Parallel Processing Model for Cholesky Decomposition Algorithm in AlgoWiki Project

Alexander Antonov[1]***, Alexey Frolov***[2]***, Hiroaki Kobayashi***[3]***, Igor Konshin***[124]***, Alexey Teplov***[1]***, Vadim Voevodin***[1]***, Vladimir Voevodin***[1]

The comprehensive analysis of algorithmic properties of well-known Cholesky decomposition is performed on the basis of multifold AlgoWiki technologies. A detailed analysis of information graph, data structure, memory access profile, computation locality, scalability and other algorithm properties is conducted, which allows us to demonstrate a lot of unevident properties split up into machine-independent and machine-dependent subsets. The comprehension of the parallel algorithm structure enable us to implement efficiently the algorithm at hardware platform specified.

Keywords: AlgoWiki, algorithm properties, Cholesky decomposition, memory access locality, dynamic characteristics, scalability.

Introduction

The fundamental problem of high-performance computing consists in the accurate coordination between algorithm and program structure and hardware features that results in high efficiency. Modern computers possess great capability, but if there is a lack of the above mentioned coordination, the final implementation efficiency can be very low. A lot of studies are devoted to the most efficient implementation of some specific algorithm.

This paper presents a comprehensive analysis of Cholesky decomposition algorithm. The analysis is based on AlgoWiki methodology and allows specifying the most important features of the algorithm and its parallel implementation model in order to get the most efficient processing.

AlgoWiki [1, 2] is an open encyclopedia of parallel algorithmic features on the Internet which provides collaboration with the worldwide computing community on algorithm descriptions. The main goal of this project is to develop fundamentals and formalize the mapping of algorithms to the architecture of parallel computers. As a result of the current research, the universal description of the structure of algorithm properties has been developed.

All fundamental algorithmic properties that determine implementation efficiencies on modern computing platforms are divided into machine-dependent and machine-independent subsets. This division is made intentionally in order to separate these features of algorithms, which define their perspective implementations on parallel computational systems from a range of questions associated with consequent stages of programming and execution of the resulting programs on particular computing systems.

The AlgoWiki open encyclopedia is based on wiki technology, which allows for any researcher to add and improve the material. A pilot version of the Internet encyclopedia can be found at [3]. The main part of the AlgoWiki project is the algorithms classification and description of their serial and parallel properties. The algorithm descriptions were performed by groups corresponding to the types of operations involved. It is assumed that this classification will eventually be expanded while other researchers will give their contribution to this project.

[1] Moscow State University
[2] Institute of Numerical Mathematics of the Russian Academy of Sciences
[3] Tohoku University
[4] Dorodnicyn Computing Centre of the Russian Academy of Sciences

To demonstrate the analysis of the parallel processing model we have chosen one of the most popular algorithms — the Cholesky decomposition. This algorithm is widely used for the direct solution of dense and sparse linear systems. The modification of this algorithm is exploited to construct efficient preconditioners for iterative methods.

On the basis of the Cholesky decomposition algorithm properties we demonstrate a mathematical algorithm description, the analysis of sequential and parallel complexity, the information graph of the algorithm, properties of software implementations, analysis of data locality, scalability, as well as dynamic characteristics and performance efficiency of the algorithm implementation. On the basis of the detailed algorithm analysis we give the conclusions on the most efficient algorithm implementation according to facilities of the supercomputer used.

1. Properties and structure of the algorithm

1.1. General description

The Cholesky decomposition algorithm was first proposed by Andre-Louis Cholesky in 1910 [8, 9]. The idea of this algorithm was published by Benoit in 1924 [7] and later was used by Banachiewicz in 1938 [5, 6]. The Cholesky decomposition is also known as the square-root method [10, 15, 16] due to the square root operations used in this decomposition and rarely exploited in some implementations of Gaussian elimination.

Originally, the Cholesky decomposition was used only for dense real symmetric positive definite matrices. At present, the application of this decomposition is much wider. For example, in order to increase the computational performance, its block versions are often applied.

In case of sparse matrices, the Cholesky decomposition is also broadly used as the main stage of a direct method for solving linear systems. In order to reduce the memory requirements and matrix profile, special reordering strategies (such as RCM) are applied to minimize an algorithm arithmetic complexity. A number of reordering strategies (Metis, Zoltan, etc.) are used to identify the independent matrix blocks for parallel computing systems.

