

Abstracts of Dissertations

Institute of Information and
Communication Technologies

BULGARIAN ACADEMY OF
SCIENCES



4 / 2022



COMPOSITE
NUMERICAL METHODS
AND SCALABLE TILE
ALGORITHMS

Dimitar Slavchev

СЪСТАВНИ ЧИСЛЕНИ
МЕТОДИ И
СКАЛИРУЕМИ БЛОЧНИ
АЛГОРИТМИ

Димитър Славчев

Автореферати на дисертации

Институт по информационни и
комуникационни технологии

БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

ISSN: 1314-6351

Поредицата „Автореферати на дисертации на Института по информационни и комуникационни технологии при Българската академия на науките“ представя в електронен формат автореферати на дисертации за получаване на научната степен „Доктор на науките“ или на образователната и научната степен „Доктор“, защитени в Института по информационни и комуникационни технологии при Българската академия на науките. Представените трудове отразяват нови научни и научно-приложни приноси в редица области на информационните и комуникационните технологии като Компютърни мрежи и архитектури, Паралелни алгоритми, Научни пресмятания, Лингвистично моделиране, Математически методи за обработка на сензорна информация, Информационни технологии в сигурността, Технологии за управление и обработка на знания, Грид-технологии и приложения, Оптимизация и вземане на решения, Обработка на сигнали и разпознаване на образи, Интелигентни системи, Информационни процеси и системи, Вградени интелигентни технологии, Йерархични системи, Комуникационни системи и услуги и др.

Редактори

Геннадий Агре

Институт по информационни и комуникационни технологии, Българска академия на науките
E-mail: agre@iinf.bas.bg

Райна Георгиева

Институт по информационни и комуникационни технологии, Българска академия на науките
E-mail: rayna@parallel.bas.bg

Даниела Борисова

Институт по информационни и комуникационни технологии, Българска академия на науките
E-mail: dborissova@iit.bas.bg

Настоящото издание е обект на авторско право. Всички права са запазени при превод, разпечатване, използване на илюстрации, цитирания, разпространение, възпроизвеждане на микрофилми или по други начини, както и съхранение в бази от данни на всички или част от материалите в настоящето издание. Копирането на изданието или на част от съдържанието му е разрешено само със съгласието на авторите и/или редакторите

The series Abstracts of Dissertations of the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences presents in an electronic format the abstracts of Doctor of Sciences and PhD dissertations defended in the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences. The studies provide new original results in such areas of Information and Communication Technologies as Computer Networks and Architectures, Parallel Algorithms, Scientific Computations, Linguistic Modelling, Mathematical Methods for Sensor Data Processing, Information Technologies for Security, Technologies for Knowledge management and processing, Grid Technologies and Applications, Optimization and Decision Making, Signal Processing and Pattern Recognition, Information Processing and Systems, Intelligent Systems, Embedded Intelligent Technologies, Hierarchical Systems, Communication Systems and Services, etc.

Editors

Gennady Agre

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
E-mail: agre@iinf.bas.bg

Rayna Georgieva

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
E-mail: rayna@parallel.bas.bg

Daniela Borissova

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
E-mail: dborissova@iit.bas.bg

This work is subjected to copyright. All rights are reserved, whether the whole or part of the materials is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. Duplication of this work or part thereof is only permitted under the provisions of the authors and/or editor.



BULGARIAN ACADEMY OF SCIENCES

Abstract of PhD Thesis

COMPOSITE NUMERICAL METHODS AND SCALABLE TILE ALGORITHMS

Dimitar Georgiev Slavchev

Supervisor: Prof. Svetozar Margenov

Approved by Supervising Committee:

**Assoc. Prof. Kiril Shterev
Prof. Svetoslav Markov
Assoc. Prof. Miglena Koleva
Prof. Ivan Lirkov
Prof. Pencho Marinov**



**INSTITUTE OF INFORMATION AND
COMMUNICATION TECHNOLOGIES**

**Department of Scientific Computations with Laboratory on
3D Digitization and Microstructure Analysis**

General characteristic of the dissertation

Relevance of the topic

Numerical solution of large-scale problems requires the use of high-performance computer systems, as well as specialized hardware and software – graphics cards, accelerators, high-speed communication between the system's servers, software standards and packages for communication between processor cores and servers, software packages implementing effective numerical methods and much more.

There are a number of methods for discretization of differential equations. For example the mesh method such as finite element method, boundary elements method and finite differences method. After applying such methods the problem is reduced to solving large systems linear equations. The Gaussian eliminated is the universal method for solving such problems. In the general case it has high computational complexity – $O(n^3)$, where n is the number of unknowns [34].

When the discretization of the differential equation is carried out with the boundary element method and when the finite element method is applied to non-local problems (such as the examined in this dissertation *anomalous* (fractional) diffusion) [2], the arising matrix is *dense*. One way of lowering the computational complexity of solving problems with such matrices is the hierarchical compression introduced by Hackbusch [10]. In it the *structure* of the matrix is utilized. The aim is to reduce the required memory and to improve the computational complexity. Here the term dense matrix *structure* is understood as the presence of low-rank off-diagonal blocks. This property allows the representation of the off-diagonal blocks as the product of smaller matrices. There are several types of hierarchical matrices: \mathcal{H} , \mathcal{H}^2 , Hierarchically Semi-Separable (HSS) and so on.

Overview of key results in the field

The huge progress in the capabilities of modern high-performance computing systems further underlines the role of efficient numerical methods and parallel algorithms. Supercomputer simulations are crucial for development in a number advanced areas. Examples are *in silico* molecular biology and medicine design, analysis of turbulent flows, non-destructive testing, 3D image processing, fluid dynamics and many others.

After suitable discretization, mathematical models are usually reduced to problems of linear algebra, among which central role has the solution of systems of linear algebraic equations. For this purpose, specialized software tools are developed.

In the general case variants of the Gaussian method are employed in order to solve systems of linear algebraic equations with dense matrices. Such methods

utilize the sequential elimination of the unknowns. In general, the dense matrix is thought to be homogeneous, as it is not assumed to contain zeros. The Gaussian elimination method has computational complexity $O(n^3)$. In this dissertation an alternative approach based on hierarchical compression is investigated. The goal is to reduce the computational complexity of the solution process. Here by *structure* of dense matrices we understand the existence of low-rank off-diagonal blocks. More to the point, such property is found in a matrix approximating the original. The existence of such suitable *structure* is the basis of the hierarchical compression and the arising methods for solving systems of linear algebraic equations for classes of computational mathematics problems. Hierarchical compression is introduced by Hackbusch in [10], where the so-called \mathcal{H} -matrices are studied. Other types of hierarchical compression are \mathcal{H}^2 -matrices [11] and hierarchical semi-separable matrices (HSS) [15]. A theoretical basis for HSS compression methods can be found in [33].

A large part of this dissertation is dedicated to the numerical solution of fractional diffusion problems. *Fractional* (also known as anomalous) diffusion describes non-local processes observed in different physical and social media. Unlike ordinary (local) diffusion, anomalous diffusion includes the so-called fast transitions or tunnel effects. Various examples of mathematical models have been published in the literature for processes and phenomena described by fractional diffusion. Some examples are: flows in strongly non-homogeneous porous media, superconductivity, diffusion of polymers in supercold media [4]; electrodiffusion of ions into nerve cells [14] and photon diffusion diagnostics [30]; image processing and machine learning [19]; spread of viral diseases, computer viruses and crime [6]. The *fractional* Laplace operator describes anomalous diffusion in space. There are different definitions of the *fractional* Laplacian. It is important to note that they are not equivalent. For example, in [13] the difference between integral and spectral definitions is studied (see also articles [16] and [12] and the literature in them).

Goals and objectives of the dissertation

The major goals of the thesis are:

- Comparative analysis of the performance and parallel speed-up of frequently used software packages applying direct Gaussian elimination for solving systems of linear algebraic equations with dense matrix on CPUs and accelerators (MICs).
- Analysis of the performance, parallel speed-up and accuracy of an approximate method for solving systems of linear algebraic equations based on hierarchical semi-separable compression from the software package STRUMPACK for systems with suitable *structure*.

