

ldl() — Bunch–Kaufman decomposition

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

Description

`ldl(A, L, D, p)` returns the Bunch–Kaufman decomposition (with diagonal pivoting) of A in a permuted lower-triangular matrix L and a symmetric block-diagonal matrix D with 1×1 and 2×2 diagonal blocks, along with a permutation vector p .

With the permutation vector, $L[p, .]$ becomes a lower-triangular matrix with unit diagonal.

Up to roundoff error, the returned results are such that

$$A[p, p] = L[p, .] * D * L[p, .]'$$

`ldl(A, L, D)` is similar to `ldl(A, L, D, p)`, but the permutation vector p is omitted from the output.

Up to roundoff error, the returned results are such that

$$A = L * D * L'$$

Syntax

`void ldl(numeric matrix A, L, D)`

`void ldl(numeric matrix A, L, D, p)`

where

1. A is symmetric ([Hermitian](#)) indefinite.
2. the types of L , D , and p are irrelevant; results are returned there.

Remarks and examples

stata.com

The Bunch–Kaufman decomposition is a generalization of the [Cholesky decomposition](#).

Bunch–Kaufman decomposition of matrix A can be written as

$$PAP' = PLDL' P'$$

where P is a [permutation matrix](#) that permutes the rows of A .

L is the permuted lower-triangular matrix. With the permutation matrix P , PL is a lower-triangular matrix with unit diagonal.

2 `ldl()` — Bunch–Kaufman decomposition

D is the symmetric block-diagonal matrix D with 1×1 and 2×2 diagonal blocks.

Rather than returning P directly, returned is p corresponding to P . Lowercase p is a column vector that contains the subscripts of the rows in the desired order. That is,

$$PL = L[p, .]$$

The advantage of this is that p requires less memory than P , and the reorganization, should it be desired, can be performed more quickly; see [M-1] **Permutation**.

► Example 1: Bunch–Kaufman decomposition

The Bunch–Kaufman decomposition of A can be written as

$$A = L * D * L'$$

`ldl(A, L, D)` will make this calculation:

```
: A
[symmetric]
   1   2   3
1   2
2   3   2
3   4   1   2

: ldl(A, L = ., D = ., p = .)
: L
   1   2   3
1   1   0   0
2  -.1666666667 .8333333333 1
3   0   1   0

: D
[symmetric]
   1   2   3
1   2
2   4   2
3   0   0  1.666666667

: p
   1
1   1
2   3
3   2

: mreldif(A, L * D * L')
1.11022e-16
: mreldif(A[p, p], L[p, .] * D * L[p, .]')
1.11022e-16
```

Conformability

`ldl(A, L, D, p)`:

input:

A: $n \times n$

output:

L: $n \times n$

D: $n \times n$

p: $n \times 1$ (optional)

Diagnostics

`ldl(A, L, D, p)` returns missing results if *A* contains missing values.

Also see

[M-5] [cholesky\(\)](#) — Cholesky square-root decomposition

[M-4] [Matrix](#) — Matrix functions