Various versions of the Cholesky decomposition are successfully used in iterative methods to construct preconditioners for sparse symmetric positive definite matrices. In the case of incomplete triangular decomposition, the elements of a preconditioning matrix are specified only in predetermined positions (for example, in the positions of the original matrix nonzero elements; this version is known as an IC0 decomposition). In order to construct a more accurate decomposition, a filtration of small elements is performed using a filtration threshold. The use of such a threshold allows one to obtain an accurate decomposition, but the number of nonzero elements may increase. A decomposition algorithm of the second-order accuracy is proposed in [12]; this algorithm allows one to increase the accuracy of decomposition with about the same number of nonzero elements in the factors. In its parallel implementation, a special version of an additive preconditioner is applied on the basis of the second-order decomposition [13].

Here we consider the original version of the Cholesky decomposition for dense real symmetric positive definite matrices. Let us emphasize some basic properties of the algorithm.

Matrix symmetry. The matrix symmetry allows one to store in computer memory slightly more than half of the number of its elements and to reduce the number of operations by a factor of two compared to Gaussian elimination. Note that the LU-decomposition does not require the square-root operations when using the property of symmetry and, hence, is somehow faster than the Cholesky decomposition, but it requires storing the entire matrix.

Accumulation mode. The Cholesky decomposition allows one to use the so-called 'accumulation' mode due to the fact that the significant part of computation involves 'dot product' operations. Hence, these dot products can be accumulated in double precision for additional accuracy. In this mode, the Cholesky method has the least 'equivalent perturbation'. During the process of decomposition, no growth of the matrix elements can occur, since the matrix is symmetric and positive definite. Thus, the Cholesky algorithm is unconditionally stable.

1.2. Mathematical description and information graph

The input data of the algorithm is a symmetric positive definite matrix A of size n with elements a_{ij}, while the output is the lower triangular matrix L with elements l_{ij}.

The Cholesky algorithm can be represented in the form:

$$\begin{aligned}
l_{11} &= \sqrt{a_{11}}, \\
l_{j1} &= \frac{a_{j1}}{l_{11}}, \quad j = 2, ..., n, \\
l_{ii} &= \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \quad i = 2, ..., n, \\
l_{ji} &= \left(a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right) / l_{ii}, \quad i = 2, ..., n-1, \quad j = i+1, ..., n.
\end{aligned}$$

In the last formula the basic elimination step is performed which is the main computational kernel of the algorithm.

The sequential complexity of the Cholesky decomposition algorithm is as follows: n square roots, $n(n-1)/2$ divisions, $(n^3-n)/6$ multiplications and $(n^3-n)/6$ additions (subtractions). Thus, a sequential version of the Cholesky algorithm is of cubic complexity.

There is a block versions of this algorithm; however, here we consider only its standard 'dot' version for simplicity of its informational graph presentation.

An informational graph [17] is a directed acyclic graph the vertices of which correspond to the operations of the algorithm, and the edges — to the transfer of information between them. The informational graph of the dot version of algorithm consists of three groups of vertices positioned in the three-dimensional spaces with integer coordinates. The first and the second groups of vertices belong to the one-dimensional domains corresponding to square root and division a/b operations, respectively. The third group of vertices belongs to the three-dimensional domain corresponding to elimination operation $a - bc$.

The resulting information graph is illustrated in Fig. 1 for $n = 4$. In this graph, the vertices of the first group are highlighted in yellow and marked as SQ; the vertices of the second group are highlighted in green and marked as Div; the vertices of the third group are highlighted in red and marked as F. The vertices corresponding to the results of operations (output data) are marked by large circles. The arcs doubling one another are depicted as a single one. The additional vertices corresponding to input and output data are marked in blue and pink, respectively.

