WIKIPEDIA
The Free Encyclopedia
WIKIPEDIA

Cholesky decomposition

In linear algebra, the **Cholesky decomposition** or **Cholesky factorization** (pronounced /ʃəˈlɛski/ *shə-LES-kee*) is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, which is useful for efficient numerical solutions, e.g., Monte Carlo simulations. It was discovered by André-Louis Cholesky for real matrices, and posthumously published in 1924.[1] When it is applicable, the Cholesky decomposition is roughly twice as efficient as the LU decomposition for solving systems of linear equations.[2]

Statement

The Cholesky decomposition of a Hermitian positive-definite matrix $\mathbf{A}$, is a decomposition of the form

$$\mathbf{A} = \mathbf{LL}^*,$$

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.[3]

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

When $\mathbf{A}$ is a real matrix (hence symmetric positive-definite), the factorization may be written

$$\mathbf{A} = \mathbf{LL}^\mathsf{T},$$

where $\mathbf{L}$ is a real lower triangular matrix with positive diagonal entries.[4][5][6]

Positive semidefinite matrices

If a Hermitian matrix $\mathbf{A}$ is only positive semidefinite, instead of positive definite, then it still has a decomposition of the form $\mathbf{A} = \mathbf{LL}^*$ where the diagonal entries of $\mathbf{L}$ are allowed to be zero.[7] The decomposition need not be unique, for example:

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{LL}^*, \qquad \mathbf{L} = \begin{bmatrix} 0 & 0 \\ \cos\theta & \sin\theta \end{bmatrix},$$

for any θ. However, if the rank of $\mathbf{A}$ is r, then there is a unique lower triangular $\mathbf{L}$ with exactly r positive diagonal elements and $n - r$ columns containing all zeroes.[8]

Alternatively, the decomposition can be made unique when a pivoting choice is fixed. Formally, if $\mathbf{A}$ is an $n \times n$ positive semidefinite matrix of rank r, then there is at least one permutation matrix $\mathbf{P}$ such that $\mathbf{P}\ \mathbf{A}\ \mathbf{P}^{\mathrm{T}}$ has a unique decomposition of the form $\mathbf{P}\ \mathbf{A}\ \mathbf{P}^{\mathrm{T}} = \mathbf{L}\ \mathbf{L}^*$ with $\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & 0 \\ \mathbf{L}_2 & 0 \end{bmatrix}$, where $\mathbf{L}_1$ is an $r \times r$ lower triangular matrix with positive diagonal.[9]

LDL decomposition

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

$$\mathbf{A} = \mathbf{LDL}^*,$$

where $\mathbf{L}$ is a lower unit triangular (unitriangular) matrix, and $\mathbf{D}$ is a diagonal matrix. That is, the diagonal elements of $\mathbf{L}$ are required to be 1 at the cost of introducing an additional diagonal matrix $\mathbf{D}$ in the decomposition. The main advantage is that the LDL decomposition can be computed and used with essentially the same algorithms, but avoids extracting square roots.[10]

For this reason, the LDL decomposition is often called the *square-root-free Cholesky* decomposition. For real matrices, the factorization has the form $\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$ and is often referred to as **LDLT decomposition** (or $\mathbf{LDL}^{\mathbf{T}}$ decomposition, or **LDL′**). It is reminiscent of the eigendecomposition of real symmetric matrices, $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}$, but is quite different in practice because $\mathbf{\Lambda}$ and $\mathbf{D}$ are not similar matrices.

The LDL decomposition is related to the classical Cholesky decomposition of the form $\mathbf{LL}^*$ as follows:

$$\mathbf{A} = \mathbf{LDL}^* = \mathbf{L}\mathbf{D}^{1/2}\left(\mathbf{D}^{1/2}\right)^*\mathbf{L}^* = \mathbf{L}\mathbf{D}^{1/2}\left(\mathbf{L}\mathbf{D}^{1/2}\right)^*.$$

