Algebraic and algorithmic classification of matrix algorithms

G. Malaschonok,

National University Kyiv-Mohyla Academy, Kyiv, Ukraine

March 30, 2021

Outline of the talk

Problems of Extrim Scale Computing

Outline of the talk

Outline of the talk

Outline of the talk

Outline of the talk

Problems of Extrim Scale Computing

Three Classes of Matrx Algorithms

Rational Matrx Algorithms - MA_1

Irrational (expressed in radicals) Matrx Algorithms - MA_2

Iterative (not expressed in radicals) Matrx Algorithms - MA_3

Outline of the talk

- Problems of Extrim Scale Computing
- Three Classes of Matrx Algorithms
- Rational Matrx Algorithms - MA_1
- Irrational (expressed in radicals) Matrx Algorithms - MA_2
- Iterative (not expressed in radicals) Matrx Algorithms - MA_3
- Dynamic Decentralized Computation Control Scheme

Abstract

Due to the growing size of the matrices used in applications, it is useful to carefully distinguish between some groups of matrix algorithms. We propose to use algebraic classification as the main way to group matrix algorithms. From an algorithmic point of view, we propose to highlight the class of block-recursive algorithms. These algorithms make it possible to ensure a uniform load of a computing cluster, to solve the problem of protecting against failure of its individual nodes, and, in addition, they have the complexity of matrix multiplication.

Introduction

Appearance of the supercomputer system with hundreds of thousands of cores poses many problems. The three main ones are

1) uniform load of equipment,

(See at: Dongarra J. With Extrim Scale Computing the Rules Have Changed. In Mathematical Software. ICMS 2016, 5th International Congress, Procdistributed memoryeedings (G.-M. Greuel, T. Koch, P. Paule, A. Sommese, eds.), Springer, LNCS, volume 9725, pp. 3-8, 2016)

Introduction

Appearance of the supercomputer system with hundreds of thousands of cores poses many problems. The three main ones are

1) uniform load of equipment,

2) control the growth of the error of numbers during calculations

(See at: Dongarra J. With Extrim Scale Computing the Rules Have Changed. In Mathematical Software. ICMS 2016, 5th International Congress, Procdistributed memoryeedings (G.-M. Greuel, T. Koch, P. Paule, A. Sommese, eds.), Springer, LNCS, volume 9725, pp. 3-8, 2016)

Introduction

Appearance of the supercomputer system with hundreds of thousands of cores poses many problems. The three main ones are

1) uniform load of equipment,

2) control the growth of the error of numbers during calculations

3) protection against possible physical failures of individual processors.

(See at: Dongarra J. With Extrim Scale Computing the Rules Have Changed. In Mathematical Software. ICMS 2016, 5th International Congress, Procdistributed memoryeedings (G.-M. Greuel, T. Koch, P. Paule, A. Sommese, eds.), Springer, LNCS, volume 9725, pp. 3-8, 2016)

Introduction: first problem

The first problem is the uniform load of equipment.

In the paper "Reazul Hoque, Thomas Herault, George Bosilca, **Jack Dongarra**: Dynamic Task Discovery in PaRSEC- A data-flow task-based Runtime. Proc. ScalA17, Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems,November 12-17, 2017, Denver, CO, USA (2017)" the authors

presented a new task insertion extension for their system PaRSEC, Dynamic Task Discovery (DTD), supporting shared and distributed memory environments.

We suggest another dynamic control scheme (Drop-Amine-Pine) for a parallel computing process.
But it can be used only for block-recursive algorithms. In such algorithms, independent separate subtasks apply to blocks, so it is easy to organize decentralized control of the entire computational process.

Introduction: second problem

The second problem is the accumulation of errors during calculations. The larger the matrix size, the more error can accumulate.

We can offer only two ways to solve this problem. The first way is to increase the number of bits in the machine word. The second way is to find the exact solution in rational numbers. Both solutions increase the computational complexity of the algorithm.

It is important to understand which method should be used in a specific algorithm. For this purpose, we propose to distinguish between three groups of matrix algorithms.