1.3. Parallelization resources and other properties of the algorithm

In order to decompose a matrix of order n, a parallel version of the Cholesky algorithm should sequentially perform the following layers of operations: n layers for square roots (a single

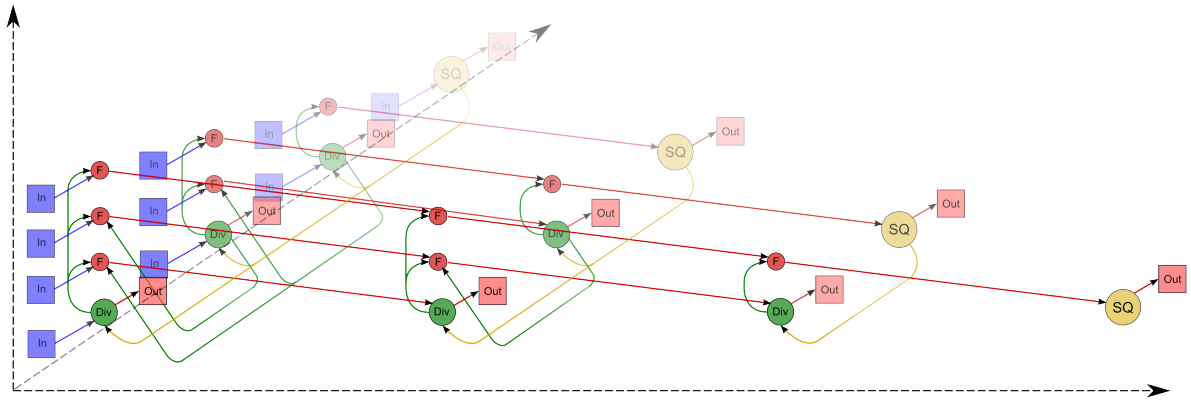

Figure 1. The informational graph of the Cholesky algorithm with input and output data: SQ is the square-root operation, F is the operation $a - bc$, Div is division, In and Out indicate input and output data

square root on each layer); $n-1$ layer for divisions (on every layer, the number of divisions is linear and, depending on a particular layer, varies from 1 to $n-1$); $n-1$ layer of multiplications and $n-1$ layer of additions/subtractions (on every layer, the number of these operations is quadratic and varies from 1 to $(n^2-n)/2$).

Contrary to a sequential version, in a parallel version the square-root and division operations require a significant part of overall computational time. The existence of isolated square roots on some layers of the parallel form may cause other difficulties for particular parallel computing architectures. In the case of symmetric linear systems, the Cholesky decomposition is preferable compared to Gaussian elimination because of the reduction in computational time by a factor of two. However, this is not the case for its parallel version.

In addition, we should mention the fact that the accumulation mode requires multiplications and subtractions in double precision. In a parallel version, this means that almost all intermediate computations should be performed with the data given in their double precision format. Contrary to a sequential version, this almost doubles the memory expenditure.

Thus, the Cholesky decomposition belongs to the class of algorithms of linear complexity in the sense of the height of its parallel form, whereas its complexity is quadratic in the sense of the width of its parallel form.

In the case of unlimited computer resources, the ratio of the sequential complexity to the parallel complexity is *quadratic* with respect to the matrix size n. That is, having n^2 processors the Cholesky algorithm can be completed in n steps, while the sequential version can be finished only in $n^3/6$ steps.

The computational power of the Cholesky algorithm considered as the ratio of the number of operations to the amount of input and output data is only *linear* with respect to n.

The Cholesky decomposition is unique for the positive definite matrix, but another order of associative operations may result in the accumulation of round-off errors; however, the effect of this accumulation is not so high as in case when the accumulation mode is not applied while computing dot products.

The information graph arcs from the vertices corresponding to the square-root and division operations can be considered as groups of data such that the function relating the multiplicity of these vertices and the number of these operations is a linear function of the matrix order and the vertex coordinates. These groups may contain long arcs; the remaining arcs are local.

The most known is the compact packing of a graph in the form of its projection onto the matrix triangle, whose elements are recomputed by the packed operations. The long arcs can be eliminated and replaced by a combination of short-range arcs.

The following inequality is valid for the 'equivalent perturbation' M of the Cholesky decomposition (see [16]):

$$||M||_E \leq 2||\delta A||_E,$$

where δA is the perturbation of the matrix A caused by the representation of the matrix elements in the computer memory. Such a slow growth of matrix elements during decomposition is due to the fact that the matrix is symmetric and positive definite.

