1 SUMMARY

To **calculate the minimax solution** to a system of **linear equations subject to simple bounds** on the solution variables. Calculates the vector $\mathbf{x}=\{x_j\}_n$ that minimizes

$$\max_{1\le i\le m}\left|\sum_{j=1}^{n} a_{ij}x_j+b_i\right|$$

subject to $|x_j|\le g$, j=1,2,...,n, where n need not be equal to m.

More general conditions can be applied by making a transformation of the variables and a bound on the accuracy of the maximum is returned.

The method is described in Powell, AERE CSS.11, (1974) and is based on the exchange algorithm with a modification which can short-cut some iterations.

ATTRIBUTES — Version: 1.0.0. **Types:** `MA25A`, `MA25AD`. **Original date:** December 1974. **Origin:** M.J.D.Powell, Harwell.

2 HOW TO USE THE PACKAGE

2.1 The argument list and calling sequence

The single precision version:

```
CALL MA25A(M,N,A,IADIM,B,GBAR,X,TOL,W,IW)
```

The double precision version:

```
CALL MA25AD(M,N,A,IADIM,B,GBAR,X,TOL,W,IW)
```

`M` is an `INTEGER` variable and must be set to the number of equations, i.e. the value of m in § 1.

`N` is an `INTEGER` variable and must be set to the number of variables, i.e. the value of n in § 1.

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) two dimensional array for the coefficients of the equations. The value of a_{ij} in § 1 must be set in `A(i,j)`.

`IADIM` is an `INTEGER` variable and must be set to the first dimension of the array `A`. Therefore the inequality `IADIM` ≥ `M` must hold.

`B` is `REAL` (`DOUBLE PRECISION` in the D version) array for the coefficients b_i, i=1,2,...,m which must be set in `B(i)`.

`GBAR` is a `REAL` (`DOUBLE PRECISION` in the D version) variable, which must be set to the value of g in § 1. It must be positive, and it may be given a very large value if the user does not want any bounds on x_j, j=1,2,...,n. may be used to adjust the steplength, see §4.

`X` is a `REAL` (`DOUBLE PRECISION` in the D version) array in which the subroutine will provided the calculated values of x_j, j=1,2,...n. Therefore the length of the array must be at least n.

`TOL` is a `REAL` (`DOUBLE PRECISION` in the D version) variable, which refers to the accuracy of the calculated maximum residual. We let h^* be the least maximum residual, but due to computer rounding errors it will not be attained. Therefore at the end of the calculation `TOL` is set to a bound on the difference between h^* and the maximum residual given by the final vector of variables. Typically the true error in the final maximum residual is about half the bound. Usually `TOL` is much less than h^*. In this case the final residuals are due mainly to the equations themselves, and not to the rounding errors in the calculation of the subroutine. The value of `TOL` when the subroutines is called is also important. If it set to a positive value, τ say, then the subroutine finishes as soon as it is known that a vector of variables has been found whose residuals are all less than or equal to $(h^* + \tau)$. Thus some iterations can be saved. Usually the initial value of `TOL` should be set to zero, in order that the subroutine calculates the solution to an accuracy that is near the best that can be achieved.

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) work array of length at least ($n^2+9n+12$).

`IW` is an `INTEGER` work array of length at least ($m+3n+4$) .

3 GENERAL INFORMATION

Use of Common: none.

Workspace: all supplied by the user in the arrays `W` and `IW`.

Other subroutines: none.

Input/Output: none.

4 METHOD

The method is described in report CSS 11 by M.J.D.Powell. It is similar to expressing the calculation as a linear programming problem, and solving the dual form. However this basic procedure is extended to save iterations and to gain accuracy. The method is applicable when the equations have several variables. Then one of the solutions is obtained, and in this case some of the variables will be at their bounds $\pm g$.

More general bounds of the form $l_i \le x_i \le u_i$ can be treated by making the change of variables

$x_i \leftarrow (2\, x_i - l_i - u_i) / (u_i - l_i)$, i=1,2,...,n, in which case g=1, and the coefficients of the equations have to be revised.

The subroutine is more advanced than `MA19`, and as far as we can tell it is better from all points of view. It is more accurate because it includes a refinement technique, the value of `TOL` gives the user an indication of the accuracy of the calculation, and numerical experiments show on average that it is faster.