- Development of reordering algorithms for the unknowns for systems of linear algebraic equations arising from discretization with finite element method of fractional diffusion. The reordering is aimed at improving the effectiveness of hierarchical semi-separable compression when applied on the stiffness matrix.
- Numerical solution of elliptic and parabolic problems in the field of fractional diffusion, modeled with the integral formulation of the fractional Laplacian and discretized with finite elements.

Research methodology

In this thesis we analyze the effectiveness, in terms of performance, parallel speed-up and accuracy (for approximate solutions), of tile methods for solving dense systems of linear algebraic equations. For this purpose, software packages are used, in which the studied methods are applied.

For the problem, examined in Chapter 2 we utilize the parallel program developed in [22] for the discretization and generation of the system of linear algebraic equations. For the fractional diffusion problems, examined in Chapters 3 and 4, the MatLab program developed in [2] is used for the generation of the system of linear algebraic equations. We developed Matlab programs for the calculation of the reordering schemes (Appendix A) and the lumped matrix of mass (Appendix B).

Content structure

The Introduction Chapter provides the motivation for the current work. A short description of the methods and problems is provided.

Chapter 1 has an introductory character and describes the utilized tile methods for solving dense systems of linear algebraic equations, as well as an estimation of their computational complexity. In Section 1.1 we provide a short description of the universal direct method Gaussian elimination and the LU factorization based on it. Section 1.4 examines the hierarchical methods for solving systems of linear algebraic equations, developed for solving systems with *structured* matrices (both dense and sparse). The advantages of HSS compression are also described – lower computational complexity for problems with suitable matrix *structure*.

Chapter 2 presents numerical results for laminar flow around Zhukovsky airfoils. The arising system has a *dense* matrix and is used as a benchmark for the comparative analysis of the analyzed tile algorithms.

In Chapter 3 we examine a two dimensional *anomalous* diffusion problem modeled with the *fractional* Laplace operator. Finite element method is utilized for the discretization in time.

Chapter 4 examines a parabolic in time problem for two dimensional *anomalous* diffusion.

Chapter Conclusion presents the concluding remarks summarizing the results obtained in the thesis. We also present a list of the papers and reports in scientific forums on which this work is based on.

Chapter 1 Methods for solving systems of linear algebraic equations with dense systems

Many problems in the calculation practice are solved numerically by reduction into a system of linear algebraic equations. For example, when utilizing the *boundary element method* or the *finite element method with* on a *fractional* power Laplace operator (*fractional* diffusion), the arising system has a dense coefficient matrix.

1.1 Direct methods

Gaussian elimination is an universal method for solving systems of linear algebraic equations. It is the basis of most direct methods. For example the LU factorization is based on the sequential elimination of unknowns in the Gauss method. The LU factorization is a primary method, implemented in the high performance software libraries for computational linear algebra.

1.2 Gauss method

The Gauss method for solving systems of linear algebraic equations $Ax = b$ involves two parts: (i) Forward elimination. Applies elementary row operations to transform the system into one with an Upper triangular matrix; (ii) Back substitution. Recursively in reverse order the off-diagonal elements in the i -th row of the matrix are eliminated for $i = n - 1, n - 2, \dots, 1$.

The computational complexity of the Gauss method is determined by the forward elimination [34]:

$$\mathcal{N}^{Gauss} \sim \frac{2n^3}{3} = O(n^3).$$

1.3 LU factorization

The LU factorization is expressing the matrix A as a product of two triangle matrices $A = LU$. Here L is a lower triangular matrix with ones on the main diagonal, and U is an upper triangular matrix. This factorization is calculated

with a modified Gaussian elimination and is used in high performance libraries (LAPACK, MKL, ACML, PLASMA, ATLAS, etc.) for solving systems of linear algebraic equations.

The forward elimination can be written as

$$\underbrace{L_{n-1}L_{n-2}\dots L_2L_1}_{\tilde{L}}A = U,$$

where L_1, L_2, \dots, L_{n-1} are lower triangular matrices with ones on the main diagonal. We can check that \tilde{L} and $L = \tilde{L}^{-1}$ are also lower triangular matrices with ones on the main diagonal. Thus:

$$\tilde{L}A = U \iff A = LU, \quad L = \tilde{L}^{-1}.$$

After factorization of A , the system of linear algebraic equations is reduced to solving two systems with triangular matrices. We denote

$$L \underbrace{Ux}_y = b,$$

afterwards we will:

1. Solve the system $Ly = b$ with forward substitution;
2. Solve the system $Uy = y$ with backward substitution;

The numerical complexity of the factorization is $O\left(\frac{2}{3}n^3\right)$, while the forward and backward substitutions have $O(n^2)$.

1.4 Hierarchical matrices. Methods for solving systems of linear algebraic equations with hierarchical semi-separable compression

Hierarchical semi-separable matrices is applied for the approximation of data-sparse matrices. By data-sparse we understand such matrices, that have *structure* that allows approximation by compressed matrices, which could be expressed with lower amount of elements. In the general case the data-sparse matrices do not satisfy the condition to have $O(n)$ nonzero elements. Hackbusch introduced the term Hierarchical matrices in [10] by developing the theory and algorithms for the so called \mathcal{H} -matrices.

Methods utilizing hierarchical matrices are a part of the wider group of methods for solving systems with the so called *structured* matrices. An overview of the existing methods for such methods can be found inside [3], including hierarchically

semi-separable matrices. In this work we will examine the efficiency of algorithms based on this class of methods.

STRUMPACK (STRUctured Matrices PACKage) is a parallel software library, that implements hierarchical semi-separable compression for solving dense systems of linear algebraic equations [20]. The algorithm involves three steps:

1. *Hierarchically Semi-separable compression (approximation) of the matrix of the system.* When certain conditions are satisfied this step has computational complexity of $O(r^2n)$, where r is the maximum rank of the off-diagonal blocks of the approximating matrix. It is calculated during the compression. In the general case the complexity is $O(r^2n)$.
2. *ULV-like factorization.* In this step the compressed matrix is factorized. For this step a variant of the Gauss method is applied, similar to the one described in the LU factorization above. First $O(r)$ unknowns are eliminated, then the rest of the $O(n - r)$. Computational complexity of this step is $O(r^2n)$.
3. *Solution.* In this step the compressed and factorized matrix is used along the right hand side to obtain the solution. Computational complexity is $O(rn)$.

The overall computational complexity of the method is $O(r^2n)$. As seen later this evaluation is valid for certain assumptions. In the general case the complexity is $O(rn^2)$.

1.5 Hierarchical Semi-Separable compression

In this subsection we will examine in short the hierarchical semi-separable matrices (HSS). They are first introduced by Martinsson in [15]. Algorithms implementing it are described in [20] as a part of the STRUMPACK project, for solving systems of linear algebraic equations with dense matrices. Hierarchical compression may be applied to any non-singular matrix, however it is *effective* only if the original matrix A has suitable *structure* – meaning that its off-diagonal blocks have low rank. By *effective* compression we mean one that approximates the matrix, such that the memory needed to store it is much lower and we can apply transformations on the compressed matrix with lower computational complexity.

We will denote the compress matrix of A with H . The algorithm could be written as:

1. We divide the matrix A in four blocks. We assume that the off-diagonal blocks have low rank (and Singular Value Decomposition, or another rank

calculating factorization, can be applied to them):

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} = \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix}.$$

The matrices U , B and V are called *generators*. If the off-diagonal blocks have low rank, U will be “tall and slim”, B will be small and square (or close to) and V will be “short and wide”. The relation between rows and columns will depend on the rank r . D are unchanged diagonal blocks. The “big” notation will be described below.