2. Algorithm implementation

2.1. Sequential implementation of the algorithm

In its simplest version without permuting the summation, the Cholesky decomposition can be represented as follows:

$$\begin{array}{lll}
For & i = 1, ..., n & Do: \\
 & s := a_{ii} & \\
 & For \quad k = 1, ..., i-1 & Do: \\
 & \qquad s := s - a_{ik}a_{ik} & \\
 & EndDo & \\
 & a_{ii} := \sqrt{s} & \\
 & For \quad j = i+1, ..., n & Do: \\
 & \qquad s := a_{ij} & \\
 & \qquad For \quad k = 1, ..., i-1 & Do: \\
 & \qquad\qquad s := s - a_{ik}a_{kj} & \\
 & \qquad EndDo & \\
 & \qquad a_{ij} := s/a_{ii} & \\
 & EndDo & \\
EndDo & &
\end{array}$$

Here s is a double precision variable of the accumulation mode, and the product $a_{ik}a_{kj}$ is assumed to be performed in double precision (as in Fortran intrinsic function 'dprod') provided that original matrix A is stored in a single precision. It means that the entire loop on $k = 1, ..., i-1$ can be implemented as a separate 'inner product' function with the use of accumulation mode.

In addition, the alternative 'outer product' version of the decomposition by Golub and Van Loan [11] can be mentioned. But in this research we focus on the 'inner product' version to make it possible to exploit a higher precision variable s in accumulation mode.

A block version of the Cholesky algorithm is usually implemented in such a way that the scalar operations in its sequential versions are replaced by the corresponding block-wise operations instead of using the loop unrolling and reordering techniques.

In order to ensure the locality of memory access in the Cholesky algorithm, the original matrix A and factors L and U should be stored in the lower triangle by rows (or in the upper

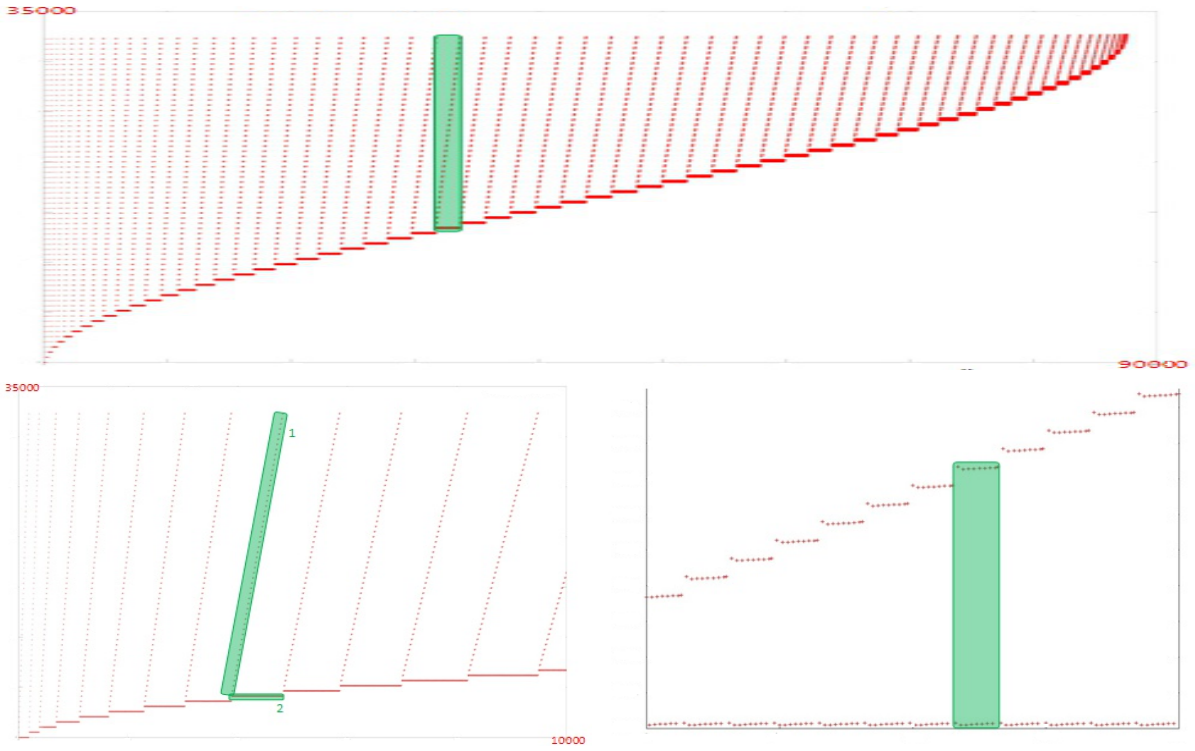

Figure 2. Implementation of the sequential Cholesky decomposition algorithm. A general memory-access profile

one by columns) to exploit the sequential access to memory and to improve paging and cache memory usage.

