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

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**

*Practice 1. Application of sweep and reduction methods  
for solving linear equation system with band matrix*

Nizhni Novgorod

2014

# Objectives

The objective of the practice is to demonstrate a practical application of the parallel linear algebra algorithms by example of an applied problem from financial mathematics.

# Abstract

In the given laboratory work a linear algebraic equation system with a tridiagonal matrix is investigated. For solving such system the special methods are known such as, for example, the *sweep* and *cyclic reduction methods* considered in this work. The cyclic reduction method is a little more complex to implement, but rounding errors influence on this method less than on the sweep method. The problems of the cyclic reduction method parallelization in the shared memory systems are discussed in this work.

# GUIDELINES

This work presents a conceptual statement for the problem of pricing a compound option. The changes in the compound option price are described by a partial differential equation. The differential equation is solved using the Crank-Nicolson computational scheme: a discrete grid is introduced in the range of parameters, and the differential equation is approximated by a linear algebraic equation system with a tridiagonal matrix. A peculiarity of this equation system consists in the fact that there is the same number on each diagonal of the matrix. The evaluation of the optimal price of the compound option is reduced, in the final analysis, to multiple solving of a tridiagonal equation system.

Two methods for solving tridiagonal systems are investigated – the sweep and the cyclic reduction methods.

The *sweep method* consists of two phases: a forward phase (calculation of the auxiliary coefficients *αi*, *βi*) and a backward phase (calculation of the values of the unknowns *xi* with the help of *αi*, *βi*). It is known that the fulfillment of the *diagonal dominance* condition is necessary for the computational stability of the sweep method. The sweep method complexity amounts to 8*n*+*O*(1).

The analysis of the formulas for the sweep method shows that in some cases the method could result in great error. A potential error source is the formulas for calculating the “sweep” coefficients, which include a division operation by the difference between close values.

The *reduction method* does not have such disadvantage and, moreover, when being realized on the modern computational systems, it demonstrates the greater efficiency in comparison with the sweep method. A main restriction of the reduction method consists in the fact that it is applied only for the matrices of the size equal to a power of two. The idea of the reduction method consists in two following steps. The forward phase of the method is a sequential elimination of the unknowns from the source system: first, the unknowns with the odd numbers, then – with the numbers multiple to 2 (but not multiple to 4), etc. The backward phase of the method is a restoration of the values of the odd variables based on the known values of the even variables. The complexity of the reduction method amounts 11*n*+*O*(1) operations.

The sweep method is to be implemented in this work as well as the cyclic reduction method for solving the linear equation system that arises while discretizing the differential equation. The last of two is also to be parallelized. The parallelization of the cyclic reduction method is realized as follows. Each next iteration of the forward and backward reduction phase depends on the previous one. On the other hand, the elimination and restoration of each specific variable at certain iteration may be performed independently, i.e. the nested loops of the forward and backward phases may be parallelized.

In order to parallelize the reduction method using OpenMP technology, it is enough to add directive **pragma omp** **parallel for** in front of the corresponding loops. In this case the loops iterations do not have data dependencies, so there is no effect concerning the threads’ access to the same memory area.

A parallel program using the Intel TBB technology is designed slightly more complicated. In the TBB library the function **tbb::parallel\_for()**is used to parallelize loops with a known number of repetitions. This function takes a loop iteration space and an instance of a function class as input parameters. The built-in one-dimensional iteration space **tbb :: blocked\_range** may be used for our problem. The function class is actually an unwound loop body. In order to parallelize the forward phase, it is necessary to implement the function class **FFCompFunctor** and the function **fcomp()** that contains calling of **tbb::parallel\_for()** for the parallel elimination of system variables. The function class **FFCompreverseFunctor** and the functions **fcompreverse()** for parallel variable restoration are to be implemented to parallelize the backward phase.

Upon completion of the software implementation, comparison of the application performance using the sequential versions of the sweep and cyclic reduction methods and analysis of scalability for the parallel implementation of the reduction are performed.

The results of the computational experiments carried out on the sequential versions demonstrate that the time of problem solving by means of the sweep method is more than twice greater than the solving time using the cyclic reduction. This contradicts the theoretical complexity estimate. One of the possible explanations for this fact is that the number of operations for both methods is a quantity of order *O*(*n*), and, in this case, an architecture of the computer on which the experiments were carried out has a significant impact on the result. Thus, an operation of the sweep method is performed for almost 2 cycles on average, while an operation of the cyclic reduction method is performed for about 1 cycle (which is the norm for high-performance applications). At the same time, the number of instructions in the reduction is about 1.3 times greater. Since the linear algebraic equation system is solved many times, the time of determining the optimal price of the compound option using the reduction method is less than when using the sweep method.

