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

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 6. Solving of systems of linear algebraic equations with sparse matrix by example of Cholesky factorization.*

Nizhni Novgorod

2014

# Objectives

The objective of the lecture is the investigation of the direct methods for solving system of linear equations with sparse matrix by example of Cholesky factorization.

# Abstract

The solving of a system of linear equations with a sparse matrix is considered in the lecture by example of Cholesky factorization. The main problem that arises when solving sparse systems is a problem of fill-in of a factor. The approaches to solve this problem that are related to the reordering of the equations system matrix are discussed. A general scheme for solving the sparse system of linear equations is stated. This scheme includes reordering, symbolic and numerical factorization and backward phase. Two widely known methods for searching an optimal matrix permutation are considered: minimum degree and nested dissection methods. The approaches to the efficient implementation of the symbolic and numerical factorization are described. The advantage of the nested dissection method when parallelizing the algorithm is shown.

# GUIDELINES

The Cholesky method for solving a linear equations system with a symmetric positive definite matrix *A* is investigated. The matrix of the system is supposed to be sparse.

The most important fact about applying the Cholesky method for the sparse matrices consists in that the factor is, as a rule, *filled*. This means the nonzero elements appear in such positions of the matrix *L* that were filled with zero in the lower triangular part of *A*. However, if, before the factorization, the system is reordered in ways including renumbering of the variables and rearranging the equations (that is equivalent to symmetric permutation of the rows and columns of the matrix *A*), the fill-in of the factor will be reduced considerably.

The problem of searching for a permutation that minimizes the fill-in when calculating the Cholesky factor is *NP*-hard (because, generally speaking, the minimum should be calculated over all *n!* permutations). Therefore the heuristic algorithms are used in practice. They, as a rule, give an acceptable result although do not guarantee that the optimal permutation is found.

Thus, the following phases are carried out when solving sparse systems by the Cholesky method:

1. *Reordering* – the computation of the permutation matrix *P*;
2. *Symbolic factorization* *(decomposition)* – the construction of a pattern of the matrix *L* and the allocation of the memory for storing the nonzero elements;
3. *Numerical factorization* *(decomposition)* – the calculation of the values of the matrix *L* and placing them in the allocated memory;
4. *Backward phase* – the solving of two triangular equations systems.

The steps 1 and 2 are specific to the sparse matrices, while the steps 3 and 4 are performed for any problem.

In the given lecture we consider two algorithms for determining the permutation matrix: the *minimum degree* and *nested dissection* methods. Both of them are based on graph model: the matrix *A* is considered as an adjacency matrix for a certain graph *G(A)*; all the operations are performed on this graph and lead to obtaining the permutation.

Later, let us examine the symbolic and numerical phases of the decomposition. It is shown how to take advantage of matrix sparseness in order to reduce a number of the performed operations. For example, for calculating the pattern of the *j*-th column in the symbolic decomposition it is not necessary to merge all of the patterns of the preceding columns with numbers 1, 2, …, (j-1), but only those in which the nonzero element in the *j*-th row is the first nonzero element of the given column below the diagonal. The numerical factorization algorithm is based on the similar ideas.

In order to construct the parallel algorithm, a notion of a *fill-in graph F(A)* (a graph with an adjacency matrix formed from the factor *L*) and an *elimination tree T(A)* (a spanning tree of the fill-in graph with the root at the *n*-th vertex). The importance of the elimination tree consists in that it determines the data dependencies between the columns of the matrix *L* during the numerical decomposition. The values of the column with the number that corresponds to a certain tree vertex depend on all of the columns whose numbers are in the subtree with the root at this vertex. If *Ti* and *Tj* are disjoint subtrees of the elimination tree *T(A)*, the values of the columns with the numbers that correspond to the vertices of the subtree *Ti* do not depend on the values of the columns with the numbers that correspond to the vertices of the subtree *Tj*. The specified property gives an opportunity to calculate the values of such columns in parallel.

Moreover, the elimination tree determines the data dependencies when conducting the backward phase as well. For the disjoint subtrees *Ti* and *Tj* of the elimination tree *T(A)* the values of the unknowns with the numbers that correspond to the vertices of the subtree *Ti* do not depend on the values of the unknowns with the numbers that correspond to the vertices of the subtree *Tj*. This allows to perform the backward phase in parallel.