2. Assuming, that the diagonal blocks D also have off-diagonal low rank blocks, they are compressed in the same fashion, the process continues recursively. The second level of recursive compression can be written as:

$$A = \begin{bmatrix} \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix} & U_3^{\text{big}} B_{3,6} V_6^{\text{big}*} \\ U_6^{\text{big}} B_{6,3} V_3^{\text{big}*} & \begin{bmatrix} D_4 & U_4^{\text{big}} B_{4,5} V_5^{\text{big}*} \\ U_5^{\text{big}} B_{5,4} V_4^{\text{big}*} & D_5 \end{bmatrix} \end{bmatrix}$$

3. There is a recursive property between the *generators* of different levels of compression. This is denoted with the “big” notation. The following relations apply

$$U_3^{\text{big}} = \begin{bmatrix} U_1^{\text{big}} & 0 \\ 0 & U_2^{\text{big}} \end{bmatrix} U_3 \quad \text{and} \quad V_3^{\text{big}} = \begin{bmatrix} V_1^{\text{big}} & 0 \\ 0 & V_2^{\text{big}} \end{bmatrix} V_3 \quad (1)$$

The third level of recursive HSS compression can be written as:

$$A = \begin{bmatrix} \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix} & \begin{bmatrix} U_1^{\text{big}} & 0 \\ 0 & U_2^{\text{big}} \end{bmatrix} U_3 B_{3,6} V_6^* \begin{bmatrix} V_4^{\text{big}*} & 0 \\ 0 & V_5^{\text{big}*} \end{bmatrix} \\ \begin{bmatrix} U_4^{\text{big}} & 0 \\ 0 & U_5^{\text{big}} \end{bmatrix} U_6 B_{6,3} V_3^* \begin{bmatrix} V_1^{\text{big}*} & 0 \\ 0 & V_2^{\text{big}*} \end{bmatrix} & \begin{bmatrix} D_4 & U_4^{\text{big}} B_{4,5} V_5^{\text{big}*} \\ U_5^{\text{big}} B_{5,4} V_4^{\text{big}*} & D_5 \end{bmatrix} \end{bmatrix} \quad (2)$$

Generators with the notation “big” can be computed outside of the highest levels of recursive compression. U can be computed from the U_τ and U^{big} at the higher level of compression. At the last level $U = U^{\text{big}}$.

In the general case equation 2 isn’t exact, but approximate. This means that as a result we obtain an approximation of A , that we will denote as $H \approx A$.

To achieve this a suitable threshold ε must be supplied. It is necessary for the calculation of the *generators*. When a larger threshold is used, the *generators* are

smaller and the compressed matrix takes less memory and allows more effective arithmetic operations with it, but this is at the price of lower accuracy. If a smaller threshold is chosen it is exactly the opposite.

As we will show in the later chapters, the ordering of the unknowns while assembling the matrix A is pivotal for the effectiveness of the HSS compression. If A is scrambled randomly, that will almost assuredly destroy any suitable for this method *structure*.

For certain classes of problems it is possible to reorder the unknowns in such a way, that the arising *structure* of the matrix is improved. For example in [18], several methods for clusterization are studied for Kernel Ridge Regression. In Chapter 3 we will propose and analyze several methods for reordering of the unknowns for a system of linear algebraic equations, arising from the discretization of an elliptic problem with a *fractional* power of the Laplace operator (fractional diffusion problem).

1.6 Compression with randomized sampling

The HSS algorithm deployed in STRUMPACK is based on applying randomized sampling, which utilizes multiplication of random vectors with the original matrix A . This method is first proposed by Martinsson in [15]. This algorithm doesn't need the explicit form of A . Instead a function that accesses (or calculates on demand) and another one that calculates a product of A with a vector is needed. The advantages of this approach, as well as an adaptive random sampling algorithm are studied by Gorman et al. in [9]. The Randomized sampling is also useful when integrating HSS kernels in sparse solvers [8].

In the general case the computational complexity of a matrix-vector product is $O(n^2)$. This leads to $O(rn^2)$ for the HSS compression algorithm. For certain classes of problems r is much smaller than n . For example for 2D Poisson problems (FEM) r is a constant, while for 3D Helmholtz (BEM) it rises slowly with n . If a fast algorithm for multiplying the compressed matrix with a vector is supplied, the complexity of the compression can be lowered to $O(r^2n)$.

1.7 ULV-like factorization and solution

The compressed matrix H in HSS form can be factorized with a special form of LU factorization known as ULV factorization [5]. This factorization uses orthogonal transformations to first eliminate $n - r$. The rest r unknowns are then eliminated with LU factorization. The factorization employed inside STRUMPACK is realized as ULV-like factorization, that instead uses orthogonal transformations utilizes the HSS structure of the compressed matrix H .

The process is visualized in Figure 1.

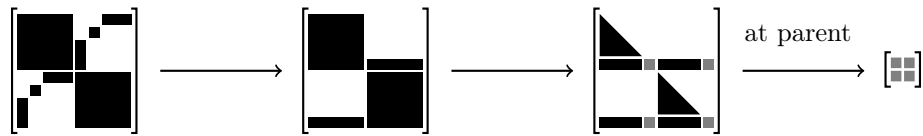


Figure 1: ULV-like factorization.

After applying the ULV-like factorization, the system of linear algebraic equations $Ax = b$ is reduced to solving two systems with triangular matrices. The computational complexity of this step is $O(rn)$ [20].

Chapter 2 Boundary Element Method for numerical solution of a two dimensional flow around airfoils problem

This chapter studies a numerical method for a computational simulation of laminar flow around Zhukovsky airfoils. This work employs the method described in [17]. We have developed an implementation for a cascade of airfoils in ideal fluid. The method is based on spline collocation with interpolation in parts. A parallel C program is developed in [22] for this problem.

After discretization of the integral equation with the boundary element method the problem is reduced to a dense system of linear algebraic equations. The results from the studied in the thesis hierarchical method are compared with results obtained with Gaussian elimination, implemented in several popular software packages. On the CPUs we compare the performance of Intel Math Kernel Library (MKL) and Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA), while on the Intel Xeon Phi coprocessors (abbreviated as MICs from the Many Integrated Core architecture) the performance of MKL is compared with Matrix Algebra on GPU and Multicore Architectures (MAGMA) for MIC architecture (abbreviated as MAGMA MIC).

2.1 Problem statement

2.2 Boundary Element Method calculating the flow function in an ideal fluid in unbounded two dimensional domain

Let $\Omega \subset \mathbb{R}^2$ be an unbounded multi-connected domain with a smooth enough internal border S . The flow function Ψ satisfies the Laplace equation:

$$\nabla^2 \Psi \equiv \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = 0 \tag{3}$$

in $\Omega \subset \mathbb{R}^2$ and can be written as

$$\Psi(P) = -\frac{1}{4\pi} \int_S \gamma(\sigma) \ln(r^2(P, Q)) d\sigma_Q + \Psi_\infty(P) + C_0, \quad P \in \Omega. \tag{4}$$

where $r^2(P, Q) = (x - \xi)^2 + (y - \eta)^2$, $P = (x, y)$, $Q = (\xi, \eta)$ and $d\sigma_Q$ is a measure on S . The first term on the right hand side represents a simple layer (stream function of the vortex layer), where $\gamma(\sigma)$ is the density of the layer, Ψ_∞ is a harmonic function added to the potential of the layer in order to satisfy the condition at infinity for external boundary problems. The velocity field $\vec{C} = (u, v)$ is defined by

$$u = \frac{\partial \Psi}{\partial y}, \quad v = \frac{\partial \Psi}{\partial x}$$

satisfying the equations

$$u = \frac{1}{2\pi} \int_S \gamma(\sigma) \frac{y - \eta}{r^2} d\sigma, \quad v = -\frac{1}{2\pi} \int_S \gamma(\sigma) \frac{x - \xi}{r^2} d\sigma.$$

2.3 Fluid flow around airfoils

In this section we will examine the problem of fluid flow around Zhukovsky airfoils. We assume that the flow at infinity has homogeneous velocity $\vec{C}_\infty = (1, 0)$. Here S denotes the contours of the airfoils. For the examined problem the flow function Ψ satisfies the Laplace equations (3). The airfoils S are impermeable, therefore the following boundary condition is in force $\Psi|_S = K = \text{const}$. In order to satisfy the conditions at \vec{C}_∞ , we should choose Ψ_∞ such that

$$\vec{C}_\infty = \left(\frac{\partial \Psi_\infty}{\partial y}, -\frac{\partial \Psi_\infty}{\partial x} \right).$$

Here, we have

$$\Psi_\infty(P) = \gamma_\infty(P).$$

The boundary condition takes the form

$$\gamma(P) - \frac{1}{4\pi} \int_S \gamma(\sigma) \ln(r^2(P, Q)) d\sigma_Q + C = 0. \quad (5)$$

Finally, the Kutta-Joukowski's condition $\gamma(A) = 0$, is used to obtain a unique solution of the boundary value problem, where A are the points on the sharp tip of the airfoils.