The results of the computational experiments also indicate a poor scalability of the parallel algorithms (for both OpenMP- and TBB-versions) since the speedup is, at the best, slightly greater than two for 4 or 8 threads. This is essentially explained by the algorithmic peculiarities of the reduction method. That is why it is necessary either to develop more efficient parallelizing schemes for the cyclic reduction (that is nontrivial due to dependencies between the iterations of the forward and backward phases) or to apply other methods for solving linear equation systems, a parallel implementation of which would have better scalability in comparison with the cyclic reduction.

# RECOMMENDATIONS for students

A detailed statement for the problem of calculating price of a compound option is stated in [1]. The cyclic reduction method (in both sequential and parallel variants) is considered in [2]. The used technologies for parallel programming are described in [3−6].

# References

1. Gong. P, He. Z and Zhu. SP. Pricing convertible bonds based on a multi-stage compound option model, Physica A, 336, 2006, 449-462.
2. Golub, G. H., Ortega, J. M.: Scientific Computing. An Introduction with Parallel Computing. Academic Press, Inc., 1993.
3. Intel® Threading Building Blocks. Reference Manual. Version 1.6. Intel® Corporation, 2007.
4. Intel® Threading Building Blocks. Tutorial. Version 1.6. Intel® Corporation, 2007.
5. Official Website OpenMP [www.openmp.org].
6. TBB library page on the Intel Corporation Website [http://software.intel.com/en-us/articles/intel-tbb].

# EXERCISES

1. In order to solve a tridiagonal system, use the built-in function of the MKL library. Estimate the efficiency of using of the library functions in comparison with the sweep method and the sequential implementation for the cyclic reduction method.
2. Implement the counter-sweep method. Estimate the efficiency of using of the counter-sweep method in comparison with the usual sweep and the parallel implementation of the cyclic reduction method. Explain the derived results of the experiments.
3. Implement the block sweep method. Estimate the efficiency of using it in comparison with all the previous implementations of the methods for solving tridiagonal systems. Explain the derived results of the experiments. Estimate the scalability of the performed implementation of the block sweep.

# TEST QUESTIONS

1. What is the complexity of the sweep method?
   1. + 8*n* + *O*(1)
   2. 8*n*2 + *O*(*n*)
   3. 8*log(*n*)* + *O*(1)
2. Specify the condition of the computational stability for the sweep method
   1. The matrix must be symmetric
   2. +The matrix must be diagonally dominant
   3. The matrix must be nonsingular
3. Choose a preferable format for storing a tridiagonal matrix of size *n*×*n*.
   1. A two-dimensional array of size *n*×*n*
   2. + Three arrays of length *n*, which correspond to three diagonals
   3. *n* arrays of length 3, which correspond to nonzero elements in the columns of the matrix
4. What is the complexity of the reduction method?
   1. + 11*n*+*O*(1)
   2. 11*n*2+*O*(*n*)
   3. 11*log(*n*)* + *O*(1)
5. The reduction method could be applied for the systems with tridiagonal matrix
   1. of any size
   2. of size multiple to 2
   3. + of size multiple to 2*n*
6. In the reduction method
   1. + the parallelization is possible only within one step of the forward or backward phase;
   2. the forward or backward phase could be parallelized entirely;
   3. the parallelization is not possible, the method is not parallelizable.
7. What mode of the Intel Parallel Inspector XE tool should be used in order to detect multithreading errors?
   1. Threading Errors
   2. +Threading Error Analysis
   3. Threading Analysis
8. The Intel Parallel Amplifier XE tool, running in the Concurrency-analysis mode, constructs for each function a timeline, which characterizes the processor resources usage during the program execution. Why are some functions absent in the corresponding list?
   1. +The compiler inlined them, or these functions were not called during the execution.
   2. These functions were not called during the execution.
   3. The compiler inlined them.
9. What construct of the OpenMP technology is used for parallelizing loops?
   1. +#pragma omp parallel for
   2. #pragma omp parallel
   3. #pragma omp for
10. What is the method initialize(…) of the tbb::task\_scheduler\_init class responsible for?
    1. It allows to set the minimum grainsize when partitioning the iteration space during the loop parallelization.
    2. + It activates an object of the tbb::task\_scheduler\_init class, also its optional parameter allows to set the threads number.
    3. It allows to set the threads number.
11. If the deactivate method terminate() is not called explicitly for an object of the tbb::task\_scheduler\_init class, then
    1. +the object will be deactivated at the moment of calling of a destructor of the corresponding object.
    2. the object will not be deactivated.
    3. the object will be deactivated at the moment when the parallel section of the program completes.