Let a set of matrices be given. All source numbers are rational numbers. If your algorithm uses only rational operations, then you have the opportunity to get an exact answer with respect to the input data.

Three classes of matrix algorithms

We have to change the computational paradigm acording to the class of matrix algorithms. All matrix algorithms are divided into three separate classes:

(MA_1) the rational direct matrix algorithms,

(MA_2) the irrational direct matrix algorithms (expressed in radicals),

(MA_3) the iterative matrix algorithms (not expressed in radicals).

The first class (MA_1)

The first class (MA_1) contains algorithms that use only four arithmetic operations. As a result, only rational functions can be computed. This class includes:

– an algorithm for solving systems of linear equations,
– calculating the inverse matrix, a determinant,
– a similar three-diagonal matrix,
– a characteristic polynomial,
– a generalized inverse matrix,
– a kernel of a linear operator,
– LU, LEU and LDU decompositions,
– Bruhat decomposition and so on.

The second (MA_2) class

The second (MA_2) class consists of all direct methods that did not fall into the first class.
Elements of matrices that are obtained as a result of the application of these methods cannot be obtained in the form of rational functions. But all numbers in result may be expressed in radicals.
This class includes algorithms for QR-decomposition of matrices, orthogonal calculations of a similar two-diagonal matrix, and others.

The third class (MA_3)

The third class (MA_3) consists of all remaining algorithms, in which iterative methods are used.

– The algorithms for calculating eigenvalues and eigenvectors of a matrix and

– algorithms for SVD decomposition.

It is a complete analogy with algorithms for solving algebraic equations:

– The first class – solving linear equations.

– The second class – equations of the 2,3 and 4 degrees.

– The third class – iterative algorithms for solving algebraic equations ($\geq$5).

MA1-algorithms

/

1. Recursive standard and Strassen's matrix multiplication

$$\begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix} \times \begin{pmatrix} B_0 & B_1 \\ B_2 & B_3 \end{pmatrix} + \begin{pmatrix} C_0 & C_1 \\ C_2 & C_3 \end{pmatrix} = \begin{pmatrix} D_0 & D_1 \\ D_2 & D_3 \end{pmatrix}$$

$D_0 = A_0B_0 + A_1B_2 + C_0, D_1 = A_0B_1 + A_1B_3 + C_1, D_2 = A_2B_0 + A_3B_2 + C_2, D_3 = A_2B_1 + A_3B_3 + C_3.$

2. Recursive inversion of triangular matrix

/

If $\mathcal{A} = \begin{pmatrix} A & 0 \\ B & C \end{pmatrix}$ **is invertible triangular matrix of order** 2^k **then**

$$\mathcal{A}^{-1} = \begin{pmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{pmatrix}.$$

3. Recursive Cholesky decomposition

$Chol(\mathcal{A}) = (H, H^{-1})$ is called an *Cholesky decomposition*, if $\mathcal{A} = HH^T$, $\mathcal{A} = \begin{pmatrix} A_1 & A_2 \\ A_2^T & A_3 \end{pmatrix}$, $H = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix}$. $\mathcal{A}$ is a positive definite symmetric matrix and H is a low triangle. Let $Chol(A_1) = (B, B^{-1})$. Then we can compute

$$C = A_2^T(B^{-1}) \quad \text{and} \quad F = A_3 - CC^T$$

Let $Chol(F) = (D, D^{-1})$. Then

$$H = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix} \text{ and } H^{-1} = \begin{pmatrix} B^{-1} & 0 \\ -D^{-1}CB^{-1} & D^{-1} \end{pmatrix}.$$

