decomposeChol

Computes the Cholesky Decomposition of a symmetric positive definite matrix A and calculate the log determinate as side product of the decomposition. Computes the lower triangle matrix L.

Syntax

```
[L, logDet] = decomposeChol(A)
```

Description

[L, logDet] = decomposeChol(A) computes lower triangle matrix of input matrix A using the Cholesky Decomposition. As side product of the decomposition the logarithmic determinante of A is returned too.

Examples

```
A = [1.0, 0.9, 0.8;
     0.9, 1.0, 0.9;
     0.8, 0.9, 1.0];
[L, logDet] = decomposeChol(A)
assert(all(L*L'==A,'all'))
assert(log(det(A)) == logDet)
```

Input Argurments

A symmetric, pos. finite double matrix of size N x N.

Output Argurments

L is a lower trinagle matrix of size N x N. Multiply L with its transposed to get matrix A.

logDet is a scalar and the logaritmic determinante of A. Computed along the diagonal of L.

Requirements

- Other m-files required: None
- Subfunctions: chol, mustBeValidMatrix
- MAT-files required: None

See Also

- chol

Created on January 04. 2021 by Tobias Wulf. Copyright Tobias Wulf 2021.

```
function [L, logDet] = decomposeChol(A)
    arguments
        A (:,:) double {mustBeReal, mustBeValidMatrix(A)}
    end

    % decompose A to lower triangle matrix
    [L, flag]= chol(A, 'lower');
    if flag ~= 0
        eid = 'chol:Fails';
        msg = 'Cholesky Decomposition fails.';
        throwAsCaller(MException(eid,msg))
    end
    % compute log determinate of A
    if nargout > 1, logDet = 2 * sum(log(diag(L))); end
end

% Custom validation function
function mustBeValidMatrix(A)
    % test if is matrix
    if ~ismatrix(A)
        eid = 'Matrix:notMatrix';
        msg = 'A is not a matrix.';
        throwAsCaller(MException(eid,msg))
    end
    % Test for N x N
    if ~isequal(size(A,1),size(A,2))
        eid = 'Matrix:notNxN';
        msg = 'Size of matrix is not N x N.';
        throwAsCaller(MException(eid,msg))
    end
    % test if symmetric
    if ~issymmetric(A)
        eid = 'Matrix:notSymmetric';
        msg = 'Matrix is not symmetric.';
        throwAsCaller(MException(eid,msg))
    end
    % test if positive definite
    if ~all(eig(A) >= 0)
        eid = 'Matrix:notPosDefinte';
        msg = 'Matrix is not pos. definte.';
        throwAsCaller(MException(eid,msg))
    end
end
```

Published with MATLAB® R2020b