2.4 Discretization

For the numerical solution of the integral equation (5) we apply the boundary element method. The problem is reduced to a system of linear algebraic equation of the form:

$$(A\gamma) s = f(s).$$

We look for a numerical solution

$$\gamma_h(S) = \sum_{i=1}^n \gamma_i \phi_i(s),$$

where $\{\phi_i(s)\}_{i=1}^n$ is the Lagrangian basis, corresponding to the discretization mesh S_h on the airfoils and $\gamma_i = \gamma_h(s_i)$, $i = 1, \dots, n$ are the BEM nodal unknowns. The collocation method is applied to the mid points of the boundary elements from S_h . Following [22] we obtain the system of linear algebraic equations

$$\sum_{i=1}^n \gamma_i \Psi_{ji} = f_j, \quad j = 1, 2, \dots, n, \quad (6)$$

където $\Psi_{ji} = \Psi_i(s_j)$, $f(s_j) = f_j$, $\Psi_i(s) = (A\phi_i)(s)$.

2.5 Analysis of numerical experiments on computers with shared memory

Computing the matrix D has computational complexity $O(n^2)$. The calculation of drag and lift forces have complexity $O(n)$. The focus of this work is on the most computationally complex part – the solving of the system of linear algebraic equations.

2.6 LU factorization

2.6.1 CPU processors with shared memory

PLASMA (Parallel Linear Algebra Software for Multicore Architectures) [7] is a software library for solving systems of linear algebraic equations implementing the standard LAPACK library. The effectiveness of PLASMA is based on using highly optimized BLAS library (Basic Linear Algebra Subprograms) in which the basic linear algebra operations are implemented – vector, matrix-vector and matrix-matrix multiplications. For this level of computation we employ the libraries MKL BLAS and ATLAS ATLAS (Automatically Tuned Linear Algebra Software) BLAS [32].

In Table 1 we present the results of the numerical experiments for solving systems of linear algebraic equations obtained when applying the boundary element method for discretization of a flow around airfoils problem. We compare the sequential and parallel execution times for PLASMA + ATLAS, PLASMA + MKL and MKL solvers for $n = 5\,000$ and $n = 40\,000$ and varying the number of threads used.

Table 1: Sequential and parallel execution times on CPU processors with shared memory

Library		Plasma + ATLAS		PLASMA + MKL		MKL	
Threads	n	time [s]	speed-up	time [s]	speed-up	time [s]	speed-up
1	5 000	8.42	1.00	5.03	1.00	5.30	1.00
16	5 000	0.67	12.57	0.47	10.69	0.47	11.26
32	5 000	0.88	9.59	0.65	7.76	0.65	8.12
1	40 000	4008.76	1.00	2497.12	1.00	2233.93	1.00
16	40 000	282.94	14.17	166.41	15.01	147.64	15.13
32	40 000	325.17	12.33	169.58	14.73	148.59	15.03

The results show good parallel speed-up for all tested libraries up to 16 threads. The speed up achieved comes close to 15, which is close to the theoretical maximum of 16.

The parallel effectiveness of PLASMA with MKL and MKL is quite similar, increasing up to 94% for the larger problem. They outperform more than 1.5 times PLASMA with ATLAS. One possible reason for this could be that the `gcc` compiler doesn't employ vectoring as well as `icc`.

2.6.2 MIC coprocessors

In this subsection we analyze the parallel effectiveness of the Intel Xeon Phi 7120P accelerators (MICs). The MICs are designed for massive parallelism and vectorization as required in High Performance computing. Every MIC has 61 cores and each core can run 4 threads simultaneously for a total of 244 threads. We use the

Table 2: Sequential and parallel times and speed up for solving systems of linear algebraic equations on MIC coprocessors

Library		MAGMA MIC		MKL	
Threads	n	time [s]	speed-up	time [s]	speed-up
1	5 000	11.70	1.00	17.64	1.00
60	5 000	5.81	2.01	2.86	6.16
120	5 000	6.76	1.73	3.55	4.97
240	5 000	5.39	2.17	3.80	4.64
1	40 000	4896.49	1.00	2101.93	1.00
60	40 000	665.53	7.36	154.23	13.63
120	40 000	432.89	11.31	93.80	22.41
240	40 000	208.48	23.49	64.43	32.62

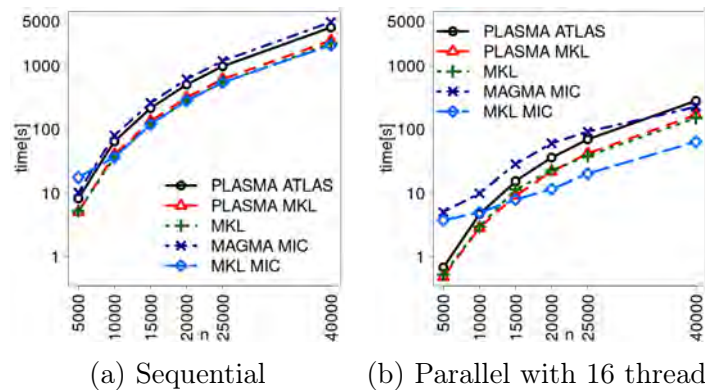


Figure 2: Comparison of the performance of LU solvers for CPU and MIC

Offload mode which reserves one of the cores for communication with the CPU, thus we can use up to 60 cores (240 threads).

In Table 2 we present the results from numerical experiments with sizes $n = 5\,000$ and $n = 40\,000$, varying the number of threads from 1 to 240. The results show very good performance of MKL in comparison with MAGMA MIC. This may be because of better communication between the threads in MKL. For the biggest problem we achieve parallel speed-up of 32.

On Figure 2 we compare the performance of the studied software libraries for the CPU and MIC architectures. The performance of PLASMA with MKL is better than that of MAGMA MIC. That may be due to better communication.

In conclusion we note that the MKL package has better performance both in the CPUs and the MIC coprocessor. The best time on the coprocessor is 4 times faster than the best on the CPU.

2.7 Hierarchical Semi-Separable compression

The Hierarchically Semi-Separable compression, implemented in the software library STRUMPACK is approximate. This means that the compressed matrix H approximates the original matrix A . The user must provide two thresholds – absolute ε_{abs} and relative ε_{rel} [9]. In the results presented on this thesis we fix the absolute threshold at $\varepsilon_{\text{abs}} = 10^{-8}$ and vary the relative one $\varepsilon_{\text{rel}} = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ and 10^{-12} .

2.7.1 Comparative analysis of hierarchical and LU factorization on CPUs with shared memory

In this subsection we analyze the performance of the HSS compression method and its software implementation in STRUMPACK in comparison with Gaussian elimination and its best (based on the analysis above) implementation MKL, where we use LU factorization. The sequential Figure 3a and parallel times Figure 3b are presented. The results affirm the better computational complexity of the HSS compression $O(rn^2)$ in comparison to the LU factorization $O(n^3)$. The impact of the relative threshold ε_{rel} can be seen clearly.

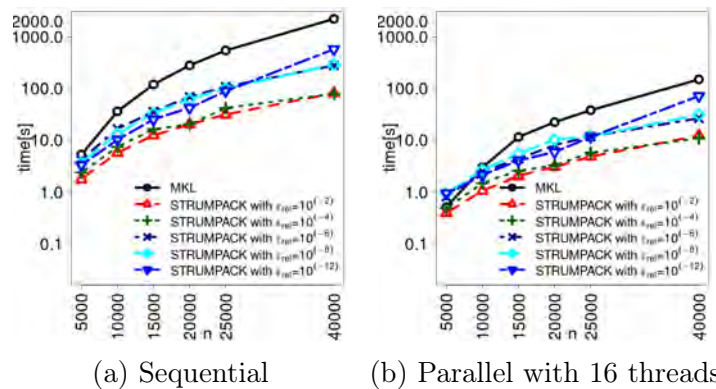


Figure 3: Performance of STRUMPACK in comparison with MKL

For the sequential experiments STRUMPACK is more efficient than the best direct solver employed – MKL. STRUMPACK shows lower parallel speed up - around ~ 2 to ~ 5 . This is due to the more complex recursive structure of the HSS compression.