2.2. Structure of memory access and locality estimation

A memory access profile is a sequence of virtual memory addresses (vertical axis) presented in the order they are accessed by the program (horizontal axis). It is illustrated in Fig. 2 for the implementation of the Cholesky algorithm with a single working array. In this profile, only the elements of this array are referenced. The above-illustrated implementation consists of a single main stage; in its turn, this stage consists of a sequence of similar iterations. The example of such iteration is highlighted in green on the top plot. The left-bottom plot gives the fragment of the several first Cholesky decomposition steps. It consists of two parts of different locality with and without of data re-usage. The right-bottom plot gives a more detailed view of a single step.

From Fig. 2 it follows that at each ith iteration, all the addresses starting with a certain one are being used and the address of the first processed element increases with increasing i. We should also note that at each iteration the number of memory accesses increases up to the middle of the algorithm; after that, this number decreases down to the end of the algorithm. This fact allows us to conclude that the data processed by the algorithm are used nonuniformly and that many iterations (especially at the beginning of the process) use a large amount of data, which decreases the memory access locality. In this case, the structure of iterations is the main factor influencing the memory access locality.

The more detailed study of the locality can be found in [3], as well as 'quantitative' estimation of locality on the basis of daps and cvg values.

2.3. Approaches and features of parallel implementations

The analysis of algorithms structure allows making a conclusion that a large variety of its parallel implementations can be proposed. For example, in the version specified in Section 2.1 only the inner loop over j is parallel. Nevertheless, a simple parallelization technique causes a large number of data transfer between the processors at each step of the outer loop; this number is almost comparable with the number of arithmetic operations. Hence, it is reasonable to partition the computations into blocks with the corresponding partitioning of the data arrays before the allocation of operations and data between the processors of the computing system in use.

A relatively successful decision depends on the features of a particular computer system. If the nodes of a multiprocessor computer are equipped with pipeline accelerators, it is reasonable to compute several dot products at once in parallel. Such a possibility exists in the case of programmable logic devices; in this case, however, the arithmetic performance is limited by a slow sequential square-root operation. In principle, it is possible to apply the so-called 'skew' parallelism; however, this approach is not used in practice because of complexity in the control structure of the algorithms implementation.

2.4. Scalability and dynamic characteristics of the algorithm implementation

The scalability analysis was performed on the Lomonosov supercomputer [14] installed at the Research Computing Center of Moscow State University. For the experiments, the Intel Xeon X5570 processors with 94 Gflops peak performance and the Intel compiler with -O2 option were used. For our experiments we used the ScaLAPACK implementation for the Cholesky decomposition from the MKL library (the pdpotrf method).

Figure 3 illustrates the performance of the parallel implementation of the Cholesky algorithm. The number of processors were taken from 4 to 256, the orders of matrices – from 1024 to 5120. The maximal performance achieved in this experiments was about 2.5 Tflops. We can conclude that the performance of the algorithm increases with the increase of the problem size, and otherwise, for a reasonable problem size, the performance increases with the increase of the number of processors used. Furthermore, the cache effects can be observed by the graph curvature.

Figure 4 illustrates the Cholesky decomposition implementation efficiency (the case of lower triangular matrices) for the matrix order 80000 by using 256 processes. It is shown on the top figure that during the runtime of the program the processor usage level is about 50%. That is a quite good result for programs executed with no use of the Hyper Threading technology. The bottom figure illustrates the variation of FLOPS performance during execution time.

A more detailed analysis of dynamic characteristics can be found in [4]. The diagrams for the number of L1/L3 cache-misses, the number of memory read/write operations, the Infiniband network usage in bytes and packages are presented. The details on collecting the dynamic characteristics of the programm under investigation can be found in [2].

The data obtained from the monitoring system allows one to make a conclusion that the program under the study was operating in an efficient and stable manner. The memory and communication environment usage is intensive, which can lead to an efficiency reduction with the increase of the matrix order or the number of processors in use.

