1 SUMMARY

To calculate $\mathbf{A}^{\dagger}$ the **generalized inverse** of an m by n ($m \le n$) rectangular matrix **A** in the special case that the **rank of A is equal to** m**,** i.e. such that $\mathbf{A}\mathbf{A}^{\dagger}\mathbf{A}=\mathbf{A}$ which with full rank can be defined as $\mathbf{A}^{\dagger}=\mathbf{A}^{T}(\mathbf{A}\mathbf{A}^{T})^{-1}$.

Householder type orthogonal transformations with row and column interchanges are used in a method described in M.J.D. Powell, AERE R.6072.

ATTRIBUTES — **Version:** 1.0.0. **Types:** `MB11A`; `MB11AD`. **Original date:** May 1969. **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 MB11A(M,N,A,IA,W)
```

The double precision version:

```
CALL MB11AD(M,N,A,IA,W)
```

`M` is an `INTEGER` variable set to m the number of rows in the matrix **A**.

`N` is an `INTEGER` variable set to n the number of columns in the matrix **A**.

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) two dimensional array which must be set to contain the elements of the matrix **A**. i.e. `A(I,J)=` a_{ij} `I=1,2,...,M, J=1,2,...,N.`

On exit the array `A` will have been overwritten by its generalised inverse so that `A(I,J)` will be changed to the `(I,J)` th element of $\mathbf{A}^{\dagger T}$.

`IA` is an `INTEGER` variable set to the first dimension of the array `A.` Note that we must have `IA` ≥ `M.`

`W` is a `REAL` (`DOUBLE PRECISION` in the D version) workspace array of length at least $2m+n$

3 GENERAL INFORMATION

Use of Common: none.

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

Other subroutines: None

Input/Output: none.

4 METHOD

First **A** is transformed to a lower triangular form, by a sequence of m elementary Householder transformations, taking account of row and column interchanges. This lower triangular matrix is inverted, and then it is replaced by another matrix that contains the same information in a more convenient form. Because of this replacement, we can now re-apply the elementary transformations to the inverted matrix, to obtain the required generalised inverse, without requiring extra storage space. The method is given in M.J.D.Powell, ‘A Fortran subroutine to invert a rectangular matrix of full rank’, AERE Report R-6072.