2.7.2 Error analysis for the HSS-based solver

Let us remind that the HSS compression is approximate, i.e. the compressed matrix H is an approximation of A . The solution of the system of linear algebraic

equations, obtained with HSS compression, is thus an approximation of the exact solution. In order to estimate the error of the method we use the solution obtained with the LU solver as a reference (see Subsection 2.6). Here we analyze the relative error R_{relative} with the following definition:

$$R_{\text{relative}} = \frac{\|x^{\text{Gauss}} - x^{\text{HSS}}\|_{l_2}}{\|x^{\text{Gauss}}\|_{l_2}} = \frac{\sqrt{\sum_{i=1}^n (x_i^{\text{Gauss}} - x_i^{\text{HSS}})^2}}{\sqrt{\sum_{i=1}^n (x_i^{\text{Gauss}})^2}}, \quad (7)$$

where x^{Gauss} is the reference solution obtained with the MKL solver, while x^{HSS} is the solution obtained with HSS compression.

In Table 3 we show the relative errors R_{relative} , varying the size of the problem $n \in \{5\,005, 10\,005, 15\,005, 20\,005, 25\,005, 40\,005\}$, as well as the relative thresholds $\varepsilon_{\text{rel}} \in \{10^{-6}, 10^{-8}, 10^{-12}\}$.

Table 3: Relative accuracy R_{relative}

n	ε_{rel}			n	ε_{rel}		
	10^{-6}	10^{-8}	10^{-12}		10^{-6}	10^{-8}	10^{-12}
5005	1.1	0.085	0.00019	15005	0.29	0.23	0.00097
20005	0.28	0.34	0.0038	25005	0.3	1.48	0.013
10005	0.75	0.17	0.00075	40005	0.37	1.59	0.027

The accuracy and computational effectiveness of the hierarchical method rely on the max rank of the off-diagonal blocks r , which is in turn dependent on the chosen error thresholds and the *structure* of the original matrix A . Higher rank r will result in smaller relative error R_{relative} as well as longer solution time and vice versa.

The analysis of the presented results shows that in order to achieve high accuracy for the method we require very small threshold corresponding to higher rank r . In these case HSS compression may not be sufficiently effective.

2.8 Parallel scalability on computer systems with distributed memory

This section is concentrated on some specifics and hardships when working with HPC systems with hybrid architecture. Those machines have distributed memory on the server level and shared memory on each server. As in the previous section the comparative analysis includes the parallel libraries MKL and STRUMPACK. In order to solve the system of linear algebraic equations, arising from the BEM discretization of flow around Zhukovsky airfoils, we employ one or two servers connected with Ethernet. We analyze the following variants with MKL and

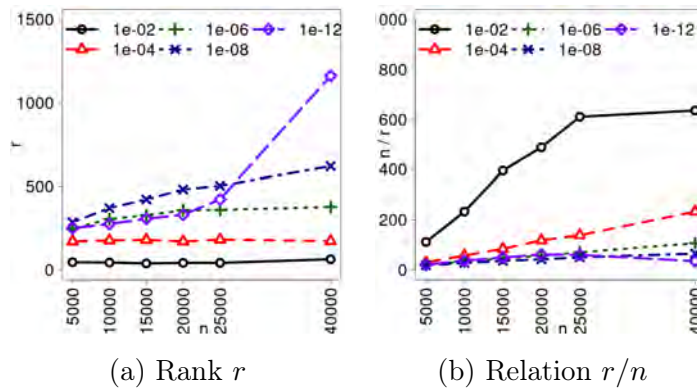


Figure 4: Maximum off-diagonal rank r

STRUMPACK: Sequential; Parallel with 24 OpenMP threads; Parallel with 24 MPI processes on 1 server; Parallel with 48 MPI processes on 2 servers; Hybrid parallelization with 2 MPI processes on 2 servers, each with 24 OpenMP threads.

2.9 LU factorization

On Figure 5 we show the execution times for solving the system of linear algebraic equations with MKL. The best times are obtained when using a single server with OpenMP, followed by MPI Figure 5b. This is explained by the slower Ethernet communications when employing more than a single server.

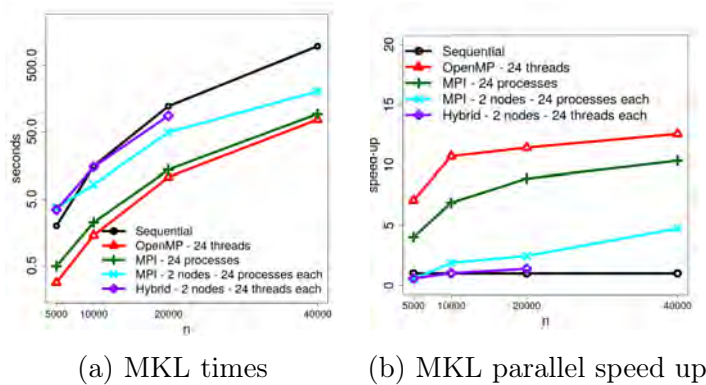


Figure 5: Parallel times and speed up with MKL.

2.10 HSS compression

When solving the system with HSS compression, the obtained parallel speed up is lower than when using the direct Gaussian solver Figure 6. This can be explained

with the recursive structure of the HSS compression.

In the numerical experiments with the lowest relative threshold ε_{rel} we obtain the best computational time when using MPI on a single server. This could likely be explained by the fact that OpenMP parallelization is implemented later than the MPI optimization.

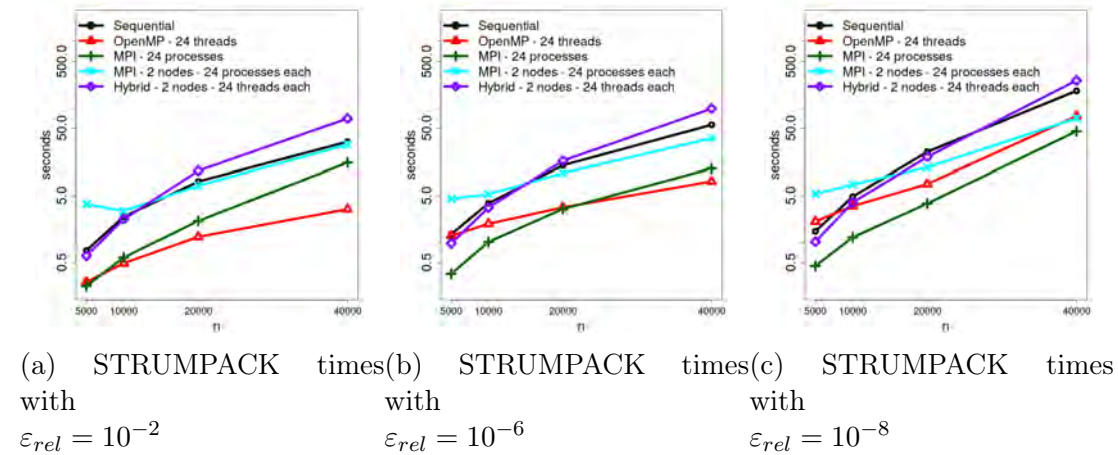


Figure 6: Times and parallel speed up for STRUMPACK.

The behavior of the parallel speed up is drastically changed when employing two servers. This is due to the much slower (relatively) connection between them – 1000 Mb Ethernet. We can conclude that effectiveness could significantly be improved with faster communication medium, like InfinBand, as well as increasing the size of the solved systems of linear algebraic equations.

2.11 Concluding remarks

Central part in the presented results is given to the examination of the Hierarchically Semi-Separable compression method. The experimental comparative analysis is based on the implementation in the STRUMPACK package. It shows much better performance than the Gaussian solvers, utilizing LU factorization. In the same time the parallel speed up of STRUMPACK is found lacking. This is explained with the more complex hierarchical structure of the algorithm.

The accuracy and computational effectiveness of the HSS compression depends on the choice of the absolute and relative thresholds. These parameters must be determined by the user. The presented analysis shows how to achieve best performance with the given accuracy.

Chapter 3 Finite Element Method for numerical solution of a two dimensional fractional diffusion problem

The fractional elliptic in space problems of power $\alpha \in (0, 1)$ are utilized in modeling *anomalous* diffusion problems. The arising boundary problems or non-local in the general case the numerical solution of such problems is a extremely computationally expensive process. Such models are used in image processing, financial mathematics, electromagnetostatics, peridynamics, modeling flow in porous media and many others.

