1 SUMMARY

This collection of subroutines, when used in conjunction with the MA42 package, **solves 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. This can be done in parallel. Using multiple fronts can also reduce the amount of work required.

At the end of the assembly and elimination processes for the subdomains, for each subdomain i there remains a frontal matrix $\mathbf{F}_i$ and a corresponding right-hand side vector $\mathbf{c}_i$. These may be assembled to give a system of the form

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

By treating each of the subdomain frontal matrices $\mathbf{F}_i$ as an elemental matrix, the *interface problem* (1) may be solved by a frontal method. Once (1) has been solved, back-substitution on the subdomains completes the solution.

MA52 provides routines for generating lists of variables belonging to more than one subdomain, for preserving the partial factorization of a matrix when the sequence of calls to MA42B/BD (using element or equation entry) is incomplete, and for performing forward or back-substitution on a subdomain.

MA52 uses reverse communication.

The use of HSL routine MC53 to obtain an efficient element ordering in each subdomain is recommended before MA52 is used.

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

MA52 was revised in October 1999 and in August 2001. This involved adding extra subroutines MA52F/FD and MA52K/KD, which allow MA52 to be used without the restriction to diagonal pivoting required by MA52B/BD and also allows MA52 to be used to solve general linear systems of equations (not in finite element form) using the multiple front method. In this case, the user must partition the system matrix into blocks of rows and the frontal method should then be applied to each block separately. At the end of the assembly and elimination processes, for each block there will remain a rectangular frontal matrix and corresponding right-hand side vector. These may be preserved using subroutine MA52F/FD or, in sparse form, using MA52K/KD.

ATTRIBUTES — Version: 1.0.1. (28th February 2008) **Types:** Real (single, double). **Calls:** MA42. **Helpful:** MC53. **Language:** Fortran 77. **Original date:** March 1994 (revised October 1999 and August 2001). **Origin:** J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Argument lists and calling sequences.

There are five entries:

(a) MA52A/AD generates *guard elements* that is, lists of variables belonging to more than one subdomain. One guard element is generated for each subdomain. If MA52A/AD is used, it must be called once for each element in the problem. The calls may be made in any order. The use of MA52A/AD is optional.

(b) MA52F/FD may be used to preserve the partial factorization when the sequence of calls to the factorization routine MA42B/BD from the MA42 frontal package is incomplete. The frontal matrix and corresponding frontal right-hand side vectors at the point at which the sequence of calls to MA42B/BD was terminated are returned to the user, together with the partial factorization and the row and column indices of the variables remaining in the

front. If direct-access files were used by MA42, the partial factorization is written to these files. The routine can only be used if no errors have been returned from MA42B/BD.

(c) MA52K/KD offers the same functionality as MA52F/FD except that the frontal matrix is stored as a sparse (rather than a dense) matrix.

(d) MA52B/BD may be used to preserve the partial factorization in the special case of element entry and diagonal pivoting being used by MA42B/BD. MA52B/BD has the same interface as MA52F/FD except that, because diagonal pivoting forces the row and column indices of the variables remaining in the front to be the same, only one set of indices is returned to the user.

(e) MA52C/CD performs forward or back-substitution on a subdomain. The routine can only be called after an earlier call to MA52F/FD or MA52B/BD for the subdomain.

It is envisaged that the user will first use MA52A/AD to generate guard elements. However, the user may find it more convenient to generate the guard elements by some other means. All that is required is, for each subdomain, the number of variables shared with other subdomains, together with a list of the global indices of these variables.

Having generated the guard elements, the user employs MA42 to obtain a partial factorization on each subdomain. This is done by calling MA42A/AD once for each of the genuine elements in the subdomain and then calling MA42A/AD for the guard element. The factorization subroutine MA42B/BD is then called for each of the genuine elements in the subdomain, but not the guard element. The factorization obtained in this way is incomplete. In the simplest implementation of the multiple front algorithm, the calls to MA42B/BD are made with diagonal pivoting switched on (|ICNTL(7)| = 1001) and the partial factorization is preserved by calling MA52B/BD. Otherwise, the partial factorization is preserved by calling MA52F/FD or MA52K/KD. The output from MA52B/BD and MA52F/FD is in the form of a dense frontal matrix and corresponding frontal right-hand side vector (or matrix for multiple right-hand sides); MA52K/KD returns the frontal matrix in sparse format. There is one frontal matrix $\mathbf{F}_i$ and one frontal right-hand side vector $\mathbf{c}_i$ for each subdomain. These may be assembled and solved using MA42.

Once (1) has been solved, the user needs to call MA52C/CD for each subdomain to perform the back-substitution needed to solve for the remaining variables.

MA52C/CD may also be used to solve for further right-hand sides. In this case, MA52C/CD must be used to perform forward substitution on each subdomain, then MA42C/CD may be employed to solve for the variables in the interface problem, and finally MA52C/CD must be recalled on each subdomain for the final back-substitution.

The calling sequence is illustrated in Section 5.

2.1.1 Generation of guard elements

A call of the following form must be made for each element in the problem. The elements may be presented in any order.

The single precision version

```
      CALL MA52A(ICALL,NVAR,IVAR,TOTELT,NDOMN,IDOMN,NGUARD,LGUARD,
     *           IGUARD,LIW,IW,LP,IFLAG)
```

The double precision version

```
      CALL MA52AD(ICALL,NVAR,IVAR,TOTELT,NDOMN,IDOMN,NGUARD,LGUARD,
     *           IGUARD,LIW,IW,LP,IFLAG)
```

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:** NVAR ≥ 1.

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 ≤ IVAR(I) ≤ LIW, I = 1, 2,..., NVAR.

