

Cholesky decomposition

From Wikipedia, the free encyclopedia

In linear algebra, the **Cholesky decomposition** or **Cholesky factorization** is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, useful for efficient numerical solutions and Monte Carlo simulations. It was discovered by André-Louis Cholesky for real matrices. When it is applicable, the Cholesky decomposition is roughly twice as efficient as the LU decomposition for solving systems of linear equations.^[1]

Contents

- 1 Statement
- 2 LDL decomposition
- 3 Example
- 4 Applications
 - 4.1 Linear least squares
 - 4.2 Non-linear optimization
 - 4.3 Monte Carlo simulation
 - 4.4 Kalman filters
 - 4.5 Matrix inversion
- 5 Computation
 - 5.1 The Cholesky algorithm
 - 5.2 The Cholesky–Banachiewicz and Cholesky–Crout algorithms
 - 5.3 Stability of the computation
 - 5.4 LDL decomposition
 - 5.5 Block variant
 - 5.6 Updating the decomposition
 - 5.6.1 Rank-one update
 - 5.6.2 Rank-one downdate
- 6 Proof for positive semi-definite matrices
- 7 Generalization
- 8 Implementations in programming languages
- 9 See also
- 10 Notes
- 11 References
- 12 External links
 - 12.1 History of science
 - 12.2 Information
 - 12.3 Computer code
 - 12.4 Use of the matrix in simulation
 - 12.5 Online calculators

Statement

The Cholesky decomposition of a Hermitian positive-definite matrix **A** is a decomposition of the form

$$\mathbf{A} = \mathbf{L}\mathbf{L}^*$$

where \mathbf{L} is a lower triangular matrix with real and positive diagonal entries, and \mathbf{L}^* denotes the conjugate transpose of \mathbf{L} . Every Hermitian positive-definite matrix (and thus also every real-valued symmetric positive-definite matrix) has a unique Cholesky decomposition.^[2]

If the matrix \mathbf{A} is Hermitian and positive semi-definite, then it still has a decomposition of the form $\mathbf{A} = \mathbf{L}\mathbf{L}^*$ if the diagonal entries of \mathbf{L} are allowed to be zero.^[3]

When \mathbf{A} has real entries, \mathbf{L} has real entries as well and the factorization may be written $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ ^[4]

The Cholesky decomposition is unique when \mathbf{A} is positive definite; there is only one lower triangular matrix \mathbf{L} with strictly positive diagonal entries such that $\mathbf{A} = \mathbf{L}\mathbf{L}^*$. However, the decomposition need not be unique when \mathbf{A} is positive semidefinite.

The converse holds trivially: if \mathbf{A} can be written as $\mathbf{L}\mathbf{L}^*$ for some invertible \mathbf{L} , lower triangular or otherwise, then \mathbf{A} is Hermitian and positive definite.

LDL decomposition

A closely related variant of the classical Cholesky decomposition is the LDL decomposition,

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^*$$

where \mathbf{L} is a lower unit triangular (unitriangular) matrix and \mathbf{D} is a diagonal matrix.

This decomposition is related to the classical Cholesky decomposition, of the form $\mathbf{L}\mathbf{L}^*$, as follows:

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^* = \mathbf{L}\mathbf{D}^{\frac{1}{2}}\mathbf{D}^{\frac{1}{2}*}\mathbf{L}^* = \mathbf{L}\mathbf{D}^{\frac{1}{2}}(\mathbf{L}\mathbf{D}^{\frac{1}{2}})^*$$

The LDL variant, if efficiently implemented, requires the same space and computational complexity to construct and use but avoids extracting square roots.^[5] Some indefinite matrices for which no Cholesky decomposition exists have an LDL decomposition with negative entries in \mathbf{D} . For these reasons, the LDL decomposition may be preferred. For real matrices, the factorization has the form $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ and is often referred to as **LDLT decomposition** (or \mathbf{LDL}^T decomposition). It is closely related to the eigendecomposition of real symmetric matrices, $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$.