4. Recursive Strassen's matrix inversion

Let $\mathcal{A} = \begin{pmatrix} A_0 & A_1 \\ A_2 & A_3 \end{pmatrix}$, $\det(\mathcal{A}) \neq 0$ and $\det(A_0) \neq 0$ then the inverse matrix can be calculated as follows:

$$\mathcal{A}^{-1} = \begin{pmatrix} \mathbf{I} & -A_0^{-1}A_1 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & 0 \\ 0 & (A_3 - A_2A_0^{-1}A_1)^{-1} \end{pmatrix}$$

$$\times \begin{pmatrix} \mathbf{I} & 0 \\ -A_2 & \mathbf{I} \end{pmatrix} \begin{pmatrix} A_0^{-1} & 0 \\ 0 & \mathbf{I} \end{pmatrix} = \begin{pmatrix} M_6 & M_1M_4 \\ M_5 & M_4 \end{pmatrix}$$

We have denoted here $M_0 = -A_0^{-1}$, $M_1 = M_0A_1$, $M_2 = A_2M_0$, $M_3 = M_2A_1$, $M_4 = (A_3 + M_3)^{-1}$, $M_5 = -M_4M_2$, $M_6 = M_1M_5 - M_0$.

MA_1. 5-a. Recursive computation of the adjoint and kernel: 1 of 2

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

$$A_{ext}(M_{11}, d_0) = (A_{11}, S_{11}, E_{11}, d_{11}).$$

$$M^1_{12} = \frac{A_{11} M_{12}}{d_0}, M^1_{21} = -\frac{M_{21} Y_{11}}{d_0}, M^1_{22} = \frac{M_{22} d_{11} - M_{21} E^T_{11} M^1_{12}}{d_0}.$$

$$A_{ext}(\bar{I}_{11} M^1_{12}, d_{11}) = (A_{12}, S_{12}, E_{12}, d_{12}), A_{ext}(M^1_{21}, d_{11}) = (A_{21}, S_{21}, E_{21}, d_{21}).$$

$$M^2_{22} = -\frac{A_{21} M^1_{22} Y_{12}}{(d_{11})^2}, \ d_s = \frac{d_{21} d_{12}}{d_{11}}.$$

$$A_{ext}(\bar{I}_{21} M^2_{22}, d_s) = (A_{22}, S_{22}, E_{22}, d_{22}).$$

$$M^2_{11} = -\frac{S_{11} Y_{21}}{d_{11}}, M^2_{12} = \frac{\left(\frac{\frac{S_{11} E^T_{21} A_{21}}{d_{11}} M^1_{22} - I_{11} M^1_{12} d_{21}}{d_{11}} \right) Y_{12} + S_{12} d_{21}}{d_{11}}, M^3_{12} = -\frac{M^2_{12} Y_{22}}{d_s},$$

MA_1. 5-b. Recursive computation of the adjoint and kernel: 2 of 2

$$M_{22}^3 = S_{22} - \frac{I_{21} M_{22}^2 Y_{22}}{d_s},\ A^1 = A_{12}A_{11},\ A^2 = A_{22}A_{21},$$

$$L = \left(\frac{A^1 - \frac{I_{11} M_{12}^1 E_{12}^T A^1}{d_{11}}}{d_{11}}\right) d_{22}, \quad P = \frac{A^2 - \frac{I_{21} M_{22}^2 E_{22}^T A^2}{d_s}}{d_{21}},$$

$$F = -\frac{\left(\frac{S_{11} E_{21}^T A_{21}}{d_{11}}\right) d_{22} + \frac{M_{12}^2 E_{22}^T A^2}{d_s}}{d_{21}}, \quad G = -\frac{\left(\frac{M_{21} E_{11}^T A_{11}}{d_0}\right) d_{12} + \frac{M_{22}^1 E_{12}^T A^1}{d_{11}}}{d_{11}},$$

$$A = \begin{pmatrix} \frac{L+FG}{d_{12}} & F \\ \frac{PG}{d_{12}} & P \end{pmatrix}, S = \begin{pmatrix} \frac{M_{11}^2 d_{22}}{d_{21}} & M_{12}^3 \\ \frac{S_{21} d_{22}}{d_{21}} & M_{22}^3 \end{pmatrix}, E = \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix},\ d = d_{22}.$$