TOTELT is an INTEGER variable that must be set by the user to the number of elements in the finite-element mesh. This argument must be unchanged between calls to MA52A/AD and is not altered by the routine. **Restriction:** TOTELT > 1.

NDOMN is an INTEGER variable that must be set by the user to the number of the subdomains which comprise the finite-element mesh. This argument must be unchanged between calls to MA52A/AD and is not altered by the routine. **Restriction:** 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 ≤ IDOMN ≤ NDOMN.

NGUARD is an INTEGER array of length NDOMN that need not be set by the user. On exit from the final call, NGUARD(IDOMN) holds the number of variables in the guard element for subdomain IDOMN (IDOMN = 1, 2,..., NDOMN). This argument must be unchanged between calls to MA52A/AD.

LGUARD is an INTEGER variable that must be set by the user to the first dimension of the array IGUARD. LGUARD must be at least as large as $\max_i$ NGUARD(i), but in practice the user needs to choose a value larger than this. This argument must be unchanged between calls to MA52A/AD and is not altered by the routine.

IGUARD is an INTEGER array of dimensions LGUARD, 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 MA52A/AD.

LIW is an INTEGER variable that must be set by the user to the length of the array IW. LIW must 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 MA52A/AD and is not altered by the routine.

IW is an INTEGER array of length LIW which is used by the routine as workspace. This argument must be unchanged between calls to MA52A/AD. If the user-supplied value of LGUARD is insufficient (IFLAG = -6) and the sequence of calls to MA52A/AD is completed, on exit from the final call IW(1) is set to a value which is sufficient for LGUARD.

LP is an INTEGER variable which must be set by the user to the stream number for the printing of error messages. Printing is suppressed if LP ≤ 0. This argument is not altered by the routine.

IFLAG is an INTEGER variable that need not be set by the user. On successful exit, IFLAG is set to 0. Negative values of IFLAG indicate an error. IFLAG = 1 is associated with a non-terminal error. For nonzero values of IFLAG, see section 2.2.

2.1.2 Preservation of the partial factorization (diagonal pivoting)

MA52B/BD or MA52F/FD may be called to preserve the partial factorization of a matrix when the sequence of calls to MA42B/BD has not been completed. MA52B/BD may be called provided element entry with diagonal pivoting is being used (that is, element entry to MA42B/BD was used with the control parameter ICNTL(7) set to ±1001). Otherwise, MA52F/FD should be used.

The single precision version

```
      CALL MA52B(LRHS,NRHS,NDF,LAST,LX,X,LW,W,LIW,IW,NFVAR,IFVAR,FVAR,
     *           FRHS,LFVAR,ISAVE,LP,IFLAG)
```

The double precision version

```
      CALL MA52BD(LRHS,NRHS,NDF,LAST,LX,X,LW,W,LIW,IW,NFVAR,IFVAR,FVAR,
     *            FRHS,LFVAR,ISAVE,LP,IFLAG)
```

LRHS,NRHS,NDF are INTEGER variables which must be unchanged since the last call made by the user to MA42B/BD.

These arguments are not altered by the routine.

LAST is an INTEGER array of length NDF which must be unchanged since the last call made by the user to MA42B/BD. On exit, LAST is restored to the values it contained on exit from the final call to MA42A/AD.

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

X is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX by LRHS that need not be set by the user. On exit, X(K,J) is set to zero (K = 1, 2,..., NDF, J = 1, 2,..., NRHS).

LW is an INTEGER variable which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

W is a REAL (DOUBLE PRECISION in the D version) array of length LW which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

LIW is an INTEGER variable which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

IW is an INTEGER array of length LIW which must be unchanged since the last call made by the user to MA42B/BD. This argument is 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 made by the user to MA42B/BD (NFVAR is equal to the row and column front sizes after the last call to MA42B/BD).

IFVAR is an INTEGER array of length LFVAR that need not be set by the user. On exit, the first NFVAR entries in IFVAR hold the indices of the variables remaining in the front after the last call made by the user to MA42B/BD.

FVAR is a REAL (DOUBLE PRECISION in the D version) array of dimensions LFVAR by LFVAR that need not be set by the user. On exit, FVAR(I,J) contains the value of the entry IFVAR(I), IFVAR(J) in the frontal matrix after the last call made by the user to MA42B/BD (I, J = 1, 2,..., NFVAR).

FRHS is a REAL (DOUBLE PRECISION in the D version) array of dimensions LFVAR by LRHS that need not be set by the user. On exit, FRHS(I,J) contains the value of the component IFVAR(I) in the J-th frontal right-hand side vector after the last call made by the user to MA42B/BD (I = 1, 2,..., NFVAR, J = 1, 2,..., NRHS).

LFVAR is an INTEGER variable that must be set by the user to the first dimension of the arrays IFVAR and FRHS, and to the first and second dimensions of the array FVAR. LFVAR must be at least as large as the front size at the point at which the sequence of calls to MA42B/BD was terminated (this front size is held in ISAVE(40) after the last call made by the user to MA42B/BD). This argument is not altered by the routine.

ISAVE is an INTEGER array of length 45 which must be unchanged since the last call made by the user to MA42B/BD. This argument is changed by the routine.

LP is an INTEGER variable that must be set by the user to the stream number for the printing of error messages. Printing is suppressed if LP ≤ 0. LP should not be equal to the stream number of any of the direct-access files used by MA42B/BD (but no check is made for this). This argument is not altered by the routine.

IFLAG is an INTEGER variable that need not be set by the user. On successful exit, IFLAG is set to 0. Negative values of IFLAG indicate an error (see Section 2.2).