Example

Here is the Cholesky decomposition of a symmetric real matrix:

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & & \\ 6 & 1 & \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ & 1 & 5 \\ & & 3 \end{pmatrix}$$

And here is its \mathbf{LDL}^T decomposition:

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 3 & 1 & \\ -4 & 5 & 1 \end{pmatrix} \begin{pmatrix} 4 & & \\ & 1 & \\ & & 9 \end{pmatrix} \begin{pmatrix} 1 & 3 & -4 \\ & 1 & 5 \\ & & 1 \end{pmatrix}$$

Applications

The Cholesky decomposition is mainly used for the numerical solution of linear equations $\mathbf{Ax} = \mathbf{b}$. If \mathbf{A} is symmetric and positive definite, then we can solve $\mathbf{Ax} = \mathbf{b}$ by first computing the Cholesky decomposition $\mathbf{A} = \mathbf{LL}^*$, then solving $\mathbf{Ly} = \mathbf{b}$ for \mathbf{y} by forward substitution, and finally solving $\mathbf{L}^*\mathbf{x} = \mathbf{y}$ for \mathbf{x} by back substitution.

For linear systems that can be put into symmetric form, the Cholesky decomposition (or its LDL variant) is the method of choice, for superior efficiency and numerical stability. Compared to the LU decomposition, it is roughly twice as efficient.

Linear least squares

Systems of the form $\mathbf{Ax} = \mathbf{b}$ with \mathbf{A} symmetric and positive definite arise quite often in applications. For instance, the normal equations in linear least squares problems are of this form. It may also happen that matrix \mathbf{A} comes from an energy functional which must be positive from physical considerations; this happens frequently in the numerical solution of partial differential equations.

Non-linear optimization

Non-linear multi-variate functions may be minimized over their parameters using variants of Newton's method called *quasi-Newton* methods. At each iteration, the search takes a step \mathbf{s} defined by solving $\mathbf{Hs} = -\mathbf{g}$ for \mathbf{s} , where \mathbf{s} is the step, \mathbf{g} is the *gradient* vector of the function's partial first derivatives with respect to the parameters, and \mathbf{H} is an approximation to the Hessian matrix of partial second derivatives formed by repeated rank 1 updates at each iteration. Two well-known update formulae are called Davidon–Fletcher–Powell (DFP) and Broyden–Fletcher–Goldfarb–Shanno (BFGS). Loss of the positive-definite condition through round-off error is avoided if rather than updating an approximation to the inverse of the Hessian, one updates the Cholesky decomposition of an approximation of the Hessian matrix itself.

Monte Carlo simulation

The Cholesky decomposition is commonly used in the Monte Carlo method for simulating systems with multiple correlated variables: The correlation matrix is decomposed, to give the lower-triangular \mathbf{L} . Applying this to a vector of uncorrelated samples, \mathbf{u} , produces a sample vector \mathbf{Lu} with the covariance properties of the system being modeled.^[6]

For a simplified example that shows the economy one gets from Cholesky's decomposition, say one needs to generate two correlated normal variables x_1 and x_2 . All one needs to do is generate two uncorrelated Gaussian random variables z_1 and z_2 . We set $x_1 = z_1$ and $x_2 = \rho z_1 + \sqrt{1 - \rho^2} z_2$.

Kalman filters

Unscented Kalman filters commonly use the Cholesky decomposition to choose a set of so-called sigma points. The Kalman filter tracks the average state of a system as a vector \mathbf{x} of length N and covariance as an N -by- N matrix \mathbf{P} . The matrix \mathbf{P} is always positive semi-definite, and can be decomposed into $\mathbf{L}\mathbf{L}^T$. The columns of \mathbf{L} can be added and subtracted from the mean \mathbf{x} to form a set of $2N$ vectors called *sigma points*. These sigma points completely capture the mean and covariance of the system state.

Matrix inversion

The explicit inverse of a Hermitian matrix can be computed via Cholesky decomposition, in a manner similar to solving linear systems, using n^3 operations ($\frac{1}{2}n^3$ multiplications).^[5] The entire inversion can even be efficiently performed in-place.

A non-Hermitian matrix \mathbf{B} can also be inverted using the following identity, where $\mathbf{B}\mathbf{B}^*$ will always be Hermitian:

$$\mathbf{B}^{-1} = \mathbf{B}^*(\mathbf{B}\mathbf{B}^*)^{-1}.$$

Computation

There are various methods for calculating the Cholesky decomposition. The computational complexity of commonly used algorithms is $O(n^3)$ in general. The algorithms described below all involve about $n^3/3$ FLOPs, where n is the size of the matrix \mathbf{A} . Hence, they are half the cost of the LU decomposition, which uses $2n^3/3$ FLOPs (see Trefethen and Bau 1997).

Which of the algorithms below is faster depends on the details of the implementation. Generally, the first algorithm will be slightly slower because it accesses the data in a less regular manner.

The Cholesky algorithm

The **Cholesky algorithm**, used to calculate the decomposition matrix L , is a modified version of Gaussian elimination.