The presented numerical experiments are for model problems in a square and circle domain, and this abstract will show results only for the former. The fractional Laplacian is defined through the Riesz potential. The theoretical basis and the specialized finite element method for its numerical solution can be found in [1] by Acosta et. al. The algorithmic implementation of the method can be found in [2] from the same authors

The performance of several software packages implementing Gaussian elimination was analyzed in Chapter 2. Here we will use only the most effective of them: Intel's math Kernel Library (MKL). In this chapter we will analyze the performance of the HSS compression based algorithm, implemented in the STRUMPACK library. We will employ several reordering schemes in order to improve the *structure* of the stiffness matrix.

3.1 Problem Statement

The *fractional* Laplacian can be defined as

$$(-\Delta)^\alpha u(x) = C(d, \alpha) \text{P.V.} \int_{\mathbb{R}^n} \frac{u(x) - u(y)}{|x - y|^{d+2\alpha}}, \quad (8)$$

where P.V. means principal value, d is the number of dimensions, $\alpha \in (0, 1)$. The normalized constant $C(d, \alpha)$ can be written as

$$C(d, \alpha) = \frac{2^{2\alpha} \alpha \Gamma\left(\alpha + \frac{d}{2}\right)}{\pi^{d/2} \Gamma(1 - \alpha)},$$

where Γ is the gamma function.

In this chapter we utilize the integral definition of the fractional Laplace operator according to the definitions in [2] by Acosta et. al. The examined boundary

value problem can be defined as

$$\begin{cases} (-\Delta)^\alpha u(x) = f(x), & x \in \Omega \\ u(x) = 0, & x \in \Omega^c. \end{cases} \quad (9)$$

Here $\Omega \subset \mathbb{R}^d$ is a bounded domain, Ω^c is the complement of Ω in \mathbb{R}^d and $f(x)$, $x \in \Omega$ is the right hand side with enough smoothness.

The variational formulation is obtained from (9) by multiplying with a test function and integrating by parts. The equation for the weak solution is: find $u \in H^\alpha(\Omega)$, such that

$$\frac{C(d, \alpha)}{2} \langle u, v \rangle_{H^\alpha \mathbb{R}^d} = \int_{\Omega} f v, \quad v \in \tilde{H}^\alpha(\Omega). \quad (10)$$

The scalar product of u and v can be defined in the Hilbert space $H^\alpha(\Omega)$ with norm $\|\cdot\|_{H^\alpha(\Omega)} = \|\cdot\|_{L^2(\Omega)} + |\cdot|_{H^\alpha(\Omega)}$. Here $|\cdot|_{H^\alpha(\Omega)}$ is the Aronszajn-Slobodeckij half norm. $\langle u, v \rangle_{H^\alpha \mathbb{R}^d}$ can be written as

$$\langle u, v \rangle_{H^\alpha \mathbb{R}^d} = \iint_{\mathbb{R}^d \times \mathbb{R}^d} \frac{(u(x) - u(y))(v(x) - v(y))}{|x - y|^{d+2\alpha}} dx dy.$$

We will point out that integration is carried out over the whole space \mathbb{R}^d

Correctness of the variational definition of the problem (10) as well as the existence and uniqueness of the solution $\tilde{H}^\alpha(\Omega)$ follow from the Lax-Milgram lemma.

3.2 Finite element settings

Let \mathcal{T} be an admissible triangulation of the domain Ω containing $N_{\mathcal{T}}$ triangular elements. We examine the finite element space \mathbb{V}_h of continuous linear in part functions over \mathcal{T} . Let $\{\varphi_1, \dots, \varphi_N\} \subset \mathbb{V}_h$ be the Lagrangian nodal basis, corresponding to the internal nodes x_1, \dots, x_N . Then $\varphi_i(x_j) = \delta_i^j$. Let $T \in \mathcal{T}$ is an element on the triangulation and let's denote with h_T and ρ_T the diameter and the inner radius. We also write $h = \max_{T \in \mathcal{T}} h_T$. We examine shape-regular triangulations, such that $\tau > 0$ independent of \mathcal{T} such that

$$h_T \leq \tau \rho_T, \quad \forall T \in \mathcal{T}.$$

Under these conditions for any $\alpha \in (0, 1)$ the discrete analog of the variational problem (10) can be written as

$$\frac{C(d, \alpha)}{2} \langle u_h, v_h \rangle_{H^\alpha(\mathbb{R}^d)} = \int_{\Omega} f v_h, \quad v_h \in \mathbb{V}_h. \quad (11)$$

Here $u_h = \sum_j u_j \varphi_j$ denotes the numerical FEM solution of the stationary diffusion problem (9).

Solving the discrete variational problem (11) is reduced to solving the system of linear algebraic equations $KU = F$, where $U = (u_j) \in \mathbb{R}^N$ are the nodal unknowns. The stiffness matrix K is symmetrical and positively defined, therefore the system has a single solution.

In the proposed in [1] FEM variant integration is reduced to a ball domain $B \supset \Omega$, such that the distance between the border $\bar{\Omega}$ and the compliment B^c is sufficient. This introduces the additional triangulation $\tilde{\mathcal{T}}_A$ over $V \setminus \Omega$, such that the summary triangulation $\tilde{\mathcal{T}} = \mathcal{T} \cup \mathcal{T}_A$ over B is admissible. An example of such triangulation is shown on Figure 7. The nodes denoted with **bold** font correspond to the unknowns u_j .

Let's denote the elements in the triangulation over B with $M_{\tilde{\mathcal{T}}}$. Then the elements inside the stiffness matrix K may be written as

$$K_{ij} = \frac{C(d, \alpha)}{2} \sum_{\ell=1}^{N_{\tilde{\mathcal{T}}}} \left(\sum_{m=1}^{N_{\tilde{\mathcal{T}}}} I_{\ell,m}^{i,j} + 2J_{\ell}^{i,j} \right), \quad \ell, m \in [1, \dots, N_{\tilde{\mathcal{T}}}], \quad (12)$$

where for the integrals I and J the following equations hold true

$$I_{\ell,m}^{i,j} = \int_{T_{\ell}} \int_{T_m} \frac{(\varphi(x) - \varphi_i(y)) (\varphi_j(x) - \varphi_j(y))}{|x - y|^{2+2\alpha}} dx dy \quad (13)$$

$$J_{\ell}^{i,j} = \int_{T_{\ell}} \int_{B^c} \frac{\varphi_i(x) \varphi_j(x)}{|x - y|^{2+2\alpha}} dy dx. \quad (14)$$

3.3 Reordering of the unknowns

In Section 1.5 we have mentioned that for certain classes of problem it is possible to significantly improve the *structure* of the matrix by reordering the unknowns. The analysis carried out in this subsection shows that for the *fractional* diffusion problem this is necessary.

The original ordering of the triangulation nodes is shown on Figure 9a. This ordering is obtained from the MatLab function `initmesh` when generating the finite element mesh. The *structure* of the matrix corresponding to this ordering is not suitable for the HSS solver from STRUMPACK.

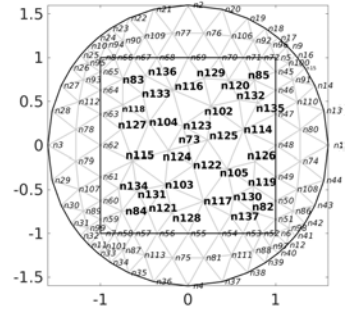


Figure 7: Admissible triangulation Ω with an auxiliary ball B .

3.4 Reordering by “Y” coordinate – “top”

With this reordering scheme the nodes in Ω are reordered by their “Y” coordinate. The obtained reordering and matrix structure can be seen on Figure 9b.

3.4.1 Reordering by lines – „stripes“

With this reordering scheme the nodes are reordered on horizontal lines. An example of this reordering and matrix structure can be see on Figure 9c.

3.4.2 Reordering on a spiral – „snake“

This algorithm reorders the unknowns on a spiral, similar to a coiled snake. The reordering and the structure of the arising matrix can be seen on Figure 9d

3.4.3 Reordering with Nested Dissection

The Nested Dissection method applies a “divide and conquer” heuristic for the division of a graph representing a sparse symmetrical matrix. The algorithm is executed in three steps:

1. Construction of an undirected graph corresponding to the triangulation, such that the vertices are the mesh nodes and the edges are the sides of the corresponding triangles.
2. Recursive partitioning of the graphs with separators (small sets of vertices which, when removed, divide the graph in subgraphs).
3. Reordering of the nodes in accordance with the recursive structure: First on subgraphs and then by separators.