2.1.3 Preservation of the partial factorization (general case)

The single precision version

```
      CALL MA52F(LRHS,NRHS,NDF,LAST,LX,X,LW,W,LIW,IW,NFVAR,IFVAR,JFVAR,
     *           FVAR,FRHS,LFVAR,ISAVE,LP,IFLAG)
```

The double precision version

```
CALL MA52FD(LRHS,NRHS,NDF,LAST,LX,X,LW,W,LIW,IW,NFVAR,IFVAR,JFVAR,
*           FVAR,FRHS,LFVAR,ISAVE,LP,IFLAG)
```

LRHS,NRHS,NDF are INTEGER variables which must be unchanged since the last call made by the user to MA42B/BD. These arguments are not altered by the routine.

LAST is an INTEGER array of length NDF which must be unchanged since the last call made by the user to MA42B/BD. On exit, LAST is restored to the values it contained on exit from the final call to MA42A/AD.

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

X is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX by LRHS that need not be set by the user. On exit, X(K,J) is set to zero (K = 1, 2,..., NDF, J = 1, 2,..., NRHS).

LW is an INTEGER variable which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

W is a REAL (DOUBLE PRECISION in the D version) array of length LW which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

LIW is an INTEGER variable which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

IW is an INTEGER array of length LIW which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

NFVAR is an INTEGER array of length 2 that need not be set by the user. On exit, NFVAR(1) and NFVAR(2) hold, respectively, the number of rows and columns remaining in the front after the last call made by the user to MA42B/BD. Note that for element entry, NFVAR(1) = NFVAR(2).

IFVAR is an INTEGER array of length LFVAR(1) that need not be set by the user. On exit from MA52B/BD, the first NFVAR(1) entries in IFVAR hold the row indices of the variables remaining in the front after the last call to MA42B/BD.

JFVAR is an INTEGER array of length LFVAR(2) that need not be set by the user. On exit from MA52F/FD, the first NFVAR(2) entries in JFVAR hold the column indices of the variables remaining in the front after the last call made by the user to MA42B/BD.

FVAR is a REAL (DOUBLE PRECISION in the D version) array of dimensions LFVAR(1) by LFVAR(2) that need not be set by the user. On exit, FVAR(I,J) contains the value of the entry IFVAR(I), JFVAR(J) in the frontal matrix after the last call made by the user to MA42B/BD (I = 1, 2,..., NFVAR(1), J = 1, 2,..., NFVAR(2)).

FRHS is a REAL (DOUBLE PRECISION in the D version) array of dimensions LFVAR(1) by LRHS that need not be set by the user. On exit, FRHS(I,J) contains the value of the component IFVAR(I) in the J-th frontal right-hand side vector after the last call made by the user to MA42B/BD (I = 1, 2,..., NFVAR(1), J = 1, 2,..., NRHS).

LFVAR is an INTEGER array of length 2. LFVAR(1) must be set by the user to the dimension of IFVAR and to the first dimension of the arrays FVAR and FRHS. LFVAR(2) must be set to the dimension of JFVAR and to the second dimension of the array FVAR. LFVAR(1) and LFVAR(2) must be at least as large as the row and column front sizes, respectively, at the point at which the sequence of calls to MA42B/BD was terminated (these front sizes are held in ISAVE(39) and ISAVE(40) after the last call made by the user to MA42B/BD). This argument is not altered by the routine.

ISAVE is an INTEGER array of length 45 which must be unchanged since the last call made by the user to MA42B/BD. This argument is changed by the routine.

LP is an INTEGER variable that must be set by the user to the stream number for the printing of error messages. Printing is suppressed if LP ≤ 0. LP should not be equal to the stream number of any of the direct-access files

used by MA42B/BD (but no check is made for this). This argument is not altered by the routine.

IFLAG is an INTEGER variable that need not be set by the user. On successful exit, IFLAG is set to 0. Negative values of IFLAG indicate an error (see Section 2.2).

2.1.4 Preservation of the partial factorization (sparse case)

MA52K/KD should be used if, at the point at which the series of calls to MA42B/BD is terminated, the frontal matrix contains a large number of zero entries. MA52K/KD stores the remaining frontal matrix as a sparse matrix using column indices (JFVAR) and row pointers (IP).

The single precision version

```
      CALL MA52K(LRHS,NRHS,NDF,LAST,LX,X,LW,W,LIW,IW,NFVAR,IFVAR,JFVAR,
     *           IP,FVAR,FRHS,LFVAR,ISAVE,LP,IFLAG)
```

The double precision version

```
      CALL MA52KD(LRHS,NRHS,NDF,LAST,LX,X,LW,W,LIW,IW,NFVAR,IFVAR,JFVAR,
     *            IP,FVAR,FRHS,LFVAR,ISAVE,LP,IFLAG)
```

LRHS,NRHS,NDF are INTEGER variables which must be unchanged since the last call made by the user to MA42B/BD. These arguments are not altered by the routine.

LAST is an INTEGER array of length NDF which must be unchanged since the last call made by the user to MA42B/BD. On exit, LAST is restored to the values it contained on exit from the final call to MA42A/AD.

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

X is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX by LRHS that need not be set by the user. On exit, X(K,J) is set to zero (K = 1, 2,..., NDF, J = 1, 2,..., NRHS).

LW is an INTEGER variable which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

W is a REAL (DOUBLE PRECISION in the D version) array of length LW which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

LIW is an INTEGER variable which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

IW is an INTEGER array of length LIW which must be unchanged since the last call made by the user to MA42B/BD. This argument is not altered by the routine.

NFVAR is an INTEGER array of length 2 that need not be set by the user. On exit, NFVAR(1) holds the number of rows remaining in the front after the last call made by the user to MA42B/BD and NFVAR(2) holds the number of nonzero entries in the frontal matrix.

