# Title



**lapack() — LAPACK linear-algebra functions**

Description Syntax Remarks and examples Reference Also see

# Description

LA_DGBMV(), LA_DGEBAK(), LA_ZGEBAK(), LA_DGEBAL(), LA_ZGEBAL(), ... are LAPACK functions in original, as-is form; see [M-1] **LAPACK**. These functions form the basis for many of Mata's linear-algebra capabilities. Mata functions such as cholesky(), svd(), and eigensystem() are implemented using these functions; see [M-4] **matrix**. Those functions are easier to use. The LA_*() functions provide more capability.

_flopin() and _flopout() convert matrices to and from the form required by the LA_*() functions.

# Syntax

*void* _flopin(*numeric matrix A*)

*void* *lapack_function*(...)

*void* _flopout(*numeric matrix A*)

where *lapack_function* may be

```
LA_DGBMV()
LA_DGEBAK()      LA_ZGEBAK()
LA_DGEBAL()      LA_ZGEBAL()
LA_DGEES()       LA_ZGEES()
LA_DGEEV()       LA_ZGEEV()
LA_DGEHRD()      LA_ZGEHRD()
LA_DGGBAK()      LA_ZGGBAK()
LA_DGGBAL()      LA_ZGGBAL()
LA_DGGHRD()      LA_ZGGHRD()
LA_DHGEQZ()      LA_ZHGEQZ()
LA_DHSEIN()      LA_ZHSEIN()
LA_DHSEQR()      LA_ZHSEQR()

LA_DLAMCH()
LA_DORGHR()
LA_DSYEVX()

LA_DTGSEN()      LA_ZTGSEN()
LA_DTGEVC()      LA_ZTGEVC()
LA_DTREVC()      LA_ZTREVC()
LA_DTRSEN()      LA_ZTRSEN()

                 LA_ZUNGHR()
```

## Remarks and examples



LAPACK stands for Linear Algebra PACKage and is a freely available set of Fortran 90 routines for solving systems of simultaneous equations, eigenvalue problems, and singular-value problems. The original Fortran routines have six-letter names like DGEHRD, DORGHR, and so on. The Mata functions `LA_DGEHRD()`, `LA_DORGHR()`, etc., are a subset of the LAPACK double-precision real and complex routine. All LAPACK double-precision functions will eventually be made available.

Documentation for the LAPACK routines can be found at http://www.netlib.org/lapack/, although we recommend obtaining *LAPACK Users' Guide* by Anderson et al. (1999).

Remarks are presented under the following headings:



## Mapping calling sequence from Fortran to Mata

LAPACK functions are named with first letter S, D, C, or Z. S means single-precision real, D means double-precision real, C means single-precision complex, and Z means double-precision complex. Mata provides the D* and Z* functions. The LAPACK documentation is in terms of S* and C*. Thus, to find the documentation for LA_DGEHRD, you must look up SGEHRD in the original documentation.

The documentation (Anderson et al. 1999, 227) reads, in part,

```
SUBROUTINE SGEHRD(N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER    IHI, ILO, INFO, LDA, LWORK, N
REAL       A(LDA, *), TAU(*), WORK(LWORK)
```

and the documentation states that SGEHDR reduces a real, general matrix, $\mathbf{A}$, to upper Hessenberg form, $\mathbf{H}$, by an orthogonal similarity transformation: $\mathbf{Q}' \times \mathbf{A} \times \mathbf{Q} = \mathbf{H}$.

The corresponding Mata function, `LA_DGEHRD()`, has the same arguments. In Mata, arguments `ihi`, `ilo`, `info`, `lda`, `lwork`, and `n` are *real scalars*. Argument `A` is a *real matrix*, and arguments `tau` and `work` are *real vectors*.

You can read the rest of the original documentation to find out what is to be placed (or returned) in each argument. It turns out that $\mathbf{A}$ is assumed to be dimensioned LDA $\times$ *something* and that the routine works on $\mathbf{A}(1,1)$ (using Fortran notation) through $\mathbf{A}(N,N)$. The routine also needs work space, which you are to supply in vector WORK. In the standard LAPACK way, LAPACK offers you a choice: you can preallocate WORK, in which case you have to choose a fairly large dimension for it, or you can do a query to find out how large the dimension needs to be for this particular problem. If you preallocate, the documentation reveals that the WORK must be of size $N$, and you set LWORK equal to $N$. If you wish to query, then you make WORK of size 1 and set LWORK equal to $-1$. The LAPACK routine will then return in the first element of WORK the optimal size. Then you call the function again with WORK allocated to be the optimal size and LWORK set to equal the optimal size.