The separators for an example square domain can be seen on Figure 8. The reordering and the matrix structure are visualized on Figure9d.

3.4.4 Reordering with recursive bisection

Recursive Bisection [21] is examined as another method for division of graphs. Unlike the Nested Dissection the graph is divided in two subgraphs by removing edges. This process continues recursively. The subgraphs in each recursive partitioning are numbered sequentially.

Recursive Bisection is used to balance the load on distributed memory HPC machines [21].

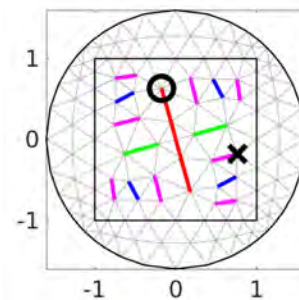


Figure 8: Reordering with nested dissection. “X” marks the first node, “O” marks the last.

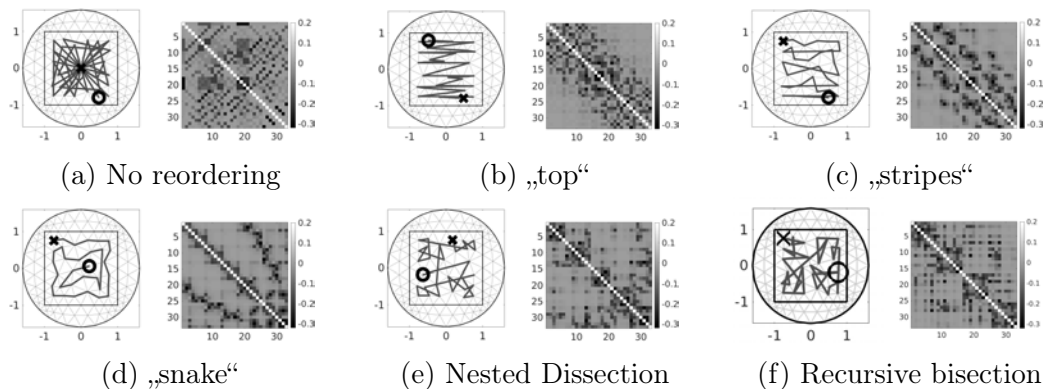


Figure 9: Reordering of the nodes in a square domain Ω and the structure of the corresponding matrix. Dark gray lines represent the ordering of the nodes from the first (marked with “X”) and the last (marked with “O”).

3.5 Analysis of numerical experiments on computational systems with shared memory

The experimental results presented in this section are obtained on a single server of the AVITOHOL supercomputer. The original ordering is obtained from the mesh generation implemented in the program from [2]. In order to improve the effectiveness of the hierarchical solver from STRUMPACK we analyze several reordering schemes.

The numerical experiments are for a *fractional* Laplacian with power $\alpha = 0.5$. Visualization of the numerical solutions for the problem in a square and circle domain are shown on Figure 10. In the abstract we show results only for the square domain.

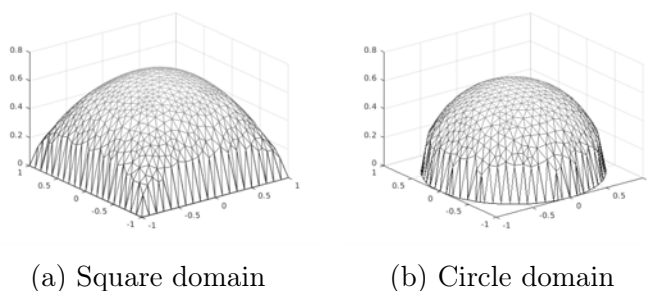


Figure 10: Solution of a *fractional* diffusion problem in square and circle domains.

In this section we analyze the performance of the two examined solver, based on their software implementation: LU factorization from MKL and HSS compression

from STRUMPACK.

3.6 Square domain

On Figure 11 we present the sequential execution times for solving the system of linear algebraic equations. In most experiments the hierarchical solver shows better times with recursive bisection, followed by the nested dissection, “top” and “stripes”. Experiments show that MKL has better performance for all values of the relative threshold except for $\varepsilon_{rel} = 10^{-2}$, where the recursive bisection, “stripes” and “top” reorderings show better results.

We can also conclude that, except for the results without reordering and with the “snake” algorithm, STRUMPACK shows similar performance to MKL.

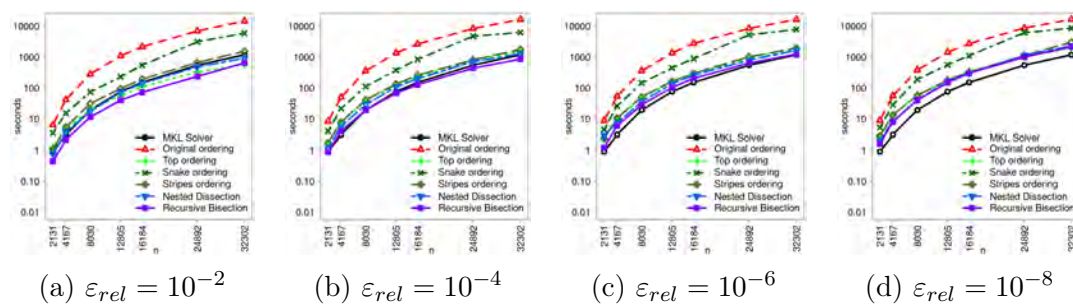


Figure 11: Comparative analysis of the execution times for solving the system of linear algebraic equations with MKL and STRUMPACK with reorderings “top”, “snake”, “stripes”, nested dissection and recursive bisection.

3.6.1 Off-diagonal rank

On Figure 12 we present the off diagonal rank r , calculated in the HSS compression step. On Figure 13 we show the relation n/r . The value of r is a measure for the effectiveness of the HSS compression on a given matrix. The smaller the rank is, the more effective the compression is. With the original ordering r has higher ranks that vary between $1/3$ and $1/4$ of the number of unknowns. The “snake” reordering shows the next highest ranks. The rest of the reorderings have similar values for the ranks with the recursive bisection having the edge on most of the experiments. The role of the maximum off-diagonal rank is visible from the results analyzed in the previous section, shown on Figure 11.

3.6.2 Parallel times and speed-up

In this subsection we analyze the parallel speed up. The numerical experiments are carried out with increasing problem size $n \in [2\ 131, 32\ 302]$. The parallel

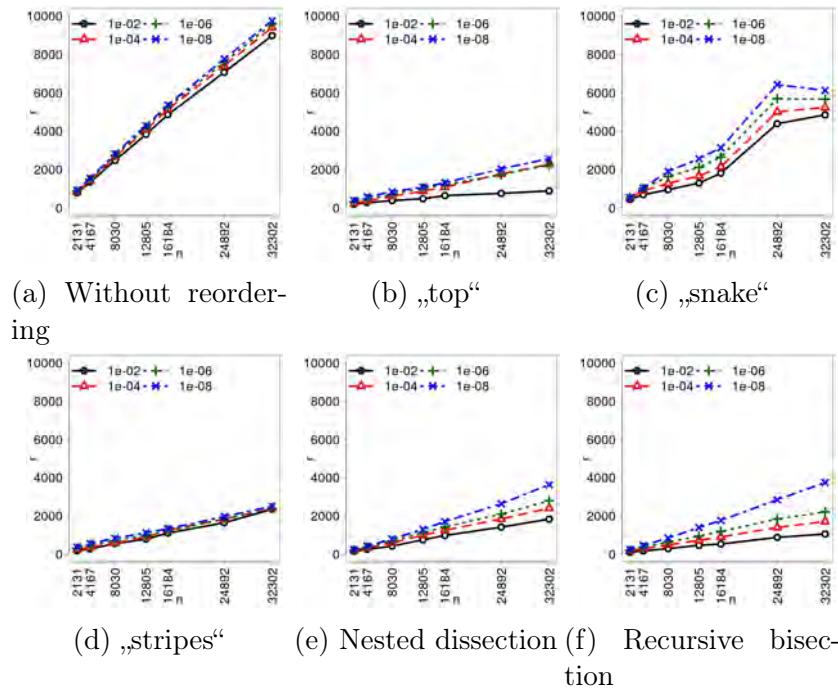


Figure 12: Maximum off-diagonal rank r .

results obtained with 16 threads are presented in Figure 14. The graphs have similar behavior and for the larger values of n the acceleration becomes linear. This is according to the theoretical assessments. We will also note that for $\varepsilon_{\text{rel}} = 10^{-2}$ and “top” reordering HSS compression has better times for the two largest sizes $n = 24\,892$ and $32\,302$. In the rest of the case the MKL solver is faster. This is due to the more complex recursive structure of the HSS compression. For larger problems we can expect STRUMPACK to show better times than MKL when using smaller thresholds, too.