IFVAR is an INTEGER array of length LFVAR(1) that need not be set by the user. On exit from MA52B/BD, the first NFVAR(1) entries in IFVAR hold the row indices of the variables remaining in the front after the last call to MA42B/BD.

JFVAR is an INTEGER array of length LFVAR(2) that need not be set by the user. On exit from MA52F/FD, the first NFVAR(2) entries in JFVAR hold the column indices of the variables remaining in the front after the last call made by the user to MA42B/BD. The entries of a single row are contigous, with the column indices of the entries in the first remaining row preceding those in the second row, and so on.

IP is an INTEGER array of length LFVAR(1)+1. On exit, IP(I) holds the position in the array JFVAR of the first entry in row I, I = 1, 2,..., NFVAR(1), and IP(NFVAR(1)+1) is set to one greater than the number of nonzero entries in the frontal matrix.

FVAR is a REAL (DOUBLE PRECISION in the D version) array of length LFVAR(2) that need not be set by the user. On exit, FVAR(J) contains the value of the entry in the frontal matrix whose column index is in JFVAR(J) (J = 1, 2,..., NFVAR(2)).

FRHS is a REAL (DOUBLE PRECISION in the D version) array of dimensions LFVAR(1) by LRHS that need not be set by the user. On exit, FRHS(I,J) contains the value of the component IFVAR(I) in the J-th frontal right-hand side vector after the last call made by the user to MA42B/BD (I = 1, 2,..., NFVAR(1), J = 1, 2,..., NRHS).

LFVAR is an INTEGER array of length 2. LFVAR(1) must be set by the user to the dimension of IFVAR and LFVAR(2) must be set to the dimension of JFVAR and FVAR. LFVAR(1) must be at least as large as the row front size at the point at which the sequence of calls to MA42B/BD was terminated (this is held in ISAVE(39) after the last call made by the user to MA42B/BD). LFVAR(2) must be at least as large as the number of nonzero entries remaining in the front (note that this can never exceed ISAVE(39)*ISAVE(40) and may be much less). This argument is not altered by the routine.

ISAVE is an INTEGER array of length 45 which must be unchanged since the last call made by the user to MA42B/BD. This argument is changed by the routine.

LP is an INTEGER variable that must be set by the user to the stream number for the printing of error messages. Printing is suppressed if LP $\leq$ 0. LP should not be equal to the stream number of any of the direct-access files used by MA42B/BD (but no check is made for this). This argument is not altered by the routine.

IFLAG is an INTEGER variable that need not be set by the user. On successful exit, IFLAG is set to 0. Negative values of IFLAG indicate an error (see Section 2.2).

2.1.5 Forward and back-substitution on a subdomain

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

The single precision version

```
CALL MA52C(IND,NRHS,LX,B,X,LW,W,LIW,IW,ISAVE,LP,IFLAG)
```

The double precision version

```
CALL MA52CD(IND,NRHS,LX,B,X,LW,W,LIW,IW,ISAVE,LP,IFLAG)
```

IND is an INTEGER variable that must be set by the user. If the user wishes to perform back-substitution following the final call to MA42B/BD for the interface problem, IND should be set to 1. If the user wishes to perform forward substitution, IND should be set to 2. If the user wishes to perform back-substitution following a call to MA42C/CD for the interface problem, IND should be set to 3. **Restriction:** 1 $\leq$ IND $\leq$ 3.

NRHS is an INTEGER variable. If IND = 1, NRHS must be unchanged since the calls made by the user to MA42B/BD. Otherwise, NRHS must be set by the user to the number of right-hand sides. **Restriction:** NRHS $>$ 0. This argument is not altered by the routine.

LX is an INTEGER variable that must be set by the user to the first dimension of the arrays B and X. LX must be at least as large as the largest integer used to index a variable in the whole domain. (The largest integer used to index a variable in the subdomain is held in ISAVE(30) on exit from the final call to MA42A/AD for the subdomain). This argument is not altered by the routine.

B is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX, NRHS. If IND =1, B is not accessed. If IND = 2, on entry B must be set by the user so that if K has been used to index a variable in the subdomain, B(K,J) is the corresponding component of the right-hand side for the J-th system (J=1,2,..., NRHS). In this case, B is altered by the routine. If IND = 3, B must be unchanged since the call to MA42C/CD for the interface problem and is unchanged by the routine.

X is a REAL (DOUBLE PRECISION in the D version) array of dimensions LX, NRHS. On entry, if IND =1 (respectively, 3), X must be unchanged since the final call to MA42B/BD (respectively, MA42C/CD) for the interface problem. On exit, if K is used to index a variable in the subdomain, X(K,J) holds the solution for

variable K to system J. If IND = 2, X is not accessed.

LW is an INTEGER variable that must be set by the user to the dimension of the array W. LW must satisfy the following conditions:
If direct-access files are being used,
LW ≥ ISAVE(7) + ISAVE(24)*(ISAVE(22) + ISAVE(23)) + NRHS* ISAVE(24) (IND = 1 or 3)
LW ≥ ISAVE(8) + ISAVE(23)*ISAVE(24) + NRHS* ISAVE(23) (IND = 2).
Otherwise,
LW ≥ ISAVE(7) + ISAVE(8) + NRHS* ISAVE(24) (IND = 1 or 3)
LW ≥ ISAVE(7) + ISAVE(8) + NRHS* ISAVE(23) (IND = 2).

W is a REAL (DOUBLE PRECISION in the D version) array of length LW. If direct-access files are not being used by MA42, the first ISAVE(7)+ISAVE(8) entries of W must be unchanged since the call to MA52B/BD and these entries are unchanged by MA52C/CD. Otherwise, W is used by MA52C/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 are not being used by MA42, LIW must be at least ISAVE(9). Otherwise, LIW must be at least 5 + ISAVE(9) + ISAVE(23) + ISAVE(24). This argument is not altered by the routine.