Then

$$A_{ext}(M, d_0) = (A, S, E, d_{22}).$$

$$I_{ij} = E_{ij} E_{ij}^T, \bar{I}_{ij} = \mathbf{I} - I_{ij}, Y_{ij} = E_{ij}^T S_{ij} - d_{ij}\mathbf{I}, \quad i, j \in 1, 2.$$

MA_1. New paradigm for NA_1 class

All NA_1 class algorithms have a complexity of $\sim n^3$ (or $\sim n^\beta$) in operations on matrix elements using standard matrix multiplication (or fast matrix multiplication with n^β operations). For numerical matrices, one can obtain exact solutions by spending another n^2 (or n^α, or n) bit operations for standard multiplication of numbers (or fast multiplication of numbers (with complexity n^α), or the use of finite fields).

In all these algorithms, we obtain an exact solution and the question of the accumulation of error does not arise here.

NA_2 class

Aalgebraic constructions of class NA_1 cannot be applied to algorithms of class MA_2. The main task is the formulation of algorithms in a block-recursive form.

We know today two such block-recursive algorithms. These are the algorithms of orthogonal decomposition: the QR-algorithm and the first part of the SVD-algorithm (which ends with the construction of a similar tridiagonal matrix).

We present only a QR-algorithm. We propose another way of presenting the algorithm of Schonhage and we calculate the exact number of operations.

MA_2. QR-algorithm 1

Let A be a matrix over a real numbers. It is required to find the upper triangular matrix R and the orthogonal Q matrix such that $A = QR$.

Consider the case of a 2×2 matrix. The desired decomposition $A = QR$ has the form:

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a & b \\ 0 & d \end{pmatrix},$$

where the numbers s and c satisfy the equation $s^2 + c^2 = 1$. If $\gamma = 0$ then we can set $c = 1,\ s = 0$. If $\gamma \neq 0$, then we get $\Delta = \alpha^2 + \gamma^2 > 0$, $c = \alpha/\sqrt{\Delta}, \quad s = \gamma/\sqrt{\Delta}$. We denote such a matrix Q by $g_{\alpha,\gamma}$.

MA_2. QR-algorithm 2

Let the matrix A be given, its elements (i, j) and $(i+1, j)$ be α and γ, and all the elements to the left be zero: $\forall(s < j) : (a_{i,s} = 0)$ & $(a_{i+1,s} = 0)$.

Let $G_{i,j} = \text{diag}(I_{i-1}, g_{\alpha,\gamma}, I_{n-i-1})$. (there are called Givens matrices).
Then the matrix $G_{i,j}A$ differs from A only in two rows i and $i+1$, but all the elements to the left of the column j remain zero, and in the column j in $i+1$ line will be 0.

SEQUANTIAL ALGORITHM

(1). First we reset the elements under the diagonal in the left column:

$$A_1 = G_{1,1} G_{2,1} ... G_{n-2,1} G_{n-1,1} A$$

(2). Then we reset the elements that are under the diagonal in the second column:

$$A_2 = G_{2,2} G_{3,2} ... G_{n-2,2} G_{n-1,2} A_1$$

(k). Denote $G_{(k)} = G_{k,k} G_{k-1,k} ... G_{n-2,k} G_{n-1,k}$, $k = 1, 2, .., n-1$. Then, to calculate the elements of the k th column, we need to obtain the product of matrices

$$A_k = G_{(k)} A_{k-1}.$$

(n-1). At the end of the calculation, the element in the $n-1$ column will be reset: $A_{n-1} = G_{(n-1)} A_{n-2} = G_{n-1,n-1} A_{n-2}$.

MA_2. QR_G decomposition 4

Let a matrix M of size $2n \times 2n$ be divided into blocks: $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$.