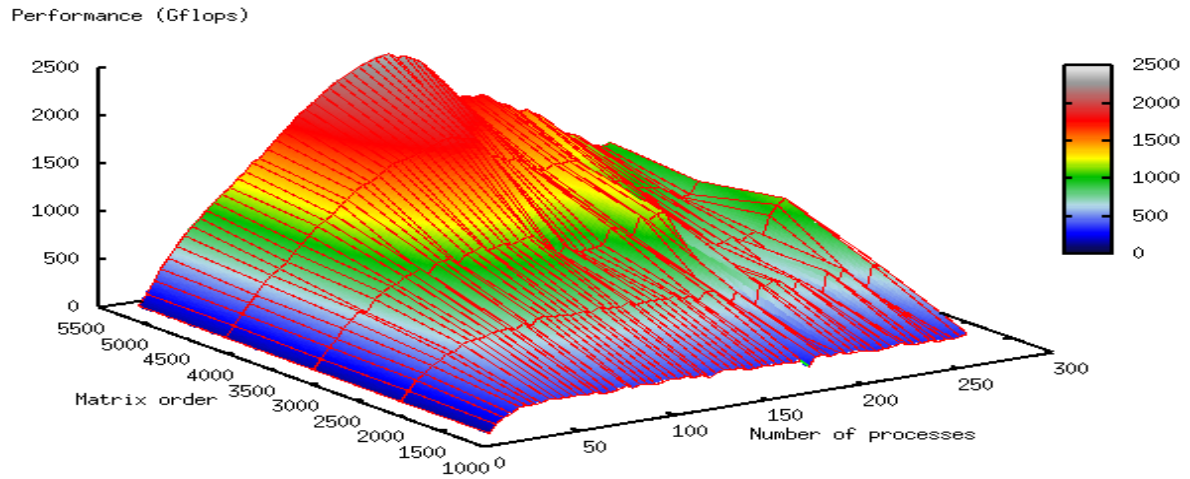

Figure 3. A parallel implementation of the Cholesky algorithm. Performance variation vs. the number of processors and the matrix order

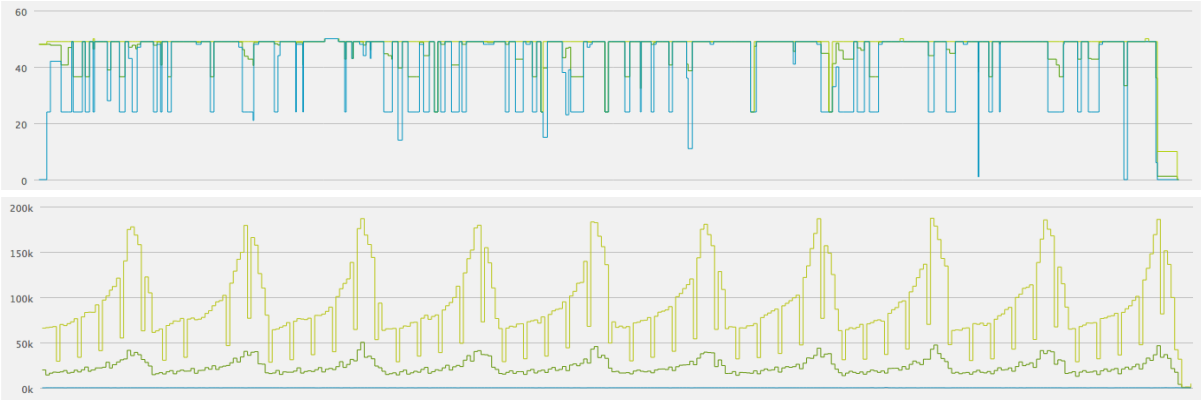

Figure 4. CPU usage diagram (top) and FLOPS performance (bottom) for the Cholesky decomposition execution time

2.5. Computer architectures and existing implementations

Analyzing the performance of the ScaLAPACK library on supercomputers, we can conclude that for large values of the matrix order n the data exchanges reduce the execution efficiency but to a smaller extent than the non-optimality in the organization of computations within a single node. At the first stages, hence, it is necessary to optimize not a block algorithm but the subroutines used on individual processors, such as the dot Cholesky decomposition, matrix multiplications, etc.

Generally speaking, the efficiency of the standard Cholesky algorithm cannot be high for parallel computer architectures. This fact can be explained by the following property of its information structure: the square-root operations are the bottleneck of the Cholesky algorithm in comparison with the division operations or with the $a - bc$ operations, since these operations can easily be pipelined or distributed among the nodes. In order to enhance the performance efficiency on parallel computers, hence, it is reasonable to use not the original Cholesky algorithm but its well-known modification without square-root operations in the form LDL^T.