The recursive algorithm starts with $i := 1$ and

$$\mathbf{A}^{(1)} := \mathbf{A}.$$

At step i , the matrix $\mathbf{A}^{(i)}$ has the following form:

$$\mathbf{A}^{(i)} = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & a_{i,i} & \mathbf{b}_i^* \\ 0 & \mathbf{b}_i & \mathbf{B}^{(i)} \end{pmatrix},$$

where \mathbf{I}_{i-1} denotes the identity matrix of dimension $i - 1$.

If we now define the matrix \mathbf{L}_i by

For complex Hermitian matrix, the following formula applies:

$$L_{j,j} = \sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k} L_{j,k}^*}.$$

$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k}^* \right), \quad \text{for } i > j.$$

So we can compute the (i, j) entry if we know the entries to the left and above. The computation is usually arranged in either of the following orders.

- The **Cholesky–Banachiewicz algorithm** starts from the upper left corner of the matrix L and proceeds to calculate the matrix row by row.
- The **Cholesky–Crout algorithm** starts from the upper left corner of the matrix L and proceeds to calculate the matrix column by column.

Either pattern of access allows the entire computation to be performed in-place if desired.

Stability of the computation

Suppose that we want to solve a well-conditioned system of linear equations. If the LU decomposition is used, then the algorithm is unstable unless we use some sort of pivoting strategy. In the latter case, the error depends on the so-called growth factor of the matrix, which is usually (but not always) small.

Now, suppose that the Cholesky decomposition is applicable. As mentioned above, the algorithm will be twice as fast. Furthermore, no pivoting is necessary and the error will always be small. Specifically, if we want to solve $\mathbf{Ax} = \mathbf{b}$, and \mathbf{y} denotes the computed solution, then \mathbf{y} solves the disturbed system $(\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b}$ where

$$\|\mathbf{E}\|_2 \leq c_n \varepsilon \|\mathbf{A}\|_2.$$

Here, $\|\cdot\|_2$ is the matrix 2-norm, c_n is a small constant depending on n , and ε denotes the unit round-off.

One concern with the Cholesky decomposition to be aware of is the use of square roots. If the matrix being factorized is positive definite as required, the numbers under the square roots are always positive *in exact arithmetic*. Unfortunately, the numbers can become negative because of round-off errors, in which case the algorithm cannot continue. However, this can only happen if the matrix is very ill-conditioned. One way to address this is to add a diagonal correction matrix to the matrix being decomposed in an attempt to promote the positive-definiteness.^[7] While this might lessen the accuracy of the decomposition, it can be very favorable for other reasons; for example, when performing Newton's method in optimization, adding a diagonal matrix can improve stability when far from the optimum.

LDL decomposition

An alternative form, eliminating the need to take square roots, is the symmetric indefinite factorization^[8]

$$\begin{aligned} \mathbf{A} = \mathbf{LDL}^T &= \begin{pmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{pmatrix} \begin{pmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{pmatrix} \begin{pmatrix} 1 & L_{21} & L_{31} \\ 0 & 1 & L_{32} \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} D_1 & & \\ L_{21}D_1 & L_{21}^2D_1 + D_2 & \\ L_{31}D_1 & L_{31}L_{21}D_1 + L_{32}D_2 & L_{31}^2D_1 + L_{32}^2D_2 + D_3 \end{pmatrix} \quad (\text{symmetric}) \end{aligned}$$

If \mathbf{A} is real, the following recursive relations apply for the entries of \mathbf{D} and \mathbf{L} :

$$\begin{aligned} D_j &= A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 D_k \\ L_{ij} &= \frac{1}{D_j} \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} D_k \right), \quad \text{for } i > j. \end{aligned}$$

For complex Hermitian matrix \mathbf{A} , the following formula applies:

$$\begin{aligned} D_j &= A_{jj} - \sum_{k=1}^{j-1} L_{jk} L_{jk}^* D_k \\ L_{ij} &= \frac{1}{D_j} \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk}^* D_k \right), \quad \text{for } i > j. \end{aligned}$$

Again, the pattern of access allows the entire computation to be performed in-place if desired.

