

[Description](#)[Syntax](#)[Remarks and examples](#)[Conformability](#)[Diagnostics](#)[Also see](#)

Description

`lsolve(A, B, c, d)` finds the minimum-norm least-squares solution for $\min \|Ax - c\|_2$ subject to $Bx = d$ and returns x . A does not have to be a square matrix.

`_lsolve(A, B, c, d, x)` does the same thing except that it overwrites x with the solution and returns 0 if a solution was found and 1 otherwise. If 1 is returned, x is overwritten with a vector of missing values.

`_lse_lapacke(A, B, c, d, x)` is the interface to the LAPACK routines that do the work. It returns 0 if a solution was found and 1 otherwise. Direct use of `_lse_lapacke()` is not recommended.

Note that these functions can be used only when set `lapack_mkl` on is in effect on Windows or Linux or when set `lapack_openblas` on is in effect on Mac; see [\[M-1\] LAPACK](#).

Syntax

numeric vector `lsolve(A, B, c, d)`
real scalar `_lsolve(A, B, c, d, x)`
real scalar `_lse_lapacke(A, B, c, d, x)`

where inputs are

A : $m \times n$ numeric matrix
 B : $p \times n$ numeric matrix
 c : $m \times 1$ or $1 \times m$ numeric vector
 d : $p \times 1$ or $1 \times p$ numeric vector

and outputs are

x : $n \times 1$ numeric vector
result: real scalar

where $p \leq n \leq m + p$, the rank of matrix B is p , and the rank of the following matrix is n :

$$\begin{bmatrix} A \\ m \times n \\ B \\ p \times n \end{bmatrix}$$

Remarks and examples

Remarks are presented under the following headings:

[Introduction](#)[Examples](#)

Introduction

The above functions solve $Ax = c$ subject to equality constraints $Bx = d$ via the least-squares method. A need not be square.

To obtain a unique solution, the functions require that

1. $p \leq n \leq m + p$,
2. the rank of matrix B is p , and
3. the rank of the following matrix is n :

$$\begin{bmatrix} A \\ B \end{bmatrix}$$

$m \times n$
 $p \times n$

The solution is found with the underlying LAPACK routines using a generalized RQ factorization of (B, A) .

Examples

► Example 1: Least squares with equality constraints

Given A , B , c , and d , we can find x , satisfying $\min \|Ax - c\|_2$ subject to $Bx = d$ using

: A = (3, 2 \ 2, 10 \ 1, 1.5)

: B = (2, 5 \ 1, 6)

: c = (1 \ 2.5 \ 5)

: d = (10 \ 2)

: A

	1	2
1	3	2
2	2	10
3	1	1.5

: B

	1	2
1	2	5
2	1	6

: c

	1
1	1
2	2.5
3	5

: d

	1
1	10
2	2

```

: x = lsesolve(A, B, c, d)
: x
      1
1  7.142857143
2 -0.8571428571

```

We can also use the `_lsesolve()` function to get the same solution as above and a return code of 0:

```

: A = (3, 2 \ 2, 10 \ 1, 1.5)
: B = (2, 5 \ 1, 6)
: c = (1 \ 2.5 \ 5)
: d = (10 \ 2)
: A
      1      2
1  3      2
2  2     10
3  1     1.5
: B
      1      2
1  2      5
2  1      6
: c
      1
1  1
2  2.5
3  5
: d
      1
1  10
2  2
:
: rc = _lsesolve(A, B, c, d, x = .)
: x
      1
1  7.142857143
2 -0.8571428571
: rc
0

```

Conformability

`lscsolve(A, B, c, d):`

input:

$A:$ $m \times n$
 $B:$ $p \times n$
 $c:$ $m \times 1$ or $1 \times m$
 $d:$ $p \times 1$ or $1 \times p$

output:

$x:$ $n \times 1$

`_lscsolve(A, B, c, d, x):`

input:

$A:$ $m \times n$
 $B:$ $p \times n$
 $c:$ $m \times 1$ or $1 \times m$
 $d:$ $p \times 1$ or $1 \times p$

output:

$x:$ $n \times 1$
result: 1×1

`_lse_lapacke(A, B, c, d, x) :`

input:

$A:$ $m \times n$
 $B:$ $p \times n$
 $c:$ $m \times 1$ or $1 \times m$
 $d:$ $p \times 1$ or $1 \times p$

output:

$x:$ $n \times 1$
result: 1×1

Diagnostics

`lscsolve(A, B, ...)`, `_lscsolve(A, B, ...)`, and `_lse_lapacke(A, B, ...)` return a result containing missing if A , B , c , or d contains missing values. If the conditions in [Introduction](#) above are not satisfied, the functions will try to find a solution, which will either produce unstable results or abort with error. The functions abort with error if set `lapack_mkl` on is not in effect on Windows or Linux or when set `lapack_openblas` on is not in effect on Mac.

`_lscsolve(A, B, ...)` and `_lse_lapacke(A, B, ...)` abort with error if A , B , c , or d is a view.

`_lse_lapacke(A, B, ...)` aborts with error if A , B , c , and d are not all real or all complex.

`_lse_lapacke(A, B, ...)` should not be used directly; use `_lscsolve()`.

Also see

[M-5] **cholsolve()** — Solve $AX=B$ for X using Cholesky decomposition

[M-5] **lsglmsolve()** — Solves a general Gauss–Markov linear model problem

[M-5] **lssolve()** — Solve $AX=B$ for X using least-squares method

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

[M-5] **qrsolve()** — Solve $AX=B$ for X using QR decomposition

[M-5] **_solvemmat()** — Solve $AX=B$ for X

[M-5] **svsolve()** — Solve $AX=B$ for X using singular value decomposition

[M-4] **Matrix** — Matrix functions

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

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the FAQ on [citing Stata documentation](#).