IW is an INTEGER array of length LW. If direct-access files are not being used by MA42, the first ISAVE(9) entries of IW must be unchanged since the call to MA52B/BD and these entries are unchanged by MA52B/BD. Otherwise, IW is used by MA52C/CD as workspace.

ISAVE is an INTEGER array of dimension 45 which must be unchanged since the final call to MA52B/BD on the subdomain. This argument is not altered by the routine.

LP is an INTEGER variable that must be set by the user to the stream number for the printing of error messages. Printing is suppressed if LP ≤ 0. LP should not be equal to any of the stream numbers used for the direct-access files associated with the subdomains and with the interface problem (but no check is made for this). This argument is not altered by the routine.

IFLAG is an INTEGER variable that need not be set by the user. On successful exit, IFLAG is set to 0. Negative values of IFLAG indicate an error (see Section 2.2.3).

2.2 Error diagnostics

A negative value for IFLAG is associated with a fatal error and a positive value is associated with a non-terminal error. If LP is greater than 0, a self-explanatory message is, in each case, output on unit LP.

2.2.1 Error diagnostics for MA52A/AD

The possible non-zero values for IFLAG returned by MA52A/AD are:

+1 The defined first dimension LGUARD of the array IGUARD is insufficient. If the user continues to call MA52A/AD, on exit from the final call a value which is sufficient will be returned (see error –6).

–1 Value of TOTELT out of range. Immediate return with input parameters unchanged. Note that TOTELT is only checked on the first entry to MA52A/AD.

–2 Value of NDOMN out of range. Immediate return with input parameters unchanged. Note that NDOMN is only checked on the first entry to MA52A/AD.

–3 Value of NVAR out of range. Immediate return with input parameters unchanged. Note that NVAR is only checked on the first entry to MA52A/AD.

–4 A variable index in the current element is out of range. Immediate return with input parameters unchanged.

–5 Value of IDOMN out of range. Immediate return with input parameters unchanged.

–6 Defined first dimension LGUARD of the array IGUARD is insufficient. However, the sequence of calls to

MA52A/AD has been completed and a value which is sufficient is given in the error message and is returned in IW(1).

2.2.2 Error diagnostics for MA52B/BD, MA52F/FD, and MA52K/KD

The possible non-zero values for IFLAG returned by MA52B/BD, MA52F/FD, and MA52K/KD are:

–1 MA52B/BD (or MA52F/FD or MA52K/KD) has been called after an error was issued by MA42B/BD. Immediate return with input parameters unchanged.

–2 The sequence of calls to MA42B/BD was completed before MA52B/AD (or MA52F/FD or MA52K/KD) was called. Immediate return with input parameters unchanged.

–3 LFVAR is too small. The error message advises the user on suitable LFVAR which will ensure success on a subsequent run with identical data. For MA52B/BD, suitable LFVAR is also held in ISAVE(40) and, for MA52F/FD, in ISAVE(39) and ISAVE(40). For MA52K/KD, a suitable value for LFVAR(1) is held in ISAVE(39). For MA52B/BD and MA52F/FD, immediate return with input parameters unchanged.

–4 NRHS has been changed since the last call made by the user to MA42B/BD. Immediate return with input parameters unchanged.

–5 LX is out of range. Immediate return with input parameters unchanged.

–6 Insufficient space was allocated to one or more of the direct-access files to permit the partial factorization to be stored. The error message gives the space necessary for success on a subsequent run with identical data. For success on a subsequent run, when calling MA42P/PD to set up the direct-access files for MA42, the user must set LENFLE(I) to be at least LENBUF(I) greater than the value which led to this error message (I = 1, 2, 3).

–7 Error detected when writing to a direct-access file. The iostat parameter is printed.

–8 Diagonal pivoting was not used in the calls to MA42B/BD. The value of the parameter ICNTL(7) used by MA42B/BD to control diagonal pivoting is printed. This error can only be returned by MA52B/BD.

2.2.3 Error diagnostics for MA52C/CD

The possible non-zero values for IFLAG returned by MA52C/CD are:

–1 The call to MA52C/CD does not follow an earlier call to MA52B/BD. Immediate return with input parameters unchanged.

–2 First dimension LX of the array X too small. The error message advises the user on a value of LX which will ensure success on a subsequent run with identical data. This value is also in ISAVE(30). Immediate return with input parameters unchanged.

–3 Defined length LW of the array W violates the restrictions on LW. Immediate return with input parameters unchanged.

–4 Defined length LIW of the array IW violates the restrictions on LIW. Immediate return with input parameters unchanged.

–5 Either NRHS $\leq$ 0 or NRHS has been changed since the call made by the user to MA52B/BD (or MA52F/FD). Immediate return with input parameters unchanged.

–6 Error detected when reading from a direct-access file. The iostat parameter is printed.

–7 IND = 2 or 3 but there is no record of the L-factor being stored during factorization. Immediate return with input parameters unchanged.

–8 Value of IND out of range. Immediate return with input parameters unchanged.

3 GENERAL INFORMATION

3.1 Summary of information.

Other routines called directly: MA52A/AD calls no other routines. MA52B/BD calls the internal subroutines MA52D/DD and MA52E/ED, MA52F/FD calls MA52E/ED and MA52G/GD, MA52K/KD calls MA52E/ED and MA52L/LD, and all three call the internal subroutine MA42L/LD from the MA42 package. MA52C/CD calls the internal subroutines MA42D/DD, MA42E/ED, and MA42L/LD from the MA42 package.

Input/output: In the event of errors, diagnostic messages are printed on unit LP.

