Algebraic classification of matrix algorithms

G. Malaschonok,

National University Kyiv-Mohyla Academy, Kyiv, Ukraine

Polynomial Computer Algebra, April 2021

Outline of the talk

Problems of Extrim Scale Computing

Outline of the talk

Problems of Extrim Scale Computing

Calculation example: Cholesky decomposition (demonstrating the main problem)

Outline of the talk

Problems of Extrim Scale Computing

Calculation example: Cholesky decomposition (demonstrating the main problem)

Three Classes of Matrx Algorithms

Outline of the talk

- Problems of Extrim Scale Computing
- Calculation example: Cholesky decomposition (demonstrating the main problem)
- Three Classes of Matrx Algorithms
- Examples of these algorithms

Abstract

Abstract

The widespread use of supercomputers for solving practical problems with numerical matrices of large sizes requires a revision of the matrix algorithms.

Symbolic algorithms are becoming popular.

The new classification of matrix algorithms into three classes is discussed.

The special role of recursive matrix algorithms and matrix algorithms for commutative domains is emphasized.

Introduction

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

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

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) control the growth of the error of numbers during calculations

1) uniform load of equipment,

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)

Investigation of accuracy: Cholesky decomposition

Let $\mathcal{A} = \begin{pmatrix} \alpha & \beta \\ \beta^T & \gamma \end{pmatrix}$ be a positive definite symmetric matrix, then exist $L = \begin{pmatrix} a & 0 \\ b & c \end{pmatrix}$ and $L^{-1} = \begin{pmatrix} a^{-1} & 0 \\ -c^{-1}ba^{-1} & c^{-1} \end{pmatrix}$: $\mathcal{A} = LL^T$. So

$$\begin{pmatrix} \alpha & \beta \\ \beta^T & \gamma \end{pmatrix} = \begin{pmatrix} a & 0 \\ b & c \end{pmatrix} \begin{pmatrix} a^T & b^T \\ 0 & c^T \end{pmatrix} = \begin{pmatrix} aa^T & ab^T \\ ba^T & bb^T + cc^T \end{pmatrix}.$$

The mapping $(L, L^{-1}) = Chol(\mathcal{A})$ with blocks :

1) $aa^T = \alpha$ (recursive step $(a, a^{-1}) = Chol(\alpha)$)

2) $b^T = a^{-1} * \beta$;

3) $cc^T = \gamma - bb^T$ (recursive step $(c, c^{-1}) = Chol(\gamma - bb^T)$)

is called Cholesky decomposition [16].

Example for matrix 2x2:

if $A = \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}$

with numbers α, β, γ, then

$$(L, L^{-1}) = (\begin{pmatrix} a & 0 \\ b & c \end{pmatrix}, \begin{pmatrix} 1/a & 0 \\ b/(ca) & 1/c \end{pmatrix}) = Chol(A),$$

with numbers a, b, c:

$a = \sqrt{\alpha}$,
$b = \beta/a$,
$d = \gamma - \beta^2/\alpha$,
$c = \sqrt{d}$.

Introduction: first problem

Experiments to investigate the accuracy: Cholesky decomposition

We took non-degenerate triangular matrix L and $A = LL^T$ was decomposed: $A \approx \mathcal{L}\mathcal{L}^T$. The norm (largest element in absolute value) of their difference $S = L - \mathcal{L}$ was calculated. We performed 100 experiments with random integer (within 10) matrices. The average (mean) error and largest (max) error are presented for double precision and BigDecimal.

Matrix size	4	8	16	32	64
max_{double}	$2 \cdot 10^{-13}$	10^{-10}	$3 \cdot 10^{-6}$	0.6	142
$mean_{double}$	$6 \cdot 10^{-15}$	$4 \cdot 10^{-12}$	$6 \cdot 10^{-8}$	0.01	7.9
$lg\ max_{(100)}$	-96	-93	-89	-79	-72.1
$lg\ mean_{(100)}$	-97	-95	-91	-82	-74
$lg\ max_{(500)}$	-497	-491	-489	-482	-471
$lg\ mean_{(500)}$	-498	-493	-491	-484	-472

Introduction: first problem

Table 2. Recommendations for the choice of accuracy in the Choletsky algorithm

#dec.dig.matr.cf.	3	6	12	16	20	24
Matrixsize	*Accur.*	*Accur.*	*Accur.*	*Accur.*	*Accur.*	*Accur.*
8×8	2	5	5	5	5	5
16×16	5	10	10	10	10	10
32×32	10	20	20	20	20	20
64×64	20	30	30	30	40	40
128×128	30	50	60	60	70	70
256×256	60	90	110	120	130	140
512×512	110	170	220	240	250	270
1024×1024	220	340	440	480	500	540
2048×2048	440	780	870	950	1000	1080
4096×4096	870	1550	1750	1890	2000	2150

Introduction: first problem

The accumulation of errors during calculations.

Only two ways to solve this problem:
1) to increase the number of bits in the machine word,
2) to find the exact solution in rational numbers.
Both solutions increase the computational complexity of the algorithm.

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 matrix algorithms,

(MA_2) the irrational 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 (non-iterative) methods that did not fall into the first class.
All numbers in result may be expressed in radicals.
This class includes algorithms for QR-decomposition of matrices, orthogonal calculations of a similar two- and three- 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).

The second problem of supercomputing

The second problem is the uniform load of equipment.

Coarse-grained parallelization algorithms are badly needed

In recent years, many research groups have been intensively searching for special mechanisms for extracting **large independent subtasks** from the algorithm OpenMP [20], StarPU [21], Legion [22], PaRSEC [23], OCR [24], HPX [25], SuperGlue [27], QUARK [28].