Block variant

When used on indefinite matrices, the \mathbf{LDL}^* factorization is known to be unstable without careful pivoting;^[9] specifically, the elements of the factorization can grow arbitrarily. A possible improvement is to perform the factorization on block sub-matrices, commonly 2×2 :^[10]

$$\begin{aligned} \mathbf{A} = \mathbf{LDL}^T &= \begin{pmatrix} \mathbf{I} & 0 & 0 \\ \mathbf{L}_{21} & \mathbf{I} & 0 \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{D}_1 & 0 & 0 \\ 0 & \mathbf{D}_2 & 0 \\ 0 & 0 & \mathbf{D}_3 \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L}_{21}^T & \mathbf{L}_{31}^T \\ 0 & \mathbf{I} & \mathbf{L}_{32}^T \\ 0 & 0 & \mathbf{I} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{D}_1 & & \\ \mathbf{L}_{21}\mathbf{D}_1 & \mathbf{L}_{21}\mathbf{D}_1\mathbf{L}_{21}^T + \mathbf{D}_2 & \\ \mathbf{L}_{31}\mathbf{D}_1 & \mathbf{L}_{31}\mathbf{D}_1\mathbf{L}_{21}^T + \mathbf{L}_{32}\mathbf{D}_2 & \mathbf{L}_{31}\mathbf{D}_1\mathbf{L}_{31}^T + \mathbf{L}_{32}\mathbf{D}_2\mathbf{L}_{32}^T + \mathbf{D}_3 \end{pmatrix} \quad (\text{symmetric}) \end{aligned}$$

where every element in the matrices above is a square submatrix. From this, these analogous recursive relations follow:

$$D_j = A_{jj} - \sum_{k=1}^{j-1} L_{jk} D_k L_{jk}^T$$

$$L_{ij} = \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} D_k L_{jk}^T \right) D_j^{-1}$$

Note the presence of matrix products and explicit inversion, this limits the practical block size.

Updating the decomposition

A task that often arises in practice is that one needs to update a Cholesky decomposition. In more details, one has already computed the Cholesky decomposition $\mathbf{A} = \mathbf{L}\mathbf{L}^*$ of some matrix \mathbf{A} , then one changes the matrix \mathbf{A} in some way into another matrix, say $\tilde{\mathbf{A}}$, and one wants to compute the Cholesky decomposition of the updated matrix: $\tilde{\mathbf{A}} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^*$. The question is now whether one can use the Cholesky decomposition of \mathbf{A} that was computed before to compute the Cholesky decomposition of $\tilde{\mathbf{A}}$.

Rank-one update

The specific case, where the updated matrix $\tilde{\mathbf{A}}$ is related to the matrix \mathbf{A} by $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{x}\mathbf{x}^*$, is known as a *rank-one update*.

Here is a little function based on ^[11] written in Matlab syntax which realizes a rank-one update:

```
function [L] = cholupdate(L,x)
    p = length(x);
    for k=1:p
        r = sqrt(L(k,k)^2 + x(k)^2);
        c = r / L(k, k);
        s = x(k) / L(k, k);
        L(k, k) = r;
        L(k+1:p,k) = (L(k+1:p,k) + s*x(k+1:p)) / c;
        x(k+1:p) = c*x(k+1:p) - s*L(k+1:p,k);
    end
end
```

Rank-one downdate

A *rank-one downdate* is similar to a rank-one update, except that the addition is replaced by subtraction: $\tilde{\mathbf{A}} = \mathbf{A} - \mathbf{x}\mathbf{x}^*$. This only works if the new matrix $\tilde{\mathbf{A}}$ is still positive definite.

The code for the rank-one update shown above can easily be adapted to do a rank-one downdate: one merely needs to replace the two additions in the assignment to r and $L(k+1:p,k)$ by subtractions.

Proof for positive semi-definite matrices

The above algorithms show that every positive definite matrix \mathbf{A} has a Cholesky decomposition. This result can be extended to the positive semi-definite case by a limiting argument. The argument is not fully constructive, i.e., it gives no explicit numerical algorithms for computing Cholesky factors.

If \mathbf{A} is an n -by- n positive semi-definite matrix, then the sequence $\{\mathbf{A}_k\} = \{\mathbf{A} + (1/k)\mathbf{I}_n\}$ consists of positive definite matrices. (This is an immediate consequence of, for example, the spectral mapping theorem for the polynomial functional calculus.) Also,

$$\mathbf{A}_k \rightarrow \mathbf{A}$$

in operator norm. From the positive definite case, each \mathbf{A}_k has Cholesky decomposition $\mathbf{A}_k = \mathbf{L}_k \mathbf{L}_k^*$. By property of the operator norm,

