

Solvers — Functions to solve $AX=B$ and to obtain A inverse
[Contents](#)[Description](#)[Remarks and examples](#)[Also see](#)

Contents

[M-5] Manual entry	Function	Purpose
Solvers		
cholsolve()	cholsolve() cholsolve_lapacke()	<i>A</i> positive definite; symmetric or Hermitian <i>A</i> positive definite using LAPACK routines; symmetric or Hermitian
lusolve()	lusolve()	<i>A</i> full rank, square, real or complex
qrsolve()	qrsolve()	<i>A</i> general; $m \times n$, $m \geq n$, real or complex; least-squares generalized solution
svsolve()	svsolve()	generalized; $m \times n$, real or complex; minimum norm, least-squares solution
Inverters		
invsym()	invsym()	generalized; real symmetric
cholinv()	cholinv() cholinv_lapacke()	positive definite; symmetric or Hermitian positive definite using LAPACK routines; symmetric or Hermitian
luinv()	luinv()	full rank; square; real or complex
qrinv()	qrinv()	generalized; $m \times n$, $m \geq n$; real or complex
pinv()	pinv()	generalized; $m \times n$, real or complex Moore–Penrose pseudoinverse

Description

The above functions solve $AX = B$ for X and solve for A^{-1} .

Remarks and examples

Matrix solvers can be used to implement matrix inverters, and so the two nearly always come as a pair.

Solvers solve $AX = B$ for X . One way to obtain A^{-1} is to solve $AX = I$. If $f(A, B)$ solves $AX=B$, then $f(A, I(\text{rows}(A)))$ solves for the inverse. Some matrix inverters are in fact implemented this way, although usually custom code is written because memory savings are possible when it is known that $B = I$.

The pairings of inverter and solver are

inverter	solver
<code>invsym()</code>	(none)
<code>cholinv()</code>	<code>cholsolve()</code>
<code>cholinvlapacke()</code>	<code>cholsolvelapacke()</code>
<code>luinv()</code>	<code>lusolve()</code>
<code>qrinv()</code>	<code>qrsolve()</code>
<code>pinv()</code>	<code>svsolve()</code>

Also see

[M-4] [Intro](#) — Categorical guide to Mata functions