Conversely, given the classical Cholesky decomposition $\mathbf{A} = \mathbf{CC}^*$ of a positive definite matrix, if $\mathbf{S}$ is a diagonal matrix that contains the main diagonal of $\mathbf{C}$, then $\mathbf{A}$ can be decomposed as $\mathbf{LDL}^*$ where

$$\mathbf{L} = \mathbf{CS}^{-1}$$

(this rescales each column to make diagonal elements 1),

$$\mathbf{D} = \mathbf{S}^2.$$

If $\mathbf{A}$ is positive definite then the diagonal elements of $\mathbf{D}$ are all positive. For positive semidefinite $\mathbf{A}$, an $\mathbf{LDL}^*$ decomposition exists where the number of non-zero elements on the diagonal $\mathbf{D}$ is exactly the rank of $\mathbf{A}$.[11] Some indefinite matrices for which no Cholesky decomposition exists have an LDL decomposition with negative entries in $\mathbf{D}$: it suffices that the first $n - 1$ leading principal minors of $\mathbf{A}$ are non-singular.[12]

Example

Here is the Cholesky decomposition of a symmetric real matrix:

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix}.$$

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

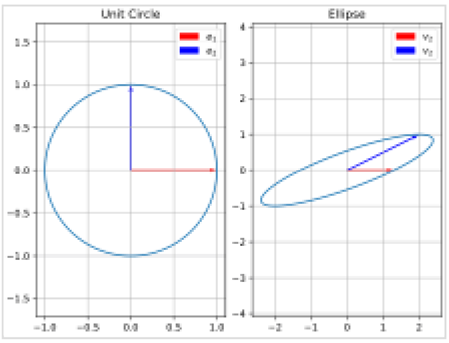
$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -4 & 5 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 9 \end{pmatrix} \begin{pmatrix} 1 & 3 & -4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}.$$

Geometric interpretation

The Cholesky decomposition is equivalent to a particular choice of conjugate axes of an ellipsoid.[13] In detail, let the ellipsoid be defined as $y^T Ay = 1$, then by definition, a set of vectors $v_1, \dots, v_n$ are conjugate axes of the ellipsoid iff $v_i^T Av_j = \delta_{ij}$. Then, the ellipsoid is precisely

$$\left\{ \sum_i x_i v_i : x^T x = 1 \right\} = f(\mathbb{S}^n)$$

where f maps the basis vector $e_i \mapsto v_i$, and $\mathbb{S}^n$ is the unit sphere in n dimensions. That is, the ellipsoid is a linear image of the unit sphere.



The ellipse is a linear image of the unit circle. The two vectors v_1, v_2 are conjugate axes of the ellipse chosen such that v_1 is parallel to the first axis and v_2 is within the plane spanned by the first two axes.

Define the matrix $V := [v_1|v_2|\cdots|v_n]$, then $v_i^T Av_j = \delta_{ij}$ is equivalent to $V^T AV = I$. Different choices of the conjugate axes correspond to different decompositions.

The Cholesky decomposition corresponds to choosing v_1 to be parallel to the first axis, v_2 to be within the plane spanned by the first two axes, and so on. This makes V an upper-triangular matrix. Then, there is $A = LL^T$, where $L = (V^{-1})^T$ is lower-triangular.

Similarly, principal component analysis corresponds to choosing $v_1, \dots, v_n$ to be perpendicular. Then, let $\lambda = 1/\|v_i\|^2$ and $\Sigma = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$, and there is $V = U\Sigma^{-1/2}$ where U is an orthogonal matrix. This then yields $A = U\Sigma U^T$.

Applications

Numerical solution of system of linear equations

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 $\mathbf{Ax} = \mathbf{b}$ can be solved 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.

An alternative way to eliminate taking square roots in the $\mathbf{LL}^*$ decomposition is to compute the LDL decomposition $\mathbf{A} = \mathbf{LDL}^*$, then solving $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$, and finally solving $\mathbf{DL}^*\mathbf{x} = \mathbf{y}$.