$$\|\mathbf{L}_k\|^2 \leq \|\mathbf{L}_k \mathbf{L}_k^*\| = \|\mathbf{A}_k\|.$$

So $\{\mathbf{L}_k\}$ is a bounded set in the Banach space of operators, therefore relatively compact (because the underlying vector space is finite-dimensional). Consequently it has a convergent subsequence, also denoted by $\{\mathbf{L}_k\}$, with limit \mathbf{L} . It can be easily checked that this \mathbf{L} has the desired properties, i.e. $\mathbf{A} = \mathbf{L} \mathbf{L}^*$ and \mathbf{L} is lower triangular with non-negative diagonal entries: for all x and y ,

$$\langle \mathbf{A}x, y \rangle = \langle \lim \mathbf{A}_k x, y \rangle = \langle \lim \mathbf{L}_k \mathbf{L}_k^* x, y \rangle = \langle \mathbf{L} \mathbf{L}^* x, y \rangle.$$

Therefore $\mathbf{A} = \mathbf{L} \mathbf{L}^*$. Because the underlying vector space is finite-dimensional, all topologies on the space of operators are equivalent. So \mathbf{L}_k tends to \mathbf{L} in norm means \mathbf{L}_k tends to \mathbf{L} entrywise. This in turn implies that, since each \mathbf{L}_k is lower triangular with non-negative diagonal entries, \mathbf{L} is also.

Generalization

The Cholesky factorization can be generalized to (not necessarily finite) matrices with operator entries. Let $\{\mathcal{H}_n\}$ be a sequence of Hilbert spaces. Consider the operator matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & & \\ \mathbf{A}_{12}^* & \mathbf{A}_{22} & \mathbf{A}_{23} & & \\ \mathbf{A}_{13}^* & \mathbf{A}_{23}^* & \mathbf{A}_{33} & & \\ & & & \ddots & \end{bmatrix}$$

acting on the direct sum

$$\mathcal{H} = \oplus_n \mathcal{H}_n,$$

where each

$$\mathbf{A}_{ij} : \mathcal{H}_j \rightarrow \mathcal{H}_i$$

is a bounded operator. If \mathbf{A} is positive (semidefinite) in the sense that for all finite k and for any

$$h \in \bigoplus_{n=1}^k \mathcal{H}_k,$$

we have $\langle h, \mathbf{A}h \rangle \geq 0$, then there exists a lower triangular operator matrix \mathbf{L} such that $\mathbf{A} = \mathbf{L}\mathbf{L}^*$. One can also take the diagonal entries of \mathbf{L} to be positive.

Implementations in programming languages

In Python, the command "cholesky" from the `numpy.linalg` module performs Cholesky decomposition. In Matlab Programming, the "chol" command can be used to simply apply this to a matrix. In R the "chol" gives Cholesky decomposition.

See also

- Symbolic Cholesky decomposition
- Minimum degree algorithm
- Matrix decomposition
- Sylvester's law of inertia
- Cycle rank
- Matrix square root

Notes

1. Press, William H.; Saul A. Teukolsky; William T. Vetterling; Brian P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing (second edition)* (<http://www.nr.com/>). Cambridge University England EPress. p. 994. ISBN 0-521-43108-5.
2. Golub & Van Loan (1996, p. 143), Horn & Johnson (1985, p. 407), Trefethen & Bau (1997, p. 174)
3. Golub & Van Loan (1996, p. 147)
4. Horn & Johnson (1985, p. 407)
5. Krishnamoorthy, Aravindh; Menon, Deepak (2011). "Matrix Inversion Using Cholesky Decomposition" **1111**. p. 4144. arXiv:1111.4144 (<https://arxiv.org/abs/1111.4144>). Bibcode:2011arXiv1111.4144K (<http://adsabs.harvard.edu/abs/2011arXiv1111.4144K>).
6. Matlab randn documentaion (<http://www.mathworks.com/help/techdoc/ref/randn.html>).
7. Fang, Haw-ren; O'Leary, Dianne P. (8 August 2006). "Modified Cholesky Algorithms: A Catalog with New Approaches" (<http://www.cs.umd.edu/~oleary/tr/tr4807.pdf>) (PDF).
8. Watkins, D. (1991). *Fundamentals of Matrix Computations*. New York: Wiley. p. 84. ISBN 0-471-61414-9.
9. Nocedal, Jorge (2000). *Numerical Optimization*. Springer.
10. Fang, Haw-ren (24 August 2007). "Analysis of Block LDLT Factorizations for Symmetric Indefinite Matrices".
11. Stewart, G. W. (1998). *Basic decompositions*. Philadelphia: Soc. for Industrial and Applied Mathematics. ISBN 0-89871-414-1.

