|  |  |
| --- | --- |
|  |  |

The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program   
of the Lobachevsky State University of Nizhni Novgorod   
among the world's research and education centers

Strategic initiative  
“Achieving leading positions in the field of supercomputer technology   
and high-performance computing”

**DIRECT METHODS   
FOR SOLVING SYSTEM OF LINEAR EQUATIONS**

*Lecture 3. Cholesky factorization for systems  
with symmetric positive definite matrix*

Nizhni Novgorod

2014

# Objectives

The objective of the lecture is the investigation of the direct methods for solving linear equations systems with symmetric positive definite matrix and also the research of the approaches to their efficient parallel implementation in shared memory systems.

# Abstract

Method for solving systems of *n* linear algebraic equations with a dense real matrix *A* and a right-hand side *b* (the case of the symmetric positive definite matrix *A*) is considered. This method is based on decomposing the matrix *A* into the multiplication of two symmetric triangular matrices (Cholesky factorization, or Cholesky decomposition). The classical variants of the algorithm are stated. The estimates for the algorithm error that arises when calculating with an error are shown. The insufficient efficiency of using the classical algorithms in parallel computation systems is demonstrated. The idea of block data processing as a way to increase memory usage efficiency is described step-by-step.

# GUIDELINES

The methods for solving systems of *n* linear algebraic equations with a symmetric positive definite (SPD) matrix *A* and a right-hand side *b* are investigated. The problem of solving a linear equations system with the given matrix *A* and the vector *b* is considered to be a problem of searching a value of the unknown vector *x* such that all the equations of the system hold.

A matrix is said to be *positive definite* if for any vector *x*∈*Rn* the inequity (*Ax*, *x*)>0 holds. The matrices with such properties appear, for example, when using least squares method, solving differential equations numerically.

The method for solving is based on decomposing the matrix *A* into the multiplication *A*=*LLT*, where *L* is a lower triangular matrix with the positive elements on the main diagonal. It is known that the Cholesky factorization can be derived for any SPD-matrix. The generalization of this decomposition to the case of complex matrices exists. If the decomposition is obtained (i.e. the forward phase of the method is executed), the solving of the source system is reduced to the sequential solving of two equations systems with the triangular matrices (the backward phase).

The calculation formulas of the method are derived analogously to the compact scheme of the *LU*-decomposition taking into account the symmetry of the matrix *A*. The time complexity of the factorization amounts to 1/3*n*3+*O*(*n*2), whereas the complexity of the backward phase is *O*(*n*2).

The various versions of the decomposition implementation (columnwise, rowwise) are possible. The efficiency of a specific implementation is conditioned by a way for storing the matrix *A* (by rows as in C or by columns as in Fortran).

The choice of the pivot elements in the Cholesky method is not required. The analysis of the method error that accumulates during the calculations with error demonstrates that the accumulated error is analogous to the error of the Gaussian method with choosing the pivot element. The factorization error depends on the machine accuracy *εm*, the system size *n* and the matrix norm ||*A*||.

The parallelization of the method could be performed in the following way (on example of columnwise algorithm). All the calculations are reduced to the independent one-type operations with the rows of the matrix *A*. As a result, the principle of parallelization by data distribution could be a base for parallel implementation of the Cholesky decomposition. All the calculations that include processing a group of the rows of the matrix *A* when calculating the elements from the column of the factor *L* could be considered as a subtask. In this case, applying the sequential scheme of data division leads to an unequal computational load among the threads: as the factor *L* for the most threads will be formed, all required calculations will be completed and the threads will happen to be idle. The possible solution for the problem of the calculations balance may consist in using the cyclic striped scheme of data distribution among the subtasks.

The results of the experiments demonstrate that, when the number of the threads is *p*≥3, the speedup is approximately 2 and it depends neither on the number of the threads, nor on the matrix size, although we could expect better results according to the preliminary estimates (an exception is the case of *N*=1000 and *N*=2000). The absence of considerable speedup is caused by inefficient cache-memory usage.

In cases when *N*=1000 and *N*=2000, the size of the lower triangle of the matrix *A* makes it possible to load it to the processor cache entirely and to use the high-speed memory for executing the operations. Meantime, if *N*>2000, the lower triangle does not fit in the cache-memory entirely, so the cache misses number increases and this leads to frequent accessing the main memory that is slow enough. The cache memory may be used efficiently if the block data division is used (this will be shown later).

The disadvantage of the classical Cholesky factorization is conditioned by the fact that the method computations conform poorly to the rules of using the *cache-memory*. The cache-memory is the extra high-speed computer memory that is used for storing the copies of the most frequently used main memory areas. The result of the cache-memory usage will be observable if the executed calculations use the same data many times and access the memory elements with the sequentially increasing addresses.

In the considered algorithm the data is located in memory by rows, while the calculations are executed by columns. This results in low efficiency of cache-memory usage. A possible way to improve the situation is the consolidation of the computational operations, leading to the sequential processing of some rectangular submatrices of the matrix *A*.