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.[2]

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 iteration k, the search steps in a direction p_k defined by solving $B_k p_k = -g_k$ for p_k, where p_k is the step direction, g_k is the gradient, and B_k is an approximation to the Hessian matrix formed by repeating rank-1 updates at each iteration. Two well-known update formulas 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.[14]

Monte Carlo simulation

The Cholesky decomposition is commonly used in the Monte Carlo method for simulating systems with multiple correlated variables. The covariance matrix is decomposed to give the lower-triangular $\mathbf{L}$. Applying this to a vector of uncorrelated observations in a sample $\mathbf{u}$ produces a sample vector $\mathbf{Lu}$ with the covariance properties of the system being modeled.[15]

The following simplified example shows the economy one gets from the Cholesky decomposition: suppose the goal is to generate two correlated normal variables x_1 and x_2 with given correlation coefficient ρ. To accomplish that, it is necessary to first generate two uncorrelated Gaussian random variables z_1 and z_2 (for example, via a Box–Muller transform). Given the required correlation coefficient ρ, the correlated normal variables can be obtained via the transformations $x_1 = z_1$ 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 $N \times N$ matrix $\mathbf{P}$. The matrix $\mathbf{P}$ is always positive semi-definite and can be decomposed into $\mathbf{LL}^{\mathrm{T}}$. 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 by Cholesky decomposition, in a manner similar to solving linear systems, using n^3 operations ($\tfrac{1}{2}n^3$ multiplications).[10] 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 $\mathbf{BB}^*$ will always be Hermitian:

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

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 $(1/3)n^3$ FLOPs ($n^3/6$ multiplications and the same number of additions) for real flavors and $(4/3)n^3$ FLOPs for complex flavors,[16] where n is the size of the matrix $\mathbf{A}$. Hence, they have 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 $\mathbf{L}$, 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 the matrix $\mathbf{L}_i$ is defined by

$$\mathbf{L}_i := \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & \sqrt{a_{i,i}} & 0 \\ 0 & \frac{1}{\sqrt{a_{i,i}}}\mathbf{b}_i & \mathbf{I}_{n-i} \end{pmatrix},$$

(note that $a_{i,i} > 0$ since $\mathbf{A}^{(i)}$ is positive definite), then $\mathbf{A}^{(i)}$ can be written as

$$\mathbf{A}^{(i)} = \mathbf{L}_i \mathbf{A}^{(i+1)} \mathbf{L}_i^*$$

where

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

Note that $\mathbf{b}_i \mathbf{b}^*{}_i$ is an outer product, therefore this algorithm is called the *outer-product version* in (Golub & Van Loan).

This is repeated for i from 1 to n. After n steps, $\mathbf{A}^{(n+1)} = \mathbf{I}$ is obtained, and hence, the lower triangular matrix L sought for is calculated as

$$\mathbf{L} := \mathbf{L}_1 \mathbf{L}_2 \dots \mathbf{L}_n.$$

The Cholesky–Banachiewicz and Cholesky–Crout algorithms

Access pattern (white) and writing pattern (yellow) for the in-place Cholesky—Banachiewicz algorithm on a 5×5 matrix

If the equation

$$\begin{aligned} \mathbf{A} = \mathbf{LL}^T &= \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix} \\ &= \begin{pmatrix} L_{11}^2 & & (\text{symmetric}) \\ L_{21}L_{11} & L_{21}^2 + L_{22}^2 & \\ L_{31}L_{11} & L_{31}L_{21} + L_{32}L_{22} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{pmatrix}, \end{aligned}$$

is written out, the following is obtained:

$$\mathbf{L} = \begin{pmatrix} \sqrt{A_{11}} & 0 & 0 \\ A_{21}/L_{11} & \sqrt{A_{22} - L_{21}^2} & 0 \\ A_{31}/L_{11} & \left(A_{32} - L_{31}L_{21}\right)/L_{22} & \sqrt{A_{33} - L_{31}^2 - L_{32}^2} \end{pmatrix}$$

and therefore the following formulas for the entries of $\mathbf{L}$:

$$L_{j,j} = (\pm)\sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2},$$

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

For complex and real matrices, inconsequential arbitrary sign changes of diagonal and associated off-diagonal elements are allowed. The expression under the square root is always positive if $\mathbf{A}$ is real and positive-definite.

