luinv() — Square matrix inversion

Description Syntax Remarks and examples Conformability Diagnostics Also see

Description

`luinv(`A`)` and `luinv(`A`,` *tol*`)` return the inverse of real or complex, square matrix A.

`_luinv(`A`)` and `_luinv(`A`,` *tol*`)` do the same thing except that, rather than returning the inverse matrix, they overwrite the original matrix A with the inverse.

`_luinv_la(`A`,` b`)` is the interface to the LAPACK routines that do the work. The output b is a real scalar, which is missing if the MKL LAPACK routine was used, is 1 if the Netlib LAPACK routine used a blocked algorithm, and is 0 otherwise. This function uses the MKL LAPACK by default.

`_luinv_lapacke(`A`)` and `_luinv_lapacke(`A`,` *tol*`)` are the interfaces to the LAPACK routines that do the work. Both of these function use MKL LAPACK by default.

In all cases, optional argument *tol* specifies the tolerance for determining singularity; see *Remarks and examples* below.

Syntax

numeric matrix	`luinv(`*numeric matrix A*`)`
numeric matrix	`luinv(`*numeric matrix A*`,` *real scalar tol*`)`
void	`_luinv(`*numeric matrix A*`)`
void	`_luinv(`*numeric matrix A*`,` *real scalar tol*`)`
real scalar	`_luinv_la(`*numeric matrix A*`,` *b*`)`
real scalar	`_luinv_lapacke(`*numeric matrix A*`)`
real scalar	`_luinv_lapacke(`*numeric matrix A*`,` *real scalar tol*`)`

Remarks and examples

These routines calculate the inverse of A. The inverse matrix A^{-1} of A satisfies the conditions

$$AA^{-1} = I$$
$$A^{-1}A = I$$

A is required to be square and of full rank. See [M-5] **qrinv()** and [M-5] **pinv()** for generalized inverses of nonsquare or rank-deficient matrices. See [M-5] **invsym()** for inversion of real, symmetric matrices.

`luinv(`A`)` is logically equivalent to `lusolve(`A`, I(rows(`A`)))`; see [M-5] **lusolve()** for details and for use of the optional *tol* argument.

Conformability

`luinv(`A`,` *tol*`)`:

A:	$n \times n$	
tol:	1×1	(optional)
result:	$n \times n$	

`_luinv(`A`,` *tol*`)`:

input:

A:	$n \times n$	
tol:	1×1	(optional)

output:

A:	$n \times n$

`_luinv_la(`A`,` *b*`)`:

input:

A:	$n \times n$

output:

A:	$n \times n$
b:	1×1
result:	1×1

`_luinv_lapacke(`A`,` *tol*`)`:

input:

A:	$n \times n$

output:

A:	$n \times n$	
tol:	1×1	(optional)
result:	1×1	

Diagnostics

The inverse returned by these functions is real if *A* is real and is complex if *A* is complex. If you use these functions with a singular matrix, returned will be a matrix of missing values. The determination of singularity is made relative to *tol*. See *Tolerance* under *Remarks and examples* in [M-5] **lusolve()** for details.

`luinv(`*A*`)` and `_luinv(`*A*`)` return a matrix containing missing if *A* contains missing values.

`_luinv(`*A*`)` aborts with error if *A* is a view.

`_luinv_la(`*A*`,` *b*`)`, `_luinv_lapacke(`*A*`)`, and `_luinv_lapacke(`*A*`,` *tol*`)` should not be used directly; use `_luinv()`.

See [M-5] **lusolve()** and [M-1] **Tolerance** for information on the optional *tol* argument.

Also see

[M-5] **cholinv( )** — Symmetric, positive-definite matrix inversion

[M-5] **_invmat( )** — Inverse and pseudoinverse of a square matrix

[M-5] **invsym( )** — Symmetric real matrix inversion

[M-5] **lud( )** — LU decomposition

[M-5] **lusolve( )** — Solve AX=B for X using LU decomposition

[M-5] **pinv( )** — Moore–Penrose pseudoinverse

[M-5] **qrinv( )** — Generalized inverse of matrix via QR decomposition

[M-4] **Matrix** — Matrix functions

[M-4] **Solvers** — Functions to solve AX=B and to obtain A inverse

For suggested citations, see the FAQ on citing Stata documentation.