The Choleskydecomposition may be organized so that the matrix operations would be basic operations. Note that implementation of the matrix operations makes the efficient cache-memory usage possible. Let us represent the matrix *A* in a block form and then execute the block decomposition. One step of the block algorithm consists in the following:

1. The execution of common Cholesky factorization for a small submatrix of size r×r.
2. Solving system with the triangular matrix and many right-hand sides.
3. The calculation of the reduced matrix of smaller size (n−r)×(n−r) as a result of matrix multiplication.

After that, the algorithm is applied to the reduced matrix.

As well as the other investigated decomposition procedures, the given scheme requires 1/3*n*3+*O*(*n*2) operations. A portion of the matrix operations may be estimated as 1−3/(2*N*), where *N* is the number of the blocks.

The block Cholesky factorization should be parallelized on the level of matrix operations.

The results indicate the good (almost linear) scalability of the block algorithm in the opposite to the standard version of the decomposition.

# RECOMMENDATIONS for students

The books [1], [2] include the description of the classical Cholesky factorization. Also the problems of stability of the obtained solutions, when calculating with errors, are studied there. The dependence of the derived solutions accuracy on the matrix conditionality of the linear system is demonstrated.

The block variants of the algorithms are stated in the books [3], [4]. Taking into account the matrix and scalar operations, the complexity estimates are given. Every algorithm description is followed by the pseudocode. The organization of the parallel computations for shared memory is considered.

The implementations of the parallel algorithms on pseudocode (for the case of distributed memory) are stated in [5]. A concrete realization of the parallel algorithms on C/C++ could be found in [6].

# References

1. David R. Kincaid and E. Ward Cheney. Numerical analysis: mathematics of scientific computing. Brooks/Cole Publishing Company, 1991.
2. Richard L. Burden, J. Douglas Faires. Numerical Analysis. Brooks Cole, 2000.
3. Gene H. Golub, Charles F. Van Loan. Matrix Computations. The John Hopkins University Press, 1996.
4. James W. Demmel. Applied Numerical Linear Algebra. SIAM, 1997.
5. M. Quinn. Parallel programming in C with MPI and OpenMP. McGraw-Hill, 2004.
6. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. Numerical Recipes. The Art of Scientific Computing. Cambridge University Press, 2007.

# EXERCISES

1. Implement the block algorithm for solving the linear equations system with a triangular matrix and several right-hand sides *LX*=*B*.
2. Implement the Cholesky factorization for a symmetric positive definite matrix.
3. Implement the parallel block Cholesky factorization for a symmetric positive definite matrix.

# TEST QUESTIONS

1. What is the complexity order of the backward phase of the Cholesky method?
   1. *n*
   2. *n*2
   3. +2*n*2
2. What is the complexity order of the Cholesky decomposition (forward phase)?
   1. *n*3+O(*n*2)
   2. + 1/3*n*3+O(*n*2)
   3. 2/3*n*3+O(*n*2)
3. The connection between the *LU*-decomposition and the Cholesky factorization consists in the following:
   1. the matrix *U* coincides with matrix *L−*1
   2. +the matrix *U* coincides with matrix *LT*
   3. the time complexity of the methods is the same
4. The parallel Cholesky decomposition, when using non-block parallelization and working with the matrices of great size,
   1. demonstrates low efficiency because of the fast execution of the sequential algorithm
   2. + demonstrates low efficiency because of the great number of the cache misses in the parallel algorithm
   3. demonstrates high efficiency
5. Which of the three variants of the Choleskyfactorization (rowwise, columnwise, with changes of submatrix) is preferable for the implementation (matrix is stored by rows)?
   1. Rowwise, since all the elements of the columns from the first to the *i*-th are used when calculating the *i*-th row and the access to the matrix column in C language is organized efficiently.
   2. + Columnwise, since all the elements of the rows from the *j*-th to the *n*-th are used when calculating the *j*-th column and the access to the matrix row in C language is organized efficiently.
   3. The algorithm changing the submatrix, since it has the smaller complexity than the complexity of the first two algorithms.
6. Is the block Cholesky decomposition algorithm more efficient than the common algorithm?
   1. No, the methods have the equal efficiency because their complexities are equal
   2. No, the common algorithm is more efficient in view of smaller complexity
   3. + Yes, the block algorithm is more efficient in view of proper cache-memory usage when their complexities are the same.
   4. Yes, the block algorithm is more efficient in view of smaller complexity
7. Is choosing the pivot element is obligatory for the Cholesky factorization?
   1. Yes, it is obligatory
   2. No, it is not (but only for the orthogonal matrices)
   3. + No, it is not obligatory (for any symmetric positive definite matrix)
8. What stage of the parallel block *LLT*-decomposition algorithm is the most time-consuming and thus it determines the total efficiency of the algorithm?
   1. Execution of the *LLT* -decomposition for submatrix
   2. Solving the system with the triangular matrix and many right-hand sides
   3. + Calculation of the reduced matrix