Restrictions:

Restrictions for MA52A/AD**:**
TOTELT ≥ 2.
NDOMN ≥ 2.
NVAR ≥ 1.
1 ≤ IVAR(I) ≤ LIW, I = 1, 2,..., NVAR.
1 ≤ IDOMN ≤ NDOMN.

Restrictions for MA52B/BD**,** MA52F/FD**, and** MA52K/KD**:** LX ≥ NDF and, in addition, the routines require that the arguments LRHS, NRHS, NDF, LAST, LW, W, LIW, IW, and ISAVE are unchanged since the last call made by the user to MA42B/BD. The routines check that NRHS is unchanged.

Restrictions for MA52C/CD**:**
NRHS ≥ 1.
1 ≤ IND ≤ 3.
If direct-access files are being used,
LW ≥ ISAVE(7) + ISAVE(24)*(ISAVE(22) + ISAVE(23)) + NRHS* ISAVE(24) (IND = 1 or 3)
LW ≥ ISAVE(8) + ISAVE(23)*ISAVE(24) + NRHS* ISAVE(23) (IND = 2).
LIW ≥ ISAVE(9) + ISAVE(23) + ISAVE(24) + 5.
Otherwise,
LW ≥ ISAVE(7) + ISAVE(8) + NRHS* ISAVE(24) (IND = 1 or 3)
LW ≥ ISAVE(7) + ISAVE(8) + NRHS* ISAVE(23) (IND = 2).
LIW ≥ ISAVE(9).

4 METHOD

MA52A/AD

On the initial call to MA52A/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 entered 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.

MA52B/BD, MA52F/FD and MA52K/KD

MA52B/BD (and MA52F/FD and MA52K/KD) first checks that the sequence of calls to MA42B/BD has not been completed and that an error has not been issued by MA42B/BD. This is done using information held in the array ISAVE. ISAVE is also used by MA52B/BD to check that diagonal pivoting was used in the calls to MA42B/BD. 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 attempts to write out the contents of the buffers to the direct-access files. A check is

made that sufficient space remains in the direct-access files to do this and, if not, control is returned to the user with an error message advising the user on the increase in space necessary for a successful subsequent run on identical data. If there is insufficient space in one or more of the direct-access files the user will have to recall MA42P/PD with the revised file sizes and repeat the calls to MA42B/BD before MA52B/BD may be recalled.

The contents of the frontal matrix and the corresponding frontal right-hand side vectors on exit from the last call made by the user to MA42B/BD are copied from the array W into the arrays FVAR and FRHS, and the integer information for the variables in the front is copied from the array IW into the arrays IFVAR and JFVAR (MA52F/FD and MA52K/KD only). MA52K/KD stores only the nonzeros remaining in the front and, using column indices and row pointers, stores the frontal matrix in sparse format.

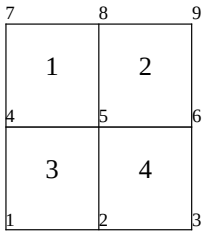
Finally, the array LAST is set to hold the values it would have contained if the sequence of calls to MA42B/BD had been completed and the array X is initialised to zero.

MA52C/CD

A check is first made that MA52B/BD or MA52F/FD 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 IND. If IND = 1 or 3, the internal subroutine from the MA42 package MA42D/DD is called to perform back-substitution, and if IND = 2, MA42E/ED is called to perform forward substitution.

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, $1 \le i \le 6$ (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 elemental matrices $\mathbf{A}^{(k)}$ $(1 \le k \le 4)$ are