References

- Dereniowski, Dariusz; Kubale, Marek (2004). "5th International Conference on Parallel Processing and Applied Mathematics" (http://www.eti.pg.gda.pl/katedry/kams/wwwkams/pdf/Cholesky_fmprg.pdf) (PDF). Lecture Notes on Computer Science **3019**. Springer-Verlag. pp. 985–992. doi:10.1007/978-3-540-24669-5_127 (https://dx.doi.org/10.1007%2F978-3-540-24669-5_127). ISBN 978-3-540-21946-0 |chapter= ignored (help).

- Golub, Gene H.; Van Loan, Charles F. (1996). *Matrix Computations* (3rd ed.). Baltimore: Johns Hopkins. ISBN 978-0-8018-5414-9{{inconsistent citations}}.
- Horn, Roger A.; Johnson, Charles R. (1985). *Matrix Analysis*. Cambridge University Press. ISBN 0-521-38632-2{{inconsistent citations}}.
- S. J. Julier and J. K. Uhlmann. "A General Method for Approximating Nonlinear Transformations of ProbabilityDistributions".
- S. J. Julier and J.K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in Proc. AeroSense: 11th Int. Symp. Aerospace/Defence Sensing, Simulation and Controls, 1997, pp. 182–193.
- Trefethen, Lloyd N.; Bau, David (1997). *Numerical linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics. ISBN 978-0-89871-361-9{{inconsistent citations}}.

External links

History of science

- *Sur la résolution numérique des systèmes d'équations linéaires*, Cholesky's 1910 manuscript, online and analyzed on BibNum (<http://bibnum.education.fr/mathematiques/algebre/sur-la-resolution-numerique-des-systemes-d-equations-lineaires>) **(French)** **(English)** [for English, click 'A télécharger']

Information

- Hazewinkel, Michiel, ed. (2001), "Cholesky factorization" (<http://www.encyclopediaofmath.org/index.php?title=p/c120160>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Cholesky Decomposition (<http://planetmath.org/?op=getobj&from=objects&id=1287>) at PlanetMath.org.
- Cholesky Decomposition (<http://rkb.home.cern.ch/rkb/AN16pp/node33.html#SECTION00033000000000000000>), The Data Analysis BriefBook
- Module for Cholesky Factorization (<http://math.fullerton.edu/mathews/n2003/CholeskyMod.html>)
- Cholesky Decomposition (<http://www.math-linux.com/spip.php?article43>) on www.math-linux.com
- Cholesky Decomposition Made Simple (<http://sciencemeanderthal.wordpress.com/2012/06/28/cholesky-decomposition-of-variance-covariance-matrices-in-the-classic-twin-study/>) on Science Meanderthal

Computer code

- LAPACK (<http://netlib.org/lapack/>) is a collection of FORTRAN subroutines for solving dense linear algebra problems
- ALGLIB (<http://www.alglib.net/>) includes a partial port of the LAPACK to C++, C#, Delphi, Visual Basic, etc.
- libflame (<http://www.cs.utexas.edu/users/flame/>) is a C library with LAPACK functionality.
- Notes and video on high-performance implementation of Cholesky factorization (<http://www.cs.utexas.edu/users/flame/Movies.html#Chol>) at The University of Texas at Austin.
- Cholesky : TBB + Threads + SSE (<http://upcommons.upc.edu/pfc/handle/2099.1/10988/>) is a book explaining the implementation of the CF with TBB,threads and SSE (in Spanish).
- library "Ceres Solver" (<https://code.google.com/p/ceres-solver/>) by Google.
- LDL decomposition (<http://infohost.nmt.edu/~borchers/ldlt.html>) routines in Matlab.

Use of the matrix in simulation

- Simulation of correlated normal random variables (<http://approximity.com/papers/ptfopt/node19.html>)
- Generating Correlated Random Variables (http://www.columbia.edu/~mh2078/MCS04/MCS_framework_FEegs.pdf), Martin Haugh, Columbia University

Online calculators

- Online Matrix Calculator (<http://www.bluebit.gr/matrix-calculator/>) Performs Cholesky decomposition of matrices online.

Retrieved from "http://en.wikipedia.org/w/index.php?title=Cholesky_decomposition&oldid=662340960"

Categories: Operator theory | Matrix decompositions | Numerical linear algebra

- This page was last modified on 14 May 2015, at 18:56.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.