There are three stages in this algorithm.
(Stage 1). The QR_G decomposition of the block C:

$$C = Q_1 C_1, \; M_1 = \mathrm{diag}(I, Q_1) M = \begin{pmatrix} A & B \\ C_1 & D_1 \end{pmatrix}.$$

(Stage 2). The cancellation of a parallelogram composed of two triangular blocks: the lower triangular part A^L of the block A and the upper triangular part C_1^U of the block C_1. Denote the upper triangular matrix A_1 and annihilating matrix Q_2:

$$Q_2 \begin{pmatrix} A \\ C_1 \end{pmatrix} = \begin{pmatrix} A_1 \\ 0 \end{pmatrix}, \; M_2 = Q_2 M_1 = \begin{pmatrix} A_1 & B_1 \\ 0 & D_2 \end{pmatrix}.$$

(Stage 3). The QR_G decomposition of the D_2 block: $D_2 = Q_3 D_3$.

$$R = \mathrm{diag}(I, Q_3) M_2 = \begin{pmatrix} A_1 & B_1 \\ 0 & D_3 \end{pmatrix}.$$

MA_2. QR-algorithm 5

As a result, we get:

$$M = Q^T R, \quad Q = \operatorname{diag}(I, Q_3) Q_2 \operatorname{diag}(I, Q_1).$$

Since the first and third stages are recursive calls of the QR_G procedures, it remains to describe the parallelogram cancellation procedure. Let's call it a QP decomposition.

MA_2. QP-decomposition 6

We are looking for the factorization $M = \begin{pmatrix} A \\ B^U \end{pmatrix} = QP = Q \begin{pmatrix} A^U \\ 0 \end{pmatrix}$.

M have dimensions $2n \times n$, B^U and A^U have an upper triangular shape, Q is the orthogonal matrix:

$$M = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \\ B_1^U & B_2 \\ 0 & B_4^U \end{pmatrix} = Q \begin{pmatrix} A_5^U & A_6 \\ 0 & A_7^U \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

We can consistently perform columns invalidation (using Givens matrices) by traversing column elements from bottom to top and traversing columns from left to right.

For the block-recursive procedure we can break the parallelogram into 4 equal parts. To cancel each of them, we will simply call the parallelogram cancellation procedure 4 times: the bottom left (P_{ld}), simultaneously the top left (P_{lu}) and the bottom right (P_{rd}), and then the top right (P_{ru}).

MA_2. QR_G-algorithm 7

The corresponding orthogonal Givens matrices of size $n \times n$ are denoted Q_{ld}. Q_{lu}. Q_{rd} and Q_{ru}. Let

$$\bar{Q}_{ld} = \mathrm{diag}(I_{n/2}, Q_{ld}, I_{n/2}), \quad \bar{Q}_{ru} = \mathrm{diag}(I_{n/2}, Q_{ru}, I_{n/2}),$$

As a result, we get:

$$Q = \bar{Q}_{ru}\,\mathrm{diag}(Q_{lu}, Q_{rd})\bar{Q}_{ld}$$

The number of multiplications of matrix blocks of size $n/2 \times n/2$ is 24. Hence the total number of operations: $Cp(2n) = 4Cp(n) + 24M(n/2)$.. Suppose that for multiplication of two matrices of size $n \times n$ you need γn^β operations and $n = 2^k$, then we get:
$Cp(2^{k+1}) = 4Cp(2^k) + 24M(2^{k-1}) = 4^k\,Cp(2^1) + 24\gamma \sum_{i=0}^{k-1} 4^{k-i-1}2^{i\beta} = 24\gamma(n^2/4)\frac{2^{k(\beta-2)}-1}{2^{(\beta-2)}-1} + 6n^2 = 6\gamma\frac{n^\beta - n^2}{2^\beta - 4} + 6n^2$

$$Cp(n) = \frac{6\gamma n^\beta}{2^\beta(2^\beta - 4)} + \frac{3n^2}{2}(1 - \frac{\gamma}{2^\beta - 4})$$