The elimination tree characterizes the parallel factorization algorithm in general, the elimination tree height is an estimate for the algorithm execution time (the higher the tree, the longer the algorithm will work), and its width is an estimate for the parallelism level in the problem (the wider the tree, the greater the parallelism level).

The results of the computational experiments, which were carried out when reordering and decomposing a series of the test matrices, are demonstrated. The results approve that the reordering according to the nested dissection method gives the elimination tree with the smaller height than when applying the minimum degree method. This means the nested dissection method is more perspective for constructing the parallel decomposition algorithm. The execution time for the algorithms of the symbolic and numerical factorization and the backward phase are also presented. These results show that the most time-consuming part of the algorithm is the numerical decomposition.

# RECOMMENDATIONS for students

Notwithstanding the long-standing edition, the books [1], [2] have not lost their relevance today. The description of the basic approaches to the solving of the sparse linear equations system is given in these books. The pseudocode implementation for the algorithms is also presented.

A C-language implementation of some algorithms working with the sparse matrices is presented in [3].

# References

1. Alan George, Joseph W. H. Liu. Computer solution of large sparse positive definite systems. Prentice-Hall, 1981.
2. Sergio Pissanetzky. Sparse matrix technology. Academic Press, 1984.
3. 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. Develop an implementation of constructing a permutation matrix by means of the minimum degree method. Carry out the computational experiments on the matrices with the different patterns.
2. Develop an implementation of constructing a permutation matrix by means of the nested dissection method. Carry out the computational experiments on the matrices with the different patterns.
3. Implement the Cholesky factorization for the sparse matrices (supposing that the matrix is already reordered by means of one of the previously implemented methods). Carry out the computational experiments on the matrices with the different patterns. Compare the fill-in coefficient of factor without reordering and the one with reordering.

# TEST QUESTIONS

1. Choose the most economical storage format for a tridiagonal matrix of size 1000×1000.
   1. The coordinate format
   2. The compressed sparse columns format
   3. + In the form of three arrays for three diagonals correspondingly.
2. Choose the most economical storage format for a matrix of size 1000×1000, which has a density of 1%.
   1. The coordinate format
   2. + The compressed sparse columns format
   3. The storage format for dense matrix
3. What is the complexity of the Cholesky method?
   1. *n*3+O(*n*2)
   2. + 1/3*n*3+O(*n*2)
   3. 2/3*n*3+O(*n*2)
4. Select the main objective of reordering the linear equations system:
   1. To minimize number of performed operations
   2. To minimize solution error
   3. + To minimize fill-in of a factor
5. The result of the nested dissection method work is
   1. + the construction of the permutation matrix *P*
   2. the found solution *x* of the problem
   3. the found lower triangle of decomposition *L*
6. Select the correct sequence of performing the steps of the problem solving:
   1. Symbolic factorization, reordering, numerical factorization, backward phase;
   2. Symbolic factorization, numerical factorization, reordering, backward phase;
   3. + Reordering, symbolic factorization, numerical factorization, backward phase.
7. What step of solving of the linear equations system is the most time-consuming?
   1. Symbolic factorization;
   2. + Numerical factorization;
   3. Backward phase.
8. The result of the nested dissection method work is
   1. + the construction of the permutation matrix *P*
   2. the found solution *x* of the problem
   3. the found lower triangle of decomposition *L*
9. A graph G(A), associated with the matrix A is constructed as follows:
   1. The matrix AAT is the adjacency matrix for the graph G(A)
   2. The matrix A is the inverse matrix for the adjacency matrix for the graph G(A)
   3. + The matrix A is the adjacency matrix for the graph G(A)
10. The elimination tree T(A) is
    1. + a spanning tree of the fill-in graph F(A) with the root at the *n*-th vertex;
    2. a spanning tree of the fill-in graph F(A) with the root at an arbitrary vertex;
    3. a spanning tree of the fill-in graph G(A) with the root at the *n*-th vertex.
11. For the efficient parallelization of the Cholesky factorization, the elimination tree T(A) must have
    1. + maximum width;
    2. maximum height;
    3. maximum number of vertices.