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

where the variable indices are indicated by the integers before each matrix (columns are identified symmetrically to rows). The corresponding elemental 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.

```
C
```

```
C Code to run MA42 on a finite element mesh composed of
C two subdomains.
C
C     .. Parameters ..
      DOUBLE PRECISION ZERO
      PARAMETER (ZERO=0.0D0)
      INTEGER NDFMAX,LRHS,LWMAX,LIWMAX,NMAXE,LFVAR,LGUARD,NDOMN,NFMAX,
     +        MAXVL,NSUB,MELT,NZMAX,MAXRVL
      PARAMETER (NDFMAX=9,LRHS=1,LWMAX=1200,LIWMAX=1200,NMAXE=4,
     +          LFVAR=9,LGUARD=3,NDOMN=2,NFMAX=6,MAXVL=30,
     +          NSUB=4,MELT=4,NZMAX=30,MAXRVL=15)
C     ..
C     .. Local Scalars ..
      INTEGER I,ICALL,IDOMN,IELT,IFLAG,J,JFILE,JNDF,JSTRT,K,KSTRT,LIW,
     +        LP,LW,LX,NDFTOT,NFVAR,NRHS,NVAR,NZ,RHSCRD,TOTELT,VALCRD
C     ..
C     .. Local Arrays ..
      DOUBLE PRECISION AVAR(NMAXE,NMAXE),B(NDFMAX,LRHS),CNTL(2),
     +                 FRHS(LFVAR,LRHS),FVAR(LFVAR,LFVAR),
     +                 RHS(NMAXE,LRHS),RHSVAL(MAXRVL),RINFO(2,NDOMN),
     +                 RJNFO(2),VALUE(MAXVL),W(LWMAX),X(NDFMAX,LRHS)
      INTEGER ELTPTR(MELT+1),ELTVAR(NZMAX),ICNTL(8),IFVAR(LFVAR),
     +        IGUARD(LGUARD,NDOMN),INFO(23,NDOMN),ISAVE(45,NDOMN),
     +        ISTRM(3),IVAR(NMAXE),IW(LIWMAX),JCNTL(8),JNFO(23),
     +        JSAVE(45),KAST(NDFMAX),LAST(NDFMAX,NDOMN),LENBUF(3),
     +        LENFLE(3),NDF(NDOMN),NELT(NDOMN),NFRONT(2),NGUARD(NDOMN),
     +        NLIST(NSUB,NDOMN),VALPTR(MELT)
C     ..
C     .. External Subroutines ..
      EXTERNAL MA42AD,MA42BD,MA42CD,MA42ID,MA42PD,MA52AD,MA52BD,
     +         MA52CD
C     ..
C Stream for error messages.
C     .. Intrinsic Functions ..
      INTRINSIC MAX
C     ..
      LP = 6
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
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.
C
      READ (*,FMT=*) (ELTPTR(I),I=1,TOTELT+1)
      NZ = ELTPTR(TOTELT+1) - 1
      READ (*,FMT=*) (ELTVAR(I),I=1,NZ)
C
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 stored. VALPTR(I) points to the position in
C VALUE of the first value for element I.
C
         READ (*,FMT=*) (VALPTR(I),I=1,TOTELT)
         READ (*,FMT=*) VALCRD
         READ (*,FMT=*) (VALUE(I),I=1,VALCRD)
C
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.
C
         READ (*,FMT=*) RHSCRD
         READ (*,FMT=*) (RHSVAL(I),I=1,RHSCRD)
C
C Input complete. Generate the guard elements using MA52A/AD.
       ICALL = 0
       DO 50 IDOMN = 1,NDOMN
          DO 40 I = 1,NELT(IDOMN)
             IELT = NLIST(I,IDOMN)
             NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
             JSTRT = ELTPTR(IELT)
             DO 30 J = 1,NVAR
                IVAR(J) = ELTVAR(JSTRT+J-1)
   30        CONTINUE
             ICALL = ICALL + 1
             CALL MA52AD(ICALL,NVAR,IVAR,TOTELT,NDOMN,IDOMN,NGUARD,
     +                   LGUARD,IGUARD,LIWMAX,IW,LP,IFLAG)
             IF (IFLAG.LT.0) GO TO 210
   40    CONTINUE
   50 CONTINUE
C
C Now run MA42 on the subdomains.
       DO 120 IDOMN = 1,NDOMN
          CALL MA42ID(ICNTL,CNTL,ISAVE(1,IDOMN))
C Force diagonal pivoting
          ICNTL(7) = 1001
C Loop over the elements in the subdomain calling MA42A/AD
          DO 70 I = 1,NELT(IDOMN)
             IELT = NLIST(I,IDOMN)
             NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
             JSTRT = ELTPTR(IELT)
             DO 60 J = 1,NVAR
                IVAR(J) = ELTVAR(JSTRT+J-1)
   60        CONTINUE
             CALL MA42AD(NVAR,IVAR,NDF(IDOMN),LAST(1,IDOMN),NDFMAX,
     +                   ICNTL,ISAVE(1,IDOMN),INFO(1,IDOMN))
             IF (INFO(1,IDOMN).LT.0) GO TO 210
   70    CONTINUE
C Call to MA42A/AD for guard element
          CALL MA42AD(NGUARD(IDOMN),IGUARD(1,IDOMN),NDF(IDOMN),
     +                LAST(1,IDOMN),NDFMAX,ICNTL,ISAVE(1,IDOMN),
     +                INFO(1,IDOMN))
          IF (INFO(1,IDOMN).LT.0) GO TO 210
C
C Prepare to call MA42P/PD.
          DO 80 I = 1,3
             ISTRM(I) = 10 + (IDOMN-1)*3 + I
             LENBUF(I) = 512
             LENFLE(I) = 2048
   80    CONTINUE
          CALL MA42PD(ISTRM,LENBUF,LENFLE,ICNTL,ISAVE(1,IDOMN),
```

```
     +               INFO(1,IDOMN))
         IF (INFO(1,IDOMN).LT.0) GO TO 210
C
C Prepare to call MA42B/BD
         NRHS = 1
         NFRONT(1) = NFMAX
         NFRONT(2) = NFMAX
         LX = NDFMAX
         LW = 1 + LENBUF(1) + LENBUF(2) + NFRONT(1)*NFRONT(2) +
     +        MAX(NFRONT(1)*LRHS,NRHS*NFRONT(2))
         LIW = LENBUF(3) + 2*NFRONT(1) + 4*NFRONT(2)
         IF (LW.GT.LWMAX) GO TO 190