For complex Hermitian matrix, the following formula applies:

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

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

So it now is possible to compute the (i, j) entry if the entries to the left and above are known. 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.

```
for (i = 0; i < dimensionSize; i++) {
    for (j = 0; j <= i; j++) {
        float sum = 0;
        for (k = 0; k < j; k++)
            sum += L[i][k] * L[j][k];

        if (i == j)
            L[i][j] = sqrt(A[i][i] - sum);
        else
            L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

The above algorithm can be succinctly expressed as combining a dot product and matrix multiplication in vectorized programming languages such as Fortran as the following,

```
do i = 1, size(A,1)
    L(i,i) = sqrt(A(i,i) - dot_product(L(i,1:i-1), L(i,1:i-1)))
    L(i+1:,i) = (A(i+1:,i) - matmul(conjg(L(i,1:i-1)), L(i+1:,1:i-1))) / L(i,i)
end do
```

where `conjg` refers to complex conjugate of the elements.

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

```
for (j = 0; j < dimensionSize; j++) {
    float sum = 0;
    for (k = 0; k < j; k++) {
        sum += L[j][k] * L[j][k];
    }
    L[j][j] = sqrt(A[j][j] - sum);

    for (i = j + 1; i < dimensionSize; i++) {
        sum = 0;
        for (k = 0; k < j; k++) {
            sum += L[i][k] * L[j][k];
        }
        L[i][j] = (1.0 / L[j][j] * (A[i][j] - sum));
    }
}
```

The above algorithm can be succinctly expressed as combining a dot product and matrix multiplication in vectorized programming languages such as Fortran as the following,

