Warning: Subroutine MA22 performs functions which are adequately treated by routines in other standard subroutine libraries (for example, LAPACK). The use of this routine is not recommended, and it may be removed from future releases of this library.

1 SUMMARY

Given a **symmetric positive definite** n by n matrix $\mathbf{A}=\{a_{ij}\}_{n\times n}$ this subroutine performs one or more of the following tasks.

(a) **Solves** the system of **linear algebraic equations**

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

given the right-hand sides b_i, i=1,2,...,n, and provides a re-entry facility for the rapid solution of further systems of equations which have the same elements a_{ij}.

(b) computes the **inverse matrix** $\mathbf{A}^{-1}$ of **A**.

(c) computes the value of the **determinant** of **A**.

In all cases the matrix **A** must be symmetric (a_{ij} = a_{ji}) and positive definite ($\mathbf{z}^{\mathrm{T}}\mathbf{Az}>0$ for any vector $\mathbf{z}\neq 0$).

The method is basically symmetric decomposition, implicit scaling and iterative refinement and applying small random perturbations in order to estimate errors, see S.Marlow and J.K.Reid, AERE R.6899.

ATTRIBUTES — Version: 1.0.0. **Types:** MA22A; MA22AD. **Calls:** FA01; FD05. **Original date:** August 1971. **Origin:** S.Marlow and J.K.Reid, Harwell.

2 HOW TO USE THE PACKAGE

2.1 Entries and argument lists

The entry points

To solve systems of linear equations as in § 1(a).

The single precision version

```
CALL MA22A(A,IA,N,B,W,E)
```

The double precision version

```
CALL MA22AD(A,IA,N,B,W,E)
```

To find inverse of matrix **A** in § 1(b).

The single precision version

```
CALL MA22B(A,IA,N,W,E)
```

The double precision version

```
CALL MA22BD(A,IA,N,W,E)
```

To find determinant of the matrix **A** as in § 1(c).

The single precision version

```
CALL MA22C(A,IA,N,DET,IDET,W)
```

The double precision version

```
CALL MA22CD(A,IA,N,DET,IDET,W)
```

Provided arguments `A, IA, N, W` have not been altered since a previous entry to `MA22A/AD` a new system with the same coefficients a_{ij} but different right-hand side elements b_i may be solved without repeating any of the work involved in the factorisation.

N.B. Iterative refinement `E>0` must not be requested unless the previous `MA22A/AD` entry also requested iterative refinement.

The single precision version

```
CALL MA22D(A,IA,N,B,W,E)
```

The double precision version

```
CALL MA22DD(A,IA,N,B,W,E)
```

The arguments

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) two-dimensional array which contains the coefficients a_{ij} on entry. Since **A** is symmetric only the lower triangle need be set, i.e. `A(i,j)` = a_{ij} , $1 \le i \le N$, $1 \le j \le i$. On an entry to `MA22A/AD`, or `MA22C/CD` the upper triangle i.e. `A(i,j)`, $1 \le i \le N$, $i < j \le N$ is overwritten ; the lower triangle is not overwritten. On an entry to `MA22B/BD` `A` will be overwritten by the inverse of the matrix **A**.

`IA` is an `INTEGER` variable set by the user to the first dimension of the array `A` . It is not altered by the subroutine.

`N` is an `INTEGER` variable set to the number of equations (i.e. the order of the matrix **A**). It is not altered by the subroutine.

`B` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least `N` which must be set to the numbers $b_1, b_2, ...b_n$ and will contain the solution $x_1, x_2, ...x_n$ on exit.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) workspace array of length at least `5*N`. It is also used for returning error estimates when iterative refinement is used. (see § 2.2).

`E` is a `REAL` (`DOUBLE PRECISION` in the D version) variable which must be set positive if iterative refinement is required and non-positive otherwise. On exit it will have one of the following values.

–2 Error condition in which execution could not continue. Any results should be treated as rubbish (see § 2.3).

–1 Error condition found. Execution continued but results may be unreliable. (see § 2.3).

0 Successful entry without iterative refinement.

> 0 Successful entry with iterative refinement. In this case the value of `E` is an estimate of the accuracy of the results (see § 2.2, 2.4)

`DET, IDET` are `REAL` (`DOUBLE PRECISION` in the D version) and `INTEGER` variables respectively, and are set so that the determinant is held as `DET*16**DET`. Usually `IDET=0` but other values will be used if necessary to avoid

overflow or underflow in DET.

For additional facilities see § 2.5.

2.2 Error estimates with iterative refinement in use.

If the diagonal elements of **A** vary widely in size, then the single number E may grossly over-estimate the errors in some of the components. In this case the user should examine the numbers W(1), ... W(N) which give separate estimates for the errors in $x_1, x_2, \ldots x_n$ (MA22A/D) or the numbers W(J)*W(I) which give estimates for the errors in $\mathbf{A}^{-1}_{ij}$ (MA22B).

2.3 Error messages.

MA22A/C/D if it is found that a pivot is small, zero or negative then a diagnostic is printed (limited to one message per entry, later messages being suppressed). Execution is continued but iterative refinement is not attempted and E is set to –1.

MA22B If a pivot is found to be zero, an error message is printed, E set to –2 and a return is made to the calling program. If a small or negative pivot is found the procedure for MA22A/D is adopted. If an element of the inverse is found to be large, an error message is printed, execution continued, and E finally set to –1.

MA22A/B/C/D If N ≤ 0 then E is set to –2 before returning to calling program.

2.4 Method.

Since the matrix is symmetric and positive definite, it is factorised into the form $\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$ where **L** is a unit lower triangular matrix and **D** is a diagonal matrix of positive elements. The square root of the diagonal elements of the matrix **A** are used as scaling factors. These are stored in W(3*N+1), ... W(4*N). The elements of **D** are stored in W(2*N+1), ... W(3*N).

When iterative refinement is requested, this is implicitly performed on the scaled system and continued until the largest change (in x or $\mathbf{A}^{-1}$) is greater than half its value on the previous iteration or is as small as the word-length allows. During this refinement pseudo-random changes are made to the data in accord with its specified accuracy (see § 2.5) and the size of the last maximal change is taken as an accuracy estimate. Finally the scaling factors are multiplied by this estimate to give individual error estimates (see § 2.5) and E is set to the largest of them.

2.5 Use of Comnon

The subroutine contains a common block called MA22E/ED of the form:

The single precision version

```
COMMON/MA22E/ EA,EB,LP
```

The double precision version

```
COMMON/MA22ED/ EA,EB,LP
```

EA is REAL (DOUBLE PRECISION in the D version) variable. If EA ≥ 0 then it gives the fractional accuracy in the elements a_{ij} of **A** . If EA < 0 then |EA| gives the absolute accuracy of the elements. This is defaulted to zero. EA=0 indicates exact data.

EB is REAL (DOUBLE PRECISION in the D version) variable. This is specified similarly to EA and applies to the numbers $b_1, b_2, \ldots b_n$.

LP is INTEGER variable set initially to 6. It specifies the Fortran unit number for messages.

3 GENERAL INFORMATION

Use of common: uses a common block called MA22E/ED, see §2.5.

Workspace: passed as arguments, `W` see § 2.1.

Other routines called directly: calls `FA01` and `FD05`.

Input/output: error messages, see `LP` in §2.5.