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, which is useful for efficient numerical solutions, e.g. 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]

Statement

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

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

where **L** is a lower triangular matrix with real and positive diagonal entries, and **L*** denotes the conjugate transpose of **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 **A** is Hermitian and positive semi-definite, then it still has a decomposition of the form **A** = **LL*** if the diagonal entries of **L** are allowed to be zero.[3]

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

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

The converse holds trivially: if **A** can be written as **LL*** for some invertible **L**, lower triangular or otherwise, then **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 **L** is a lower unit triangular (unitriangular) matrix, and **D** is a diagonal matrix.

This decomposition is related to the classical Cholesky decomposition of the form **LL*** as follows:

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

Or, given the classical Cholesky decomposition $\mathbf{L}^{\text{Cholesky}}$, the $\mathbf{LDL}^*$ form can be found by using the property that the diagonal of **L** must be 1 and that both the Cholesky and the $\mathbf{LDL}^T$ form are lower triangles,[5] if **S** is a diagonal matrix that contains the main diagonal of $\mathbf{L}^{\text{Cholesky}}$, then

$$\begin{aligned}\mathbf{D} &= \mathbf{S}^2, \\ \mathbf{L} &= \mathbf{L}^{\text{Cholesky}}\mathbf{S}^{-1}.\end{aligned}$$

The **LDL** variant, if efficiently implemented, requires the same space and computational complexity to construct and use but avoids extracting square roots.[6] Some indefinite matrices for which no Cholesky decomposition exists have an LDL decomposition with negative entries in **D**. For these reasons, the LDL decomposition may be preferred. For real matrices, the factorization has the form **A** = **LDL**$^{\mathsf{T}}$ and is often referred to as **LDLT decomposition** (or LDL$^{\mathsf{T}}$ decomposition, or LDL'). It is closely related to the eigendecomposition of real symmetric matrices, **A** = **QΛQ**$^{\mathsf{T}}$.

Example

Here is the Cholesky decomposition of a symmetric real matrix:

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

Applications

The Cholesky decomposition is mainly used for the numerical solution of linear equations . If **A** is symmetric and positive definite, then we can solve by first computing the Cholesky decomposition , then solving for **y** by forward substitution, and finally solving for **x** by back substitution.

An alternative way to eliminate taking square roots in the decomposition is to compute the Cholesky decomposition , then solving for **y**, and finally solving .

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 **Ax** = **b** with **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 **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 **s** defined by solving **Hs** = −**g** for **s**, where **s** is the step, **g** is the *gradient* vector of the function's partial first derivatives with respect to the parameters, and **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 **L**. Applying this to a vector of uncorrelated samples **u** produces a sample vector **Lu** with the covariance properties of the system being modeled.[7]

For a simplified example that shows the economy one gets from the Cholesky decomposition, say one needs to generate two correlated normal variables and with given correlation coefficient . All one needs to do is to generate two uncorrelated Gaussian random variables and . We set and .

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 **x** of length *N* and covariance as an *N* × *N* matrix **P**. The matrix **P** is always positive semi-definite and can be decomposed into $\mathbf{L}\mathbf{L}^{\mathrm{T}}$. The columns of **L** can be added and subtracted from the mean **x** to form a set of 2*N* 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 operations (multiplications).[6] The entire inversion can even be efficiently performed in-place.

A non-Hermitian matrix **B** can also be inverted using the following identity, where **BB*** will always be Hermitian:

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 ($n^3/6$ multiplications and the same number of additions), where n is the size of the matrix **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 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

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

then we can write $\mathbf{A}^{(i)}$ 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).

We repeat this for *i* from 1 to *n*. After *n* steps, we get $\mathbf{A}^{(n+1)} = \mathbf{I}$. Hence, the lower triangular matrix *L* we are looking 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 we write out the equation

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

we obtain the following formula for the entries of **L**:

$$L_{j,j} = \sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2},$$
$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right) \quad \text{for } i > j.$$

The expression under the square root is always positive if **A** is real and positive-definite.