```
do i = 1, size(A,1)
    L(i,i) = sqrt(A(i,i) - dot_product(L(1:i-1,i), L(1:i-1,i)))
    L(i,i+1:) = (A(i,i+1:) - matmul(conjg(L(1:i-1,i)), L(1:i-1,i+1:))) / L(i,i)
end do
```

where `conjg` refers to complex conjugate of the elements.

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

Stability of the computation

Suppose that there is a desire to solve a well-conditioned system of linear equations. If the LU decomposition is used, then the algorithm is unstable unless some sort of pivoting strategy is used. 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 $\mathbf{Ax} = \mathbf{b}$, and $\mathbf{y}$ denotes the computed solution, then $\mathbf{y}$ solves the perturbed system $(\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b}$, where

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

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.[17] 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 when $\mathbf{A}$ is symmetric, is the symmetric indefinite factorization[18]

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

The following recursive relations apply for the entries of $\mathbf{D}$ and $\mathbf{L}$:

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

This works as long as the generated diagonal elements in $\mathbf{D}$ stay non-zero. The decomposition is then unique. $\mathbf{D}$ and $\mathbf{L}$ are real if $\mathbf{A}$ is real.

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;[19] specifically, the elements of the factorization can grow arbitrarily. A possible improvement is to perform the factorization on block sub-matrices, commonly 2 × 2:[20]

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

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

$$\mathbf{D}_j = \mathbf{A}_{jj} - \sum_{k=1}^{j-1} \mathbf{L}_{jk}\mathbf{D}_k\mathbf{L}_{jk}^{\mathrm{T}},$$

$$\mathbf{L}_{ij} = \left(\mathbf{A}_{ij} - \sum_{k=1}^{j-1} \mathbf{L}_{ik}\mathbf{D}_k\mathbf{L}_{jk}^{\mathrm{T}}\right)\mathbf{D}_j^{-1}.$$

This involves matrix products and explicit inversion, thus limiting 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 function[21] written in Matlab syntax that realizes a rank-one update:

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

A *rank-n update* is one where for a matrix $\mathbf{M}$ one updates the decomposition such that $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{M}\mathbf{M}^*$. This can be achieved by successively performing rank-one updates for each of the columns of $\mathbf{M}$.

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):n, k)` by subtractions.

Adding and removing rows and columns

If a symmetric and positive definite matrix $\mathbf{A}$ is represented in block form as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{13} \\ \mathbf{A}_{13}^{\mathrm{T}} & \mathbf{A}_{33} \end{pmatrix}$$

and its upper Cholesky factor

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{13} \\ 0 & \mathbf{L}_{33} \end{pmatrix},$$

then for a new matrix $\tilde{\mathbf{A}}$, which is the same as $\mathbf{A}$ but with the insertion of new rows and columns,

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{12}^{\mathrm{T}} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{13}^{\mathrm{T}} & \mathbf{A}_{23}^{\mathrm{T}} & \mathbf{A}_{33} \end{pmatrix}$$

Now there is an interest in finding the Cholesky factorization of $\tilde{\mathbf{A}}$, which can be called $\tilde{\mathbf{S}}$, without directly computing the entire decomposition.

$$\tilde{\mathbf{S}} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} \\ 0 & \mathbf{S}_{22} & \mathbf{S}_{23} \\ 0 & 0 & \mathbf{S}_{33} \end{pmatrix}.$$

Writing $\mathbf{A} \setminus \mathbf{b}$ for the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$, which can be found easily for triangular matrices, and $\operatorname{chol}(\mathbf{M})$ for the Cholesky decomposition of $\mathbf{M}$, the following relations can be found:

$$\begin{aligned} \mathbf{S}_{11} &= \mathbf{L}_{11}, \\ \mathbf{S}_{12} &= \mathbf{L}_{11}^{\mathrm{T}} \setminus \mathbf{A}_{12}, \\ \mathbf{S}_{13} &= \mathbf{L}_{13}, \\ \mathbf{S}_{22} &= \operatorname{chol}\left(\mathbf{A}_{22} - \mathbf{S}_{12}^{\mathrm{T}}\mathbf{S}_{12}\right), \\ \mathbf{S}_{23} &= \mathbf{S}_{22}^{\mathrm{T}} \setminus \left(\mathbf{A}_{23} - \mathbf{S}_{12}^{\mathrm{T}}\mathbf{S}_{13}\right), \\ \mathbf{S}_{33} &= \operatorname{chol}\left(\mathbf{L}_{33}^{\mathrm{T}}\mathbf{L}_{33} - \mathbf{S}_{23}^{\mathrm{T}}\mathbf{S}_{23}\right). \end{aligned}$$