We suggest to use only block-recursive algorithms.

We suggest dynamic control runtime (Drop-Amine-Pine). It can be used for **block-recursive algorithms**.

The thired problem of supercomputing

The thired problem is protection against possible physical failures of individual processors.

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.

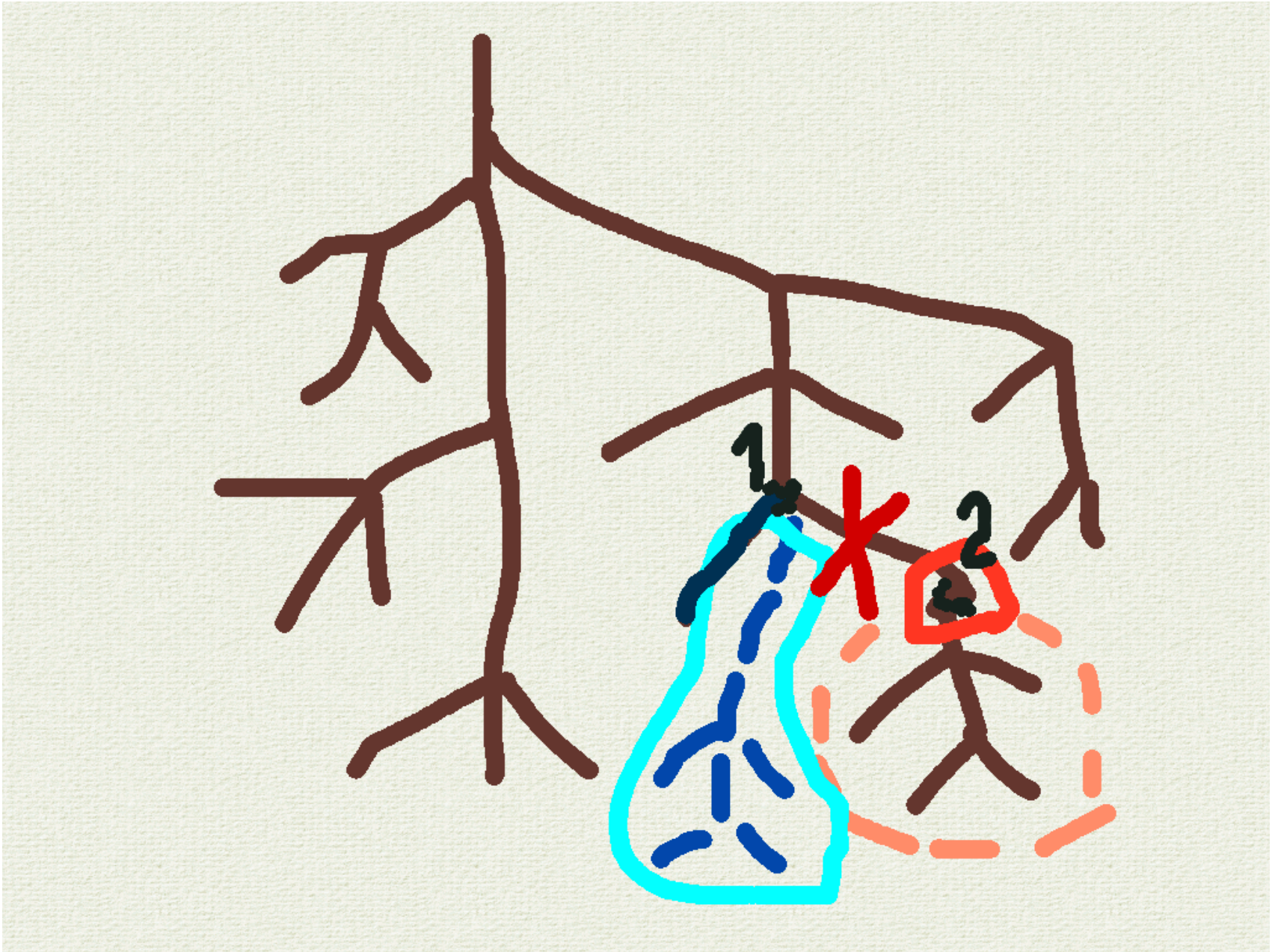

Figure 1:

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 MA_1 class

All MA_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.

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. New paradigm for the MA_2-class

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**.

MA_2. What about MA_3-class?

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

Conclusion

We proposed a new classification of matrix computational algorithms, which decomposes all algorithms into three classes: rational, irrational (expressed in radicals) 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.

We have shown many examples of such algorithms.

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

- OpenMP 4.0 Complete Specifications. (2013). http://www.openmp.org/wp-content/uploads/OpenMP4.0.0.pdf
- Emmanuel Agullo, Olivier Aumage, Mathieu Faverge, Nathalie Furmento, Florent Pruvost, Marc Sergent, and Samuel Thibault. 2014. Harnessing Supercomputers with a Sequential Task-based Runtime System. 13, 9 (2014), 1–14.
- Michael Bauer, Sean Treichler, Elliott Slaughter, and Alex Aiken. 2012. Legion: Expressing locality and independence with logical regions. In International Con- ference for High Performance Computing, Networking, Storage and Analysis, SC. https://doi.org/10.1109/SC.2012.71
- George Bosilca, Aurélien Bouteiller, Anthony Danalis, Mathieu Faverge, Thomas Hérault, and Jack Dongarra. 2013. PaRSEC: A programming paradigm exploiting heterogeneity for enhancing scalability. Computing in Science and Engineering 99 (2013), 1. https://doi.org/10.1109/MCSE.2013.98
- Jiri Dokulil, Martin Sandrieser, and Siegfried Benkner. 2016. Implementing the Open Community Runtime for Shared-Memory and