MA_2. The complexity of QR_G decomposition algorithm 8

Let us estimate the number of operations $C(n)$ in this block-recursive decomposition algorithm, assuming that the complexity of the matrix multiplication is $M(n) = \gamma n^{\beta}$, the complexity of canceling the parallelogram is $Cp(n) = \delta n^{\beta}$, where δ, β, γ are constants, $\delta = \frac{6\gamma}{2^{\beta}(2^{\beta}-4)}$ and $n = 2^k$: $C(n) = 2C(n/2) + Cp(n) + 6M(n/2) = 2C(2^{k-1})$ $+Cp(2^k) + 6M(2^{k-1}) =$

$$= \frac{\gamma 6(2^{\beta} - 3)}{(2^{\beta} - 4)(2^{\beta} - 2)}(n^{\beta} - \frac{2n}{2^{\beta}})$$

MA_2. New paradigm for the MA_2-class (MA_3)

These two MA_2-class algorithms have a complexity of $\sim n^3$ (or $\sim n^\beta$) in operations on matrix elements using standard matrix multiplication (or fast matrix multiplication with $\sim n^\beta$ operations).

We cannot avoid rounding errors. Therefore, it is necessary to be able to control the calculation error **by increasing the number of digits for storing numbers**.

Control of calculation errors in MA_3-class requires special additional studies.

Dynamic algorithms

Parallel algorithms for distributed memory are divided into two classes: static and dynamic.

Static algorithms are those in which all data transfers between nodes can be scheduled before the start of the calculations.

Dynamic matrix algorithms are based on matrix block-recursive algorithms.

In such algorithms, the matrix is recursively divided into blocks.

A block-recursive algorithm is again applied to each of the blocks.

This happens as long as the blocks remain large enough. When the block size becomes small enough, the sequential algorithms are applied to the blocks.

This limit for the size of a small block depends on the physical

The dynamic algorithm has three stages

First stage. This is the initial construction of the connections tree for computational nodes. The large blocks are sent from the root node to a child along with lists of free nodes. From these child nodes, data is sent further, but already with smaller blocks and corresponding parts of the list of free nodes.

Second stage. It occurs when either all the free nodes have received their subtasks, or when the size of the blocks has decreased to a certain boundary, which is predetermined. The tree of connections is constructed and the calculations are started on leaf vertices.

The third stage. At this stage, the results are returned from leaf vertices to the root vertex. The result of the main task is obtained at root vertex and the calculations are completed.

Automatic redistribution of subtasks

Dynamic control involves the automatic redistribution of subtasks from overloaded nodes to free nodes.

For this purpose, a scheme is provided for transmitting information about free nodes and information about overloaded nodes.

Both streams of information are transmitted along the tree towards the root vertex until they meet at a certain node. After this, the information about free vertices is redirected to the overloaded vertices.

The largest subtasks from the overloaded nodes are transmitted to the free nodes.

Protection scheme in case of a failure of a node

Protection scheme in case of a failure of a node during calculations.

Let node 1 send a subtask S to node 2. Let node 2 fail and the failure message cames to node 1. Node 1 will mark this subtask S as unsolved and return it to the list of unsolved subtasks.

All operations of transferring results from child nodes to node 2 are canceled.

No other action is required.

The computational process continues on all other nodes without changes.

This scheme was implemented in the Java programming language using the OpenMPI and MathPartner (http://mathpar.cloud.unihub.ru) packages. Its was tested on the matrix multiplication and inversion.

Conclusion

We proposed a new classification of matrix computational algorithms, which decomposes all algorithms into three classes: rational, irrational and iterative.

We discribed the new computational paradigm: using of the block-recursive matrix algorithms for creating parallel programs that are designed for supercomputers with distributed memory and dynamic decentralized control of the computational process.

We have shown many examples of such algorithms. We proposed a dynamic decentralized computation control scheme.