These formulas may be used to determine the Cholesky factor after the insertion of rows or columns in any position, if the row and column dimensions are appropriately set (including to zero). The inverse problem,

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{12}^{\mathrm{T}} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{13}^{\mathrm{T}} & \mathbf{A}_{23}^{\mathrm{T}} & \mathbf{A}_{33} \end{pmatrix}$$

with known Cholesky decomposition

$$\tilde{\mathbf{S}} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} \\ 0 & \mathbf{S}_{22} & \mathbf{S}_{23} \\ 0 & 0 & \mathbf{S}_{33} \end{pmatrix}$$

and the desire to determine the Cholesky factor

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{13} \\ 0 & \mathbf{L}_{33} \end{pmatrix}$$

of the matrix $\mathbf{A}$ with rows and columns removed,

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{13} \\ \mathbf{A}_{13}^{\mathrm{T}} & \mathbf{A}_{33} \end{pmatrix},$$

yields the following rules:

$$\begin{aligned} \mathbf{L}_{11} &= \mathbf{S}_{11}, \\ \mathbf{L}_{13} &= \mathbf{S}_{13}, \\ \mathbf{L}_{33} &= \operatorname{chol}\left(\mathbf{S}_{33}^{\mathrm{T}}\mathbf{S}_{33} + \mathbf{S}_{23}^{\mathrm{T}}\mathbf{S}_{23}\right). \end{aligned}$$

Notice that the equations above that involve finding the Cholesky decomposition of a new matrix are all of the form $\tilde{\mathbf{A}} = \mathbf{A} \pm \mathbf{x}\mathbf{x}^*$, which allows them to be efficiently calculated using the update and downdate procedures detailed in the previous section.[22]

Proof for positive semi-definite matrices

Proof by limiting argument

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

$$\mathbf{A}_k \to \mathbf{A} \quad \text{for} \quad k \to \infty$$

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,

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

The $\leq$ holds because $M_n(\mathbb{C})$ equipped with the operator norm is a C* algebra. So $(\mathbf{L}_k)_k$ 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 $(\mathbf{L}_k)_k$, 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 $(\mathbf{L}_k)_k$ tends to $\mathbf{L}$ in norm means $(\mathbf{L}_k)_k$ 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.

Proof by QR decomposition

Let $\mathbf{A}$ be a positive semi-definite Hermitian matrix. Then it can be written as a product of its square root matrix, $\mathbf{A} = \mathbf{B}\mathbf{B}^*$. Now QR decomposition can be applied to $\mathbf{B}^*$, resulting in $\mathbf{B}^* = \mathbf{Q}\mathbf{R}$, where $\mathbf{Q}$ is unitary and $\mathbf{R}$ is upper triangular. Inserting the decomposition into the original equality yields $A = \mathbf{B}\mathbf{B}^* = (\mathbf{Q}\mathbf{R})^*\mathbf{Q}\mathbf{R} = \mathbf{R}^*\mathbf{Q}^*\mathbf{Q}\mathbf{R} = \mathbf{R}^*\mathbf{R}$. Setting $\mathbf{L} = \mathbf{R}^*$ completes the proof.

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

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

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

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

there is $\langle h, \mathbf{A}h \rangle \geq 0$, 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 libraries

- C programming language: the GNU Scientific Library provides several implementations of Cholesky decomposition.
- Maxima computer algebra system: function `cholesky` computes Cholesky decomposition.
- GNU Octave numerical computations system provides several functions to calculate, update, and apply a Cholesky decomposition.
- The LAPACK library provides a high performance implementation of the Cholesky decomposition that can be accessed from Fortran, C and most languages.
- In Python, the function `cholesky` from the `numpy.linalg` module performs Cholesky decomposition.
- In Matlab, the `chol` function gives the Cholesky decomposition. Note that `chol` uses the upper triangular factor of the input matrix by default, i.e. it computes $A = R^*R$ where R is upper triangular. A flag can be passed to use the lower triangular factor instead.
- In R, the `chol` function gives the Cholesky decomposition.
- In Julia, the `cholesky` function from the `LinearAlgebra` standard library gives the Cholesky decomposition.
- In Mathematica, the function "`CholeskyDecomposition`" can be applied to a matrix.
- In C++, multiple linear algebra libraries support this decomposition:
  - The Armadillo (C++ library) supplies the command `chol` to perform Cholesky decomposition.