For complex Hermitian matrix, the following formula applies:

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

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 **Ax** = **b**, and **y** denotes the computed solution, then **y** solves the perturbed system (**A** + **E**)**y** = **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.[8] 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[9]

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

If **A** is real, the following recursive relations apply for the entries of **D** and **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 **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 **LDL*** factorization is known to be unstable without careful pivoting;[10] specifically, the elements of the factorization can grow arbitrarily. A possible improvement is to perform the factorization on block sub-matrices, commonly 2 × 2:[11]

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

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

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

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 little function[12] 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;
        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
```

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 we have a symmetric and positive definite matrix $\mathbf{A}$ 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

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

we are interested in finding the Cholesky factorisation of $\tilde{\mathbf{A}}$, which we call $\tilde{\mathbf{S}}$, 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 $\mathbf{chol}(\mathbf{M})$ for the Cholesky decomposition of $\mathbf{M}$, the following relations can be found;

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

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

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

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 we wish 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

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

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.[13]

Proof for positive semi-definite matrices

The above algorithms show that every positive definite matrix 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 is an positive semi-definite matrix, then the sequence

consists of positive definite matrices. (This is an immediate consequence of, for example, the spectral mapping theorem for the polynomial functional calculus.) Also,

in operator norm. From the positive definite case, each has Cholesky decomposition . By property of the operator norm,

So 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 , with limit . It can be easily checked that this has the desired properties, i.e. , and is lower triangular with non-negative diagonal entries: for all and ,

Therefore, . Because the underlying vector space is finite-dimensional, all topologies on the space of operators are equivalent. So tends to in norm means tends to entrywise. This in turn implies that, since each is lower triangular with non-negative diagonal entries, 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

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

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

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

we have $\langle h, \mathbf{A}h \rangle \geq 0$, then there exists a lower triangular operator matrix **L** such that **A** = **LL***. One can also take the diagonal entries of **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 Programming, the "chol" command can be used to simply apply this to a matrix.
- In R and Julia, the "chol" function gives the Cholesky decomposition.
- In Mathematica, the function "CholeskyDecomposition" can be applied to a matrix.
- In C++, the command "chol" from the armadillo library performs 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 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. Press, William H.; Saul A. Teukolsky; William T. Vetterling; Brian P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing* (second ed.). 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. variance – **LDL**$^{\mathsf{T}}$ decomposition from Cholesky decomposition – Cross Validated . Stats.stackexchange.com (2016-04-21). Retrieved on 2016-11-02.

6. Krishnamoorthy, Aravindh; Menon, Deepak (2011). "Matrix Inversion Using Cholesky Decomposition". **1111**: 4144. arXiv:1111.4144. Bibcode:2011arXiv1111.4144K .

7. Matlab randn documentation . mathworks.com.

8. Fang, Haw-ren; O'Leary, Dianne P. (8 August 2006). "Modified Cholesky Algorithms: A Catalog with New Approaches" (PDF).

9. Watkins, D. (1991). *Fundamentals of Matrix Computations*. New York: Wiley. p. 84. ISBN 0-471-61414-9.

10. Nocedal, Jorge (2000). *Numerical Optimization*. Springer.

11. Fang, Haw-ren (24 August 2007). "Analysis of Block LDLT Factorizations for Symmetric Indefinite Matrices".

12. Based on: Stewart, G. W. (1998). *Basic decompositions*. Philadelphia: Soc. for Industrial and Applied Mathematics. ISBN 0-89871-414-1.

13. 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* (PDF). Lecture Notes on Computer Science. **3019**. Springer-Verlag. pp. 985–992. doi:10.1007/978-3-540-24669-5_127 . ISBN 978-3-540-21946-0. Archived from the original (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".
- 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.
- Osborne, Michael (2010). *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature* (PDF) (thesis). University of Oxford.

External links

History of science

- *Sur la résolution numérique des systèmes d'équations linéaires*, Cholesky's 1910 manuscript, online and analyzed on BibNum (in French) (in English) [for English, click 'A télécharger']

Information

- Hazewinkel, Michiel, ed. (2001) [1994], "Cholesky factorization" , *Encyclopedia of Mathematics*, Springer Science+Business Media B.V. / Kluwer Academic Publishers, ISBN 978-1-55608-010-4
- "Cholesky Decomposition" . *PlanetMath*.
- Cholesky Decomposition , The Data Analysis BriefBook
- Cholesky Decomposition on www.math-linux.com
- Cholesky Decomposition Made Simple on Science Meanderthal

Computer code

- LAPACK is a collection of FORTRAN subroutines for solving dense linear algebra problems

- ALGLIB includes a partial port of the LAPACK to C++, C#, Delphi, Visual Basic, etc.
- libflame is a C library with LAPACK functionality.
- Notes and video on high-performance implementation of Cholesky factorization at The University of Texas at Austin.
- Cholesky : TBB + Threads + SSE is a book explaining the implementation of the CF with TBB,threads and SSE (in Spanish).
- library "Ceres Solver" by Google.
- LDL decomposition routines in Matlab.
- Armadillo is a C++ linear algebra package

Use of the matrix in simulation

- Generating Correlated Random Variables and Stochastic Processes , Martin Haugh, Columbia University

Online calculators

- Online Matrix Calculator Performs Cholesky decomposition of matrices online.

Last edited 10 days ago by an anonymous user
