

**Description**

qrsolve(A, B, ...) uses QR decomposition to solve \( AX = B \) and returns \( X \). When \( A \) is singular or nonsquare, qrsolve() computes a least-squares generalized solution. When rank is specified, it is placed the rank of \( A \).

\( \_\text{qrsolve}(A, B, ...) \), does the same thing, except that it destroys the contents of \( A \) and it overwrites \( B \) with the solution. Returned is the rank of \( A \).

In both cases, tol specifies the tolerance for determining whether \( A \) is of full rank. tol is interpreted in the standard way—as a multiplier for the default if \( tol > 0 \) is specified and as an absolute quantity to use in place of the default if \( tol \leq 0 \) is specified; see [M-1] tolerance.

**Syntax**

```
numeric matrix qrsolve(A, B)
numeric matrix qrsolve(A, B, rank)
numeric matrix qrsolve(A, B, rank, tol)
real scalar \_qrsolve(A, B)
real scalar \_qrsolve(A, B, tol)
```

where

- \( A \): numeric matrix
- \( B \): numeric matrix
- rank: irrelevant; real scalar returned
- tol: real scalar

**Remarks and examples**

qrsolve(A, B, ...) is suitable for use with square and possibly rank-deficient matrix \( A \), or when \( A \) has more rows than columns. When \( A \) is square and full rank, qrsolve() returns the same solution as lusolve() (see [M-5] lusolve()), up to roundoff error. When \( A \) is singular, qrsolve() returns a generalized (least-squares) solution.

Remarks are presented under the following headings:

- Derivation
- Relationship to inversion
- Tolerance
Derivation

We wish to solve for \( X \)

\[ AX = B \]  \hspace{1cm} (1)

Perform QR decomposition on \( A \) so that we have \( A = QRP' \). Then (1) can be rewritten as

\[ QRP'X = B \]

Premultiplying by \( Q' \) and remembering that \( Q'Q = QQ' = I \), we have

\[ RP'X = Q'B \]  \hspace{1cm} (2)

Define

\[ Z = P'X \]  \hspace{1cm} (3)

Then (2) can be rewritten as

\[ RZ = Q'B \]  \hspace{1cm} (4)

It is easy to solve (4) for \( Z \) because \( R \) is upper triangular. Having \( Z \), we can obtain \( X \) via (3), because \( Z = P'X \), premultiplied by \( P \) (and if we remember that \( PP' = I \)), yields

\[ X = PZ \]

For more information on QR decomposition, see [M-5] qrd().

Relationship to inversion

For a general discussion, see Relationship to inversion in [M-5] lusolve().

For an inverse based on QR decomposition, see [M-5] qrinv(). qrinv(\( A \)) amounts to qrsolve(\( A, I(\text{rows}(A)) \)), although it is not actually implemented that way.

Tolerance

The default tolerance used is

\[ \text{eta} = 1e-13 \times \text{trace(abs(R))} / \text{rows(R)} \]

where \( R \) is the upper-triangular matrix of the QR decomposition; see Derivation above. When \( A \) is less than full rank, by, say, \( d \) degrees of freedom, then \( R \) is also rank deficient by \( d \) degrees of freedom and the bottom \( d \) rows of \( R \) are essentially zero. If the \( i \)th diagonal element of \( R \) is less than or equal to \( \text{eta} \), then the \( i \)th row of \( Z \) is set to zero. Thus if the matrix is singular, qrsolve() provides a generalized solution.

If you specify \( tol > 0 \), the value you specify is used to multiply \( \text{eta} \). You may instead specify \( tol \leq 0 \), and then the negative of the value you specify is used in place of \( \text{eta} \); see [M-1] tolerance.
Conformability

\texttt{qrsolve}(A, B, rank, tol):

\textbf{input}:

\begin{align*}
A &: \quad m \times n, \quad m \geq n \\
B &: \quad m \times k \\
tol &: \quad 1 \times 1 \quad \text{(optional)}
\end{align*}

\textbf{output}:

\begin{align*}
\text{rank} &: \quad 1 \times 1 \quad \text{(optional)} \\
\text{result} &: \quad n \times k
\end{align*}

\texttt{-qrsolve}(A, B, tol):

\textbf{input}:

\begin{align*}
A &: \quad m \times n, \quad m \geq n \\
B &: \quad m \times k \\
tol &: \quad 1 \times 1 \quad \text{(optional)}
\end{align*}

\textbf{output}:

\begin{align*}
A &: \quad 0 \times 0 \\
B &: \quad n \times k \\
\text{result} &: \quad 1 \times 1
\end{align*}

Diagnostics

\texttt{qrsolve}(A, B, \ldots) \text{ and } \texttt{-qrsolve}(A, B, \ldots) \text{ return a result containing missing if } A \text{ or } B \text{ contain missing values.}

\texttt{-qrsolve}(A, B, \ldots) \text{ aborts with error if } A \text{ or } B \text{ are views.}

Also see

[M-5] \texttt{qrinv()} — Generalized inverse of matrix via QR decomposition

[M-5] \texttt{qrdf()} — QR decomposition

[M-5] \texttt{solvelower()} — Solve AX=B for X, A triangular

[M-5] \texttt{cholsolve()} — Solve AX=B for X using Cholesky decomposition

[M-5] \texttt{lusolve()} — Solve AX=B for X using LU decomposition

[M-5] \texttt{svsolve()} — Solve AX=B for X using singular value decomposition

[M-5] \texttt{solve\_tol()} — Tolerance used by solvers and inverters

[M-4] \texttt{matrix} — Matrix functions

[M-4] \texttt{solvers} — Functions to solve AX=B and to obtain A inverse