  - The Eigen library supplies Cholesky factorizations for both sparse and dense matrices.
  - In the ROOT package, the `TDecompChol` class is available.
- In Analytica, the function `Decompose` gives the Cholesky decomposition.
- The Apache Commons Math library has an implementation (https://commons.apache.org/proper/commons-math/commons-math-docs/apidocs/org/apache/commons/math4/legacy/linear/CholeskyDecomposition.html) which can be used in Java, Scala and any other JVM language.

See also

- Cycle rank
- Incomplete Cholesky factorization
- Matrix decomposition
- Minimum degree algorithm
- Square root of a matrix
- Sylvester's law of inertia
- Symbolic Cholesky decomposition

Notes

1. Benoit (1924). "Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues (Procédé du Commandant Cholesky)". *Bulletin Géodésique* (in French). **2**: 66–67. doi:10.1007/BF03031308 (https://doi.org/10.1007%2FBF03031308).

2. Press, William H.; Saul A. Teukolsky; William T. Vetterling; Brian P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing* (https://archive.org/details/numericalrecipes0865unse/page/994) (second ed.). Cambridge University England EPress. p. 994 (https://archive.org/details/numericalrecipes0865unse/page/994). ISBN 0-521-43108-5. Retrieved 2009-01-28.
3. Golub & Van Loan (1996, p. 143), Horn & Johnson (1985, p. 407), Trefethen & Bau (1997, p. 174).
4. Horn & Johnson (1985, p. 407).
5. "matrices - Diagonalizing a Complex Symmetric Matrix" (https://mathoverflow.net/questions/125960/diagonalizing-a-complex-symmetric-matrix). *MathOverflow*. Retrieved 2020-01-25.
6. Schabauer, Hannes; Pacher, Christoph; Sunderland, Andrew G.; Gansterer, Wilfried N. (2010-05-01). "Toward a parallel solver for generalized complex symmetric eigenvalue problems" (https://doi.org/10.1016%2Fj.procs.2010.04.047). *Procedia Computer Science*. ICCS 2010. **1** (1): 437–445. doi:10.1016/j.procs.2010.04.047 (https://doi.org/10.1016%2Fj.procs.2010.04.047). ISSN 1877-0509 (https://search.worldcat.org/issn/1877-0509).
7. Golub & Van Loan (1996, p. 147).
8. Gentle, James E. (1998). *Numerical Linear Algebra for Applications in Statistics*. Springer. p. 94. ISBN 978-1-4612-0623-1.
9. Higham, Nicholas J. (1990). "Analysis of the Cholesky Decomposition of a Semi-definite Matrix" (http://eprints.maths.manchester.ac.uk/1193/). In Cox, M. G.; Hammarling, S. J. (eds.). *Reliable Numerical Computation*. Oxford, UK: Oxford University Press. pp. 161–185. ISBN 978-0-19-853564-5.
10. Krishnamoorthy, Aravindh; Menon, Deepak (2011). "Matrix Inversion Using Cholesky Decomposition". **1111**: 4144. arXiv:1111.4144 (https://arxiv.org/abs/1111.4144). Bibcode:2011arXiv1111.4144K (https://ui.adsabs.harvard.edu/abs/2011arXiv1111.4144K). {{cite journal}}: Cite journal requires |journal= (help)
11. So, Anthony Man-Cho (2007). *A Semidefinite Programming Approach to the Graph Realization Problem: Theory, Applications and Extensions* (http://www.se.cuhk.edu.hk/~manchoso/papers/thesis.pdf) (PDF) (PhD). Theorem 2.2.6.
12. Golub & Van Loan (1996, Theorem 4.1.3)
13. Pope, Stephen B. "Algorithms for ellipsoids. (https://tcg.mae.cornell.edu/pubs/Pope_FDA_08.pdf)" Cornell University Report No. FDA (2008): 08-01.
14. Arora, Jasbir Singh (2004-06-02). *Introduction to Optimum Design* (https://books.google.com/books?id=9FbwVe577xwC&pg=PA327). Elsevier. ISBN 978-0-08-047025-2.
15. Matlab randn documentation (http://www.mathworks.com/help/techdoc/ref/randn.html). mathworks.com.
16. ?potrf Intel® Math Kernel Library [1] (https://software.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/lapack-routines/lapack-linear-equation-routines/lapack-linear-equation-computational-routines/matrix-factorization-lapack-computational-routines/potrf.html#potrf)
17. 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). {{cite journal}}: Cite journal requires |journal= (help)
18. Watkins, D. (1991). *Fundamentals of Matrix Computations* (https://archive.org/details/fundamentalsofma0000watk). New York: Wiley. p. 84 (https://archive.org/details/fundamentalsofma0000watk/page/84). ISBN 0-471-61414-9.
19. Nocedal, Jorge (2000). *Numerical Optimization*. Springer.
20. Fang, Haw-ren (24 August 2007). "Analysis of Block LDLT Factorizations for Symmetric Indefinite Matrices". {{cite journal}}: Cite journal requires |journal= (help)
21. Based on: Stewart, G. W. (1998). *Basic decompositions*. Philadelphia: Soc. for Industrial and Applied Mathematics. ISBN 0-89871-414-1.
