW(LIWMX),
     +        LAST(MAXIND)
C     ..
C     .. Local Scalars ..
      INTEGER I,IDOMN,J,LIW,LW,NDF,NFRONT
C     ..
C     .. Local Arrays ..
```

```
      DOUBLE PRECISION CNTL(5)
      INTEGER ISTRM(2),LENBUF(2)
      CHARACTER FILNAM(2)*128
C     ..
C     .. External Subroutines ..
      EXTERNAL MA62AD,MA62BD,MA62ID,MA62JD,MA62PD
C     ..
      CALL MA62ID(ICNTL,CNTL,ISAVE)
      ICNTL(5) = 1
C Number of elements in interface problem is equal to
C NDOMN, the number of subdomains.
      DO 10 IDOMN = 1,NDOMN
         READ (JFILE) NFVAR, (IFVAR(I),I=1,NFVAR),
     +     ((FRHS(I,J),I=1,NFVAR),J=1,NRHS),
     +     ((FVAR(J,I),J=1,I),I=1,NFVAR)
         CALL MA62AD(NFVAR,IFVAR,NDF,LAST,MAXIND,ICNTL,ISAVE,INFO)
         IF (INFO(1).LT.0) RETURN
   10 CONTINUE
      REWIND (JFILE)
      DO 20 IDOMN = 1,NDOMN
         READ (JFILE) NFVAR, (IFVAR(I),I=1,NFVAR),
     +     ((FRHS(I,J),I=1,NFVAR),J=1,NRHS),
     +     ((FVAR(J,I),J=1,I),I=1,NFVAR)
         CALL MA62JD(NFVAR,IFVAR,NDF,LAST,ICNTL,ISAVE,INFO,RINFO)
         IF (INFO(1).LT.0) RETURN
   20 CONTINUE
      REWIND (JFILE)
C
C Prepare to call MA62P/PD
      DO 30 I = 1,2
         ISTRM(I) = 10 + NDOMN*2 + I
         LENBUF(I) = 512
   30 CONTINUE
      CALL MA62PD(ISTRM,FILNAM,LENBUF,ICNTL,ISAVE,INFO)
      IF (INFO(1).LT.0) RETURN
C
C Prepare to call MA62B/BD
      NFRONT = INFO(6)
      LW = 3 + LENBUF(1) + NFRONT* (NFRONT+NRHS)
      LIW = LENBUF(2) + 3*NFRONT
      IF (LW.GT.LWMX) INFO(1) = -99
      IF (LIW.GT.LIWMX) INFO(1) = -98
      IF (INFO(1).LT.0) RETURN
      DO 40 IDOMN = 1,NDOMN
         READ (JFILE) NFVAR, (IFVAR(I),I=1,NFVAR),
     +     ((FRHS(I,J),I=1,NFVAR),J=1,NRHS),
     +     ((FVAR(J,I),J=1,I),I=1,NFVAR)
         CALL MA62BD(NFVAR,IFVAR,NDF,LAST,LFVAR,FVAR,NRHS,FRHS,LX,X,
     +               LENBUF,LW,W,LIW,IW,ICNTL,CNTL,ISAVE,INFO,RINFO)
         IF (INFO(1).LT.0) RETURN
   40 CONTINUE
      RETURN
      END
```

The input data used for this problem is:

```
  6   4
  2   2
  1   2
  3   4
  1   3   5   9  13
  4   5   5   6   4   5   1   2   5   6   2   3
  1   4   7  17
 26
  2.   1.   7.   3.   2.   8.   4.   3.   1.   2.   3.   6.
  3.   2.   1.   5.   2.   1.   3.   8.   2.   2.   3.   2.
  5.   4.
 12
  3.   8.   5.  10.  12.   9.  12.  11.  14.   8.  17.  14.
  6.   1.   2.   7.   4.  -1.
```

This produces the following output:

```
The solution is:

 1.000  1.000  1.000  1.000  1.000  1.000

Now solving for a further right-hand side.

The solution is:

 1.000  1.000  0.000  1.000 -1.000  0.000
```