Concerning Mata, the above works. You can follow the LAPACK documentation to the letter. Use `J()` to allocate matrices or vectors. Alternatively, you can specify all sizes as missing value (`.`), and Mata will fill in the appropriate value based on the assumption that you are using the entire matrix.

Thus, in LA_DGEHRD(), you could specify lda as missing, and the function would run as if you had specified lda equal to cols(A). You could specify n as missing, and the function would run as if you had specified n as rows(A).

Work areas, however, are treated differently. You can follow the standard LAPACK convention outlined above; or you can specify the sizes of work areas (lwork) and specify the work areas themselves (work) as missing values, and Mata will allocate the work areas for you. The allocation will be as you specified.

One feature provided by some LAPACK functions is not supported by the Mata implementation. If a function allows a function pointer, you may not avail yourself of that option.

## Flopping: Preparing matrices for LAPACK

The LAPACK functions provided in Mata are the original LAPACK functions. Mata, which is C based, stores matrices rowwise. LAPACK, which is Fortran based, stores matrices columnwise. Mata and Fortran also disagree on how complex matrices are to be organized.

Functions _flopin() and _flopout() handle these issues. Coding _flopin(*A*) changes matrix *A* from the Mata convention to the LAPACK convention. Coding _flopout(*A*) changes *A* from the LAPACK convention to the Mata convention.

The LA_*() functions do not do this for you because LAPACK often takes two or three LAPACK functions run in sequence to achieve the desired result, and it would be a waste of computer time to switch conventions between calls.

## Warning on the use of rows() and cols() after _flopin()

Be careful using the rows() and cols() functions. rows() of a flopped matrix returns the logical number of columns and cols() of a flopped matrix returns the logical number of rows!

The danger of confusion is especially great when using J() to allocate work areas. If a LAPACK function requires a work area of $r \times c$, you code,

```
_LA_function(..., J(c, r, .), ...)
```

## Warning: It is your responsibility to check info

The LAPACK functions do not abort with error on failure. They instead store 0 in info (usually the last argument) if successful and store an error code if not successful. The error code is usually negative and indicates the argument that is a problem.

## Example

The following example uses the LAPACK function DGEHRD to obtain the Hessenberg form of matrix **A**. We will begin with

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 |
| 2 | 4 | 5 | 6 | 7 |
| 3 | 7 | 8 | 9 | 10 |
| 4 | 8 | 9 | 10 | 11 |

The first step is to use `_flopin()` to put **A** in LAPACK order:

```
: _flopin(A)
```

Next we make a work-space query to get the optimal size of the work area.

```
: LA_DGEHRD(., 1, 4, A, ., tau=., work=., lwork=-1, info=0)
: lwork = work[1,1]
: lwork
  128
```

After putting the work-space size in `lwork`, we can call `LA_DGEHRD()` again to perform the Hessenberg decomposition:

```
: LA_DGEHRD(., 1, 4, A, ., tau=., work=., lwork, info=0)
```

LAPACK function DGEHRD saves the result in the upper triangle and the first subdiagonal of **A**. We must use `_flopout()` to change that back to Mata order, and finally, we extract the result:

```
: _flopout(A)
: A = A-sublowertriangle(A, 2)
: A
                 1              2              3              4
    +-------------------------------------------------------------+
  1 |            1   -5.370750529    .0345341258    .3922322703  |
  2 | -11.35781669    25.18604651    -4.40577178   -.6561483899  |
  3 |            0   -1.660145888   -.1860465116    .1760901813  |
  4 |            0              0   -8.32667e-16   -5.27356e-16  |
    +-------------------------------------------------------------+
```

# Reference

Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. 1999. *LAPACK Users' Guide*. 3rd ed. Philadelphia: Society for Industrial and Applied Mathematics.

# Also see

[M-1] **LAPACK** — The LAPACK linear-algebra routines

[R] **copyright lapack** — LAPACK copyright notification

[M-4] **matrix** — Matrix functions