Thanks for your attention

- Dongarra J. *With Extrim Scale Computing the Rules Have Changed.* In Mathematical Software. ICMS 2016, 5th International Congress, Procdistributed memoryeedings (G.-M. Greuel, T. Koch, P. Paule, A. Sommese, eds.), Springer, LNCS, volume 9725, pp. 3-8, (2016)
- *Strassen V. Gaussian Elimination is not optimal.* Numerische Mathematik. V. 13, Issue 4, 354–356 (1969)
- Malaschonok G.I. *Solution of a system of linear equations in an integral domain,* Zh. Vychisl. Mat. i Mat. Fiz. V.23, No. 6, 1983, 1497-1500, Engl. transl.: USSR J. of Comput. Math. and Math. Phys., V.23, No. 6, 497-1500. (1983)
- G.I. Malaschonok. *Algorithms for the solution of systems of linear equations in commutative rings.* Effective methods in Algebraic Geometry, Progr. Math., V. 94, Birkhauser Boston, Boston, MA, 1991, 289-298. (1991)
- G.I. Malaschonok. *Algorithms for computing determinants in commutative rings.* Diskret. Mat., 1995, Vol. 7, No. 4, 68-76. Engl. transl.: Discrete Math. Appl., Vol. 5, No. 6, 557-566 (1995).

- Malaschonok G. *Recursive Method for the Solution of Systems of Linear Equations.* Computational Mathematics. A. Sydow Ed, Proceedings of the 15th IMACS World Congress, Vol. I, Berlin, August 1997), Wissenschaft & Technik Verlag, Berlin, 475-480. (1997)
- Malaschonok G. *Effective Matrix Methods in Commutative Domains*, Formal Power Series and Algebraic Combinatorics, Springer, Berlin, 506-517. (2000)
- Malaschonok G. *Matrix computational methods in commutative rings.* Tambov, TSU, 213 p. (2002)
- Akritas A.G., Malaschonok G.I. *Computation of Adjoint Matrix.* Computational Science, ICCS 2006, LNCS 3992, Springer, Berlin, 486-489.(2006)
- Malaschonok G. *On computation of kernel of operator acting in a module* Vestnik Tambovskogo universiteta. Ser. Estestvennye i tekhnicheskie nauki [Tambov University Reports. Series: Natural and Technical Sciences], vol. 13, issue 1,129-131 (2008)
- Malaschonok G. *Fast Generalized Bruhat Decomposition.* Computer

- Malaschonok G. *On fast generalized Bruhat decomposition in the domains.* Tambov University Reports. Series: Natural and Technical Sciences. V. 17, Issue 2, P. 544-551. (http://parca.tsutmb.ru/src/MalaschonokGI17_2.pdf) (2012)
- Malaschonok G. *Generalized Bruhat decomposition in commutative domains.* Computer Algebra in Scientific Computing. CASC'2013. LNCS 8136, Springer, Heidelberg, 2013, 231-242. DOI 10.1007/978-3-319-02297-0_20. arxiv:1702.07248 (2013)
- Malaschonok G., Scherbinin A. *Triangular Decomposition of Matrices in a Domain.* Computer Algebra in Scientific Computing. LNCS 9301, Springer, Switzerland, 2015, 290-304. DOI 10.1007/978-3-319-24021-3_22. arxiv:1702.07243 (2015)
- G. Malaschonok and E. Ilchenko, "Recursive Matrix Algorithms in Commutative Domain for Cluster with Distributed Memory," 2018 Ivannikov Memorial Workshop (IVMEM), Yerevan, Armenia, 2018, pp. 40-46, doi: 10.1109/IVMEM.2018.00015.

- Gennadi Malaschonok. Recursive Matrix Algorithms, Distributed Dynamic Control, Scaling, Stability // Proc. of 12th Int. Conf. on Comp. Sci. and Information Technologies (CSIT-2019). September 23-27, 2019, Yerevan.
- G.I.Malashonok, A.A. Sidko. Parallel computing on distributed memory: OpenMPI, Java, Math Partner. Textbook. Kyiv: NaUKMA, 2020. - 266 p. ISBN 978-617-7668-14-4
- G. Malashonok A. Ivaskevych. Static block-recursive Kholetsky algorithm for a distributed memory cluster. Scientific notes of NaUKMA. Computer Science. T.3. 2020 p. 114-120.
- Gennadi Malaschonok. LDU-factorization. E-print 2011.04108, p.1-16. arXiv:2011.04108