The dot version of Cholesky algorithm is implemented in most of linear algebra libraries, such as LINPACK, LAPACK, ScaLAPACK, etc.

Only in few of libraries, the accumulation mode is implemented to reduce the effect of round-off errors. In order to save computer memory, a packed representation of the matrices A and L is also used in the form of one-dimensional arrays.

Unfortunately, in the most of libraries the dot Cholesky implementations are based on its LINPACK implementation utilizing the conventional BLAS library. The dot product is computed in BLAS with no accumulation mode that may substantially reduce the accuracy of computations.

It is interesting to note that the original Cholesky decomposition is available in a majority of libraries, whereas the LDU-decomposition algorithm without square-root operations is used only in specific cases (for example, for tridiagonal matrices), when the number of diagonal elements is comparable with the number of off-diagonal ones.

Conclusion

The detailed analysis of the Cholesky decomposition algorithm properties has been presented. It has been shown that the comprehensive theoretical and practical investigation is important to construct the efficient implementation of the algorithm.

The results described in all sections except 2.4 were obtained in the Moscow State University with the financial support of the Russian Science Foundation (agreement No. 14-11-00190). The research presented in section 2.4 was supported by the Russian Foundation for Basic Research (No. 16-07-01003). Numerical experiments were performed in the Supercomputing Center of Lomonosov Moscow State University [14].

References

1. Voevodin, V., Antonov, A., Dongarra, J.: AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. Supercomputing Frontiers and Innovations, Vol. 2, No. 1. Pp. 4–18 (2015). DOI: 10.14529/jsfi150101.

2. Antonov, A., Voevodin, Vl., Voevodin, Vad., Teplov, A.: A Study of the Dynamic Characteristics of Software Implementation as an Essential Part for a Universal Description of Algorithm Properties. 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing Proceedings. Pp. 359–363 (2016).

3. AlgoWiki: an open encyclopedia of algorithms properties. http://algowiki-project.org.

4. AlgoWiki: Cholesky decomposition. http://algowiki-project.org/en/Cholesky_decomposition.

5. Banachiewicz, T.: Principles d'une nouvelle technique de la méthode des moindres carrês. Bull. Intern. Acad. Polon. Sci. A. 134–135 (1938).

6. Banachiewicz, T.: Méthode de résoltution numérique des équations linéaires, du calcul des déterminants et des inverses et de réduction des formes quardatiques. Bull. Intern. Acad. Polon. Sci. A. 393–401 (1938).

7. Benoit, Commandant: 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 2, 67–77 (1924).

8. Brezinski, C.: André-Louis Cholesky. Springer-Verlag GmbH, 311 p. (2014).

9. Cholesky, A.-L.: Sur la résolution numérique des systèmes déquations linéaires La SABIX, Bulletins déja publiés, Sommaire du bulletin n. 39. 81–95 (2005) https://sabix.revues.org/529.

10. Faddeev, D.K., Faddeeva, V.N.: Computational Methods of Linear Algebra. Freeman (1963).

11. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd ed. Johns Hopkins Univ. Press, Baltimore, MD, USA (1996).

12. Kaporin, I.E.: High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition. Numer. Lin. Algebra Appl. 5(6), 483–509 (1998). DOI: 10.1002/(SICI)1099-1506(199811/12)5:6<483::AID-NLA156>3.0.CO;2-7.

13. Kaporin, I.E., Konshin, I.N.: A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems. Numer. Lin. Algebra Appl. 9(2), 141–162 (2002). DOI: 10.1002/nla.260.

14. Sadovnichy, V., Tikhonravov, A., Voevodin, Vl., Opanasenko, V.: "Lomonosov": Supercomputing at Moscow State University. In Contemporary High Performance Computing: From Petascale toward Exascale (Chapman & Hall/CRC Computational Science), Boca Raton, USA, CRC Press. 283–307 (2013).

15. Voevodin, V.V.: Computational Foundations of Linear Algebra. Nauka, Moscow (1977) (in Russian).

16. Voevodin, V.V., Kuznetsov, Yu.A.: Matrices and Computations. Nauka, Moscow (1984) (in Russian).

17. Voevodin, V.V., Voevodin, Vl.V.: Parallel Computing. BHV-Petersburg, St. Petersburg (2002) (in Russian).