22. Osborne, M. (2010), Appendix B.

References

- Dereniowski, Dariusz; Kubale, Marek (2004). "Cholesky Factorization of Matrices in Parallel and Ranking of Graphs". *5th International Conference on Parallel Processing and Applied Mathematics* (https://web.archive.org/web/20110716060800/http://www.eti.pg.gda.pl/katedry/kams/wwwkams/pdf/Cholesky_fmprg.pdf) (PDF). Lecture Notes on Computer Science. Vol. 3019. Springer-Verlag. pp. 985–992. doi:10.1007/978-3-540-24669-5_127 (https://doi.org/10.1007%2F978-3-540-24669-5_127). ISBN 978-3-540-21946-0. Archived from the original (http://www.eti.pg.gda.pl/katedry/kams/wwwkams/pdf/Cholesky_fmprg.pdf) (PDF) on 2011-07-16.
- Golub, Gene H.; Van Loan, Charles F. (1996). *Matrix Computations* (3rd ed.). Baltimore: Johns Hopkins. ISBN 978-0-8018-5414-9.
- Horn, Roger A.; Johnson, Charles R. (1985). *Matrix Analysis*. Cambridge University Press. ISBN 0-521-38632-2.
- S. J. Julier and J. K. Uhlmann. "A General Method for Approximating Nonlinear Transformations of ProbabilityDistributions (https://web.archive.org/web/20190224070338/http://pdfs.semanticscholar.org/523a/865ffabb50d10f85d141963d40528e952760.pdf)".
- S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems (http://kom.aau.dk/~tba/ESIF/julier97new.pdf)", 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.
- Osborne, Michael (2010). *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature* (http://www.robots.ox.ac.uk/~mosb/public/pdf/2160/full_thesis.pdf) (PDF) (thesis). University of Oxford.
- Ruschel, João Paulo Tarasconi, Bachelor degree "Parallel Implementations of the Cholesky Decomposition on CPUs and GPUs (https://www.lume.ufrgs.br/bitstream/handle/10183/151001/001009773.pdf)" Universidade Federal Do Rio Grande Do Sul, Instituto De Informatica, 2016, pp. 29-30.

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) (in French and English) [for English, click 'A télécharger']

Information

- "Cholesky factorization" (https://www.encyclopediaofmath.org/index.php?title=Cholesky_factorization), *Encyclopedia of Mathematics*, EMS Press, 2001 [1994]
- Cholesky Decomposition (https://web.archive.org/web/20060518112024/http://rkb.home.cern.ch/rkb/AN16pp/node33.html#SECTION000330000000000000000), The Data Analysis BriefBook
- 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 (DPOTRF, DPOTRF2, details (http://www.netlib.org/utk/papers/factor/node9.html) performance (http://www.netlib.org/utk/papers/factor/node13.html))
- ALGLIB (http://www.alglib.net/) includes a partial port of the LAPACK to C++, C#, Delphi, Visual Basic, etc. (spdmatrixcholesky, hpdmatrixcholesky)
- 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" (http://ceres-solver.org/) by Google.
- LDL decomposition (https://web.archive.org/web/20120807190828/http://infohost.nmt.edu/~borchers/ldlt.html) routines in Matlab.
- Armadillo (http://arma.sourceforge.net/download.html) is a C++ linear algebra package
- Rosetta Code (http://rosettacode.org/wiki/Rosetta_Code) is a programming chrestomathy site. on page topic (https://rosettacode.org/wiki/Cholesky_decomposition).
- AlgoWiki (https://algowiki-project.org/en/Open_Encyclopedia_of_Parallel_Algorithmic_Features) is an open encyclopedia of algorithms' properties and features of their implementations on page topic (https://algowiki-project.org/en/Cholesky_decomposition)
- Intel® oneAPI Math Kernel Library (https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html) Intel-Optimized Math Library for Numerical Computing ?potrf (https://software.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/lapack-routines/lapack-linear-equation-routines/lapack-linear-equation-computational-routines/matrix-factorization-lapack-computational-routines/potrf.html#potrf), ?potrs (https://software.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/lapack-routines/lapack-linear-equation-routines/lapack-linear-equation-computational-routines/solving-systems-of-linear-equations-lapack-computational-routines/potrs.html#potrs)

Use of the matrix in simulation

- Generating Correlated Random Variables and Stochastic Processes (http://www.columbia.edu/~mh2078/MonteCarlo/MCS_Generate_RVars.pdf), Martin Haugh, Columbia University

Online calculators

- Online Matrix Calculator (https://web.archive.org/web/20081212221215/http://www.bluebit.gr/matrix-calculator/) Performs Cholesky decomposition of matrices online.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Cholesky_decomposition&oldid=1255519126"