         IF (LIW.GT.LIWMAX) GO TO 200
         DO 110 I = 1,NELT(IDOMN)
            IELT = NLIST(I,IDOMN)
            NVAR = ELTPTR(IELT+1) - ELTPTR(IELT)
            JSTRT = ELTPTR(IELT)
            KSTRT = VALPTR(IELT)
            DO 100 J = 1,NVAR
               IVAR(J) = ELTVAR(JSTRT)
               DO 90 K = J,NVAR
                  AVAR(K,J) = VALUE(KSTRT)
                  IF (J.NE.K) AVAR(J,K) = AVAR(K,J)
                  KSTRT = KSTRT + 1
   90          CONTINUE
               RHS(J,1) = RHSVAL(JSTRT)
               JSTRT = JSTRT + 1
  100       CONTINUE
            CALL MA42BD(NVAR,IVAR,NDF(IDOMN),LAST(1,IDOMN),NMAXE,
     +                  AVAR,NRHS,RHS,LRHS,LX,X,NFRONT,LENBUF,LW,W,
     +                  LIW,IW,ICNTL,CNTL,ISAVE(1,IDOMN),
     +                  INFO(1,IDOMN),RINFO(1,IDOMN))
            IF (INFO(1,IDOMN).LT.0) GO TO 210
  110    CONTINUE
C
C Call MA52B/BD
         CALL MA52BD(LRHS,NRHS,NDF(IDOMN),LAST(1,IDOMN),LX,X,LW,W,LIW,
     +               IW,NFVAR,IFVAR,FVAR,FRHS,LFVAR,ISAVE(1,IDOMN),LP,
     +               IFLAG)
         IF (IFLAG.LT.0) GO TO 210
C Write the interface element matrix and element right hand side
C vector out to a file.
         JFILE = 8
         WRITE (JFILE) NFVAR, (IFVAR(I),I=1,NFVAR),
     +     ((FRHS(I,J),I=1,NFVAR),J=1,NRHS),
     +     ((FVAR(I,J),I=1,NFVAR),J=1,NFVAR)
C
C Finished with subdomain
  120 CONTINUE
      REWIND (JFILE)
C
C All subdomains dealt with. Now solve interface problem.
      CALL MA42ID(JCNTL,CNTL,JSAVE)
C Number of subdomains is number of elements in interface problem.
      DO 130 IDOMN = 1,NDOMN
         READ (JFILE) NFVAR, (IFVAR(I),I=1,NFVAR),
     +     ((FRHS(I,J),I=1,NFVAR),J=1,NRHS),
     +     ((FVAR(I,J),I=1,NFVAR),J=1,NFVAR)
         CALL MA42AD(NFVAR,IFVAR,JNDF,KAST,NDFMAX,JCNTL,JSAVE,JNFO)
         IF (JNFO(1).LT.0) GO TO 210
  130 CONTINUE
```

```
      REWIND (JFILE)
C
C Prepare to call MA42P/PD
      DO 140 I = 1,3
         ISTRM(I) = 10 + NDOMN*3 + I
         LENBUF(I) = 512
         LENFLE(I) = 2048
  140 CONTINUE
      CALL MA42PD(ISTRM,LENBUF,LENFLE,JCNTL,JSAVE,JNFO)
      IF (JNFO(1).LT.0) GO TO 210
C
C Prepare to call MA42B/BD
      NFRONT(1) = NFMAX
      NFRONT(2) = NFMAX
      LW = 1 + LENBUF(1) + LENBUF(2) + NFRONT(1)*NFRONT(2) +
     +     MAX(NFRONT(1)*LRHS,NRHS*NFRONT(2))
      LIW = LENBUF(3) + 2*NFRONT(1) + 4*NFRONT(2)
      IF (LW.GT.LWMAX) GO TO 190
      IF (LIW.GT.LIWMAX) GO TO 200
      DO 150 IDOMN = 1,NDOMN
         READ (JFILE) NFVAR, (IFVAR(I),I=1,NFVAR),
     +     ((FRHS(I,J),I=1,NFVAR),J=1,NRHS),
     +     ((FVAR(I,J),I=1,NFVAR),J=1,NFVAR)
         CALL MA42BD(NFVAR,IFVAR,JNDF,KAST,LFVAR,FVAR,NRHS,FRHS,LRHS,LX,
     +               X,NFRONT,LENBUF,LW,W,LIW,IW,JCNTL,CNTL,JSAVE,JNFO,
     +               RJNFO)
         IF (JNFO(1).LT.0) GO TO 210
  150 CONTINUE
      REWIND (JFILE)
C
C Now backsubstitute on each subdomain
      LW = LWMAX
      LIW = LIWMAX
      DO 160 IDOMN = 1,NDOMN
         CALL MA52CD(1,NRHS,LX,X,X,LW,W,LIW,IW,ISAVE(1,IDOMN),LP,IFLAG)
         IF (IFLAG.LT.0) GO TO 210
  160 CONTINUE
C
C  Solution is in first NDFTOT locations of X
      WRITE (*,FMT=9000)
      WRITE (*,FMT=9010) (X(I,1),I=1,NDFTOT)
C
C Now solve for a further right-hand side.
      WRITE (*,FMT=9060)
C Read in (assembled) right-hand side
      READ (*,FMT=*) (B(I,1),I=1,NDFTOT)
C
C Forward substitution on subdomains
      DO 170 IDOMN = 1,NDOMN
         CALL MA52CD(2,NRHS,LX,B,X,LW,W,LIW,IW,ISAVE(1,IDOMN),LP,IFLAG)
         IF (IFLAG.LT.0) GO TO 210
  170 CONTINUE
C
C Call MA42C/CD for interface problem
      CALL MA42CD(.FALSE.,NRHS,LX,B,X,LW,W,LIW,IW,JCNTL,JSAVE,JNFO)
      IF (JNFO(1).LT.0) GO TO 210
C
C Back substitution on subdomains
      DO 180 IDOMN = 1,NDOMN
         CALL MA52CD(3,NRHS,LX,B,X,LW,W,LIW,IW,ISAVE(1,IDOMN),LP,IFLAG)
         IF (IFLAG.LT.0) GO TO 210
```

```
  180 CONTINUE
      WRITE (*,FMT=9010) (X(I,1),I=1,NDFTOT)
C
      GO TO 210
  190 WRITE (*,FMT=9040) LW,LWMAX
      GO TO 210
  200 WRITE (*,FMT=9050) LIW,LIWMAX
  210 CONTINUE
      STOP
 9000 FORMAT (/' The solution is:')
 9010 FORMAT (/6F12.4)
 9040 FORMAT (' LW =',I8,' but LWMAX =',I8)
 9050 FORMAT (' LIW =',I8,' but LIWMAX =',I8)
 9060 FORMAT (' Now solving for a further right-hand side.')
      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.   2.   3.   1.   3.
   2.   6.   1.   5.   2.   1.   8.   3.   3.   2.   2.   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:

   1.0000      1.0000      1.0000      1.0000      1.0000      1.0000
Now solving for a further right-hand side.

   1.0000      1.0000      0.0000      1.0000     -1.0000      0.0000
```