3.6.3 Analysis of the accuracy of the HSS based solver

When applying the HSS solver from STRUMPACK we obtain an approximation of the solution. This is because the compressed matrix H is an approximation of the original stiffness matrix K . As with the previous chapter we will analyze the error of the method by calculating the relative error R_{relative} (7). Again we will utilize the direct Gaussian solution as reference.

The relative errors for the original ordering and the analyzed reorderings is presented in Table 4. For most of the experiments the relative error is similar to the relative threshold ε_{rel} . There are a few exceptions, where the accuracy is lower than expected. For example when using the “stripes” and “top” reorderings for the

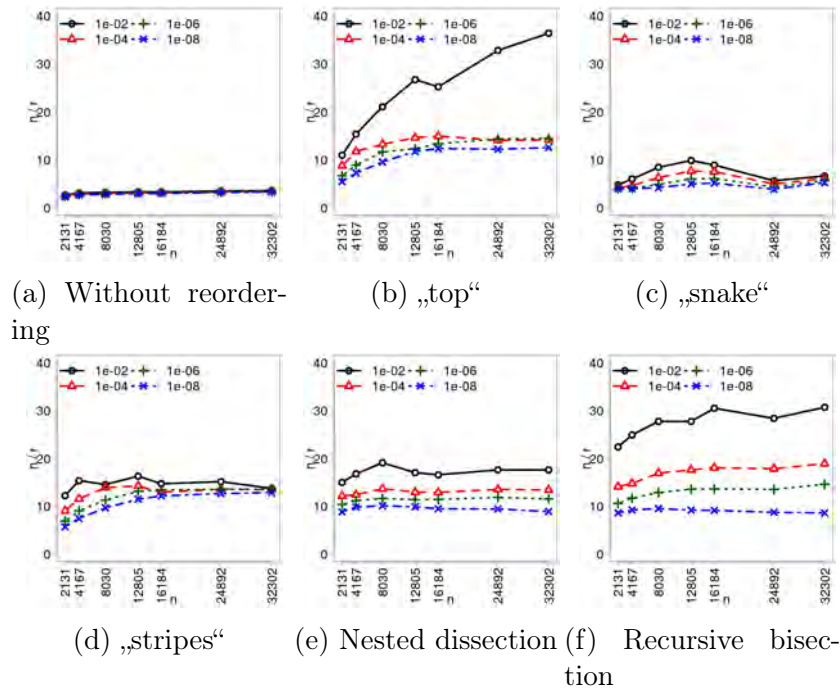


Figure 13: Relation of the number of unknowns and maximum off-diagonal rank r .

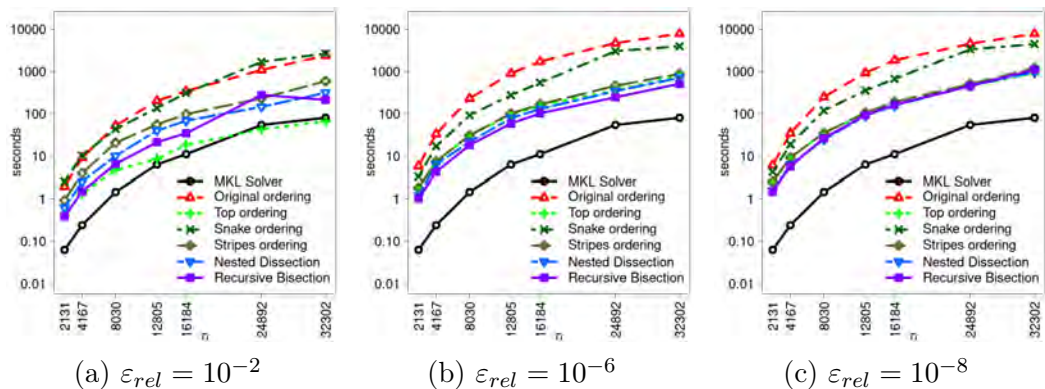


Figure 14: Comparison of the parallel times with MKL and STRUMPACK with the studied reorderings

the largest problems ($n = 32\ 302$, the relative error is substantially larger than ϵ_{rel} . This is another indicator of the need for suitable reordering methods.

The relative error depends on the effectiveness of the compression. This means that when the maximum off-diagonal rank r is small, the compression is more effective. This leads to smaller computational times, but larger relative error

Table 4: Relative error for square domain.

n	No reorder				„top“			
	10^{-2}	$R_{relative}$ 10^{-4}	$\text{3a } r_{tol}$ 10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ 10^{-4}	$\text{3a } r_{tol}$ 10^{-6}	10^{-8}
2131	0.0125	0.000124	1.633e-06	1.088e-08	0.122	0.00014	1.469e-06	1.003e-08
4167	0.0244	0.000211	2.195e-06	1.414e-08	0.194	0.000345	1.868e-06	1.95e-08
8030	0.0435	0.000422	7.515e-06	5.049e-08	0.237	0.00681	5.732e-06	3.446e-08
12805	6.74	0.0212	6.258e-06	5.04e-08	0.329	0.00976	4.3e-06	6.554e-08
16184	0.136	0.000818	1.307e-05	1.035e-07	0.335	0.00117	5.96e-06	6.16e-08
24892	0.115	0.000955	1.791e-05	1.659e-07	0.45	0.00192	4.920e-06	9.5e-08
32302	0.145	0.0011	3.409e-05	1.533e-07	0.479	0.00246	9.788e-06	9.09e-08
			„snake“				„stripes“	
n	10^{-2}	$R_{relative}$ 10^{-4}	$\text{3a } r_{tol}$ 10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ 10^{-4}	$\text{3a } r_{tol}$ 10^{-6}	10^{-8}
2131	0.0786	0.000229	1.353e-06	1.144e-08	0.111	0.000324	2.01e-06	1.283e-08
4167	0.172	0.000355	2.212e-06	2.357e-08	0.193	0.000623	2.178e-06	4.491e-08
8030	0.206	0.00107	2.737e-06	5.342e-08	0.287	0.00108	7.943e-06	4.421e-08
12805	0.324	0.00286	6.105e-06	1.995e-07	0.382	0.00117	6.793e-06	7.977e-08
16184	0.397	0.00363	8.919e-06	9.394e-08	0.393	0.00163	5.446e-06	7.988e-08
24892	0.513	0.00753	1.32e-05	1.174e-07	0.478	0.00176	7.613e-06	1.211e-07
32302	0.587	0.00647	3.129e-05	1.774e-07	0.497	0.00231	9.161e-06	1.088e-07
			Nested dissection				Recursive bisection	
n	$R_{relative}$ 10^{-2}	for STRUMPACK 10^{-4}	with r_{tol} 10^{-6}	10^{-8}	$R_{relative}$ 10^{-2}	for STRUMPACK 10^{-4}	with r_{tol} 10^{-6}	10^{-8}
2131	0.145	0.000403	2.837e-06	3.022e-08	0.0892	0.000338	2.694e-06	2.373e-08
4167	0.247	0.00138	2.866e-06	6.963e-08	0.223	0.000987	3.28e-06	5.245e-08
8030	0.378	0.00221	6.636e-06	7.277e-08	0.373	0.00172	6.402e-06	7.322e-08
12805	0.469	0.00342	8.211e-06	1.159e-07	0.424	0.000929	7.06e-05	1.043e-07
16184	0.499	0.00268	8.668e-06	1.542e-07	0.487	0.00217	8.736e-06	1.745e-07
24892	0.583	0.00318	1.04e-05	1.536e-07	0.536	0.0033	1.738e-05	2.e-07
32302	0.615	0.00539	1.432e-05	2.159e-07	0.612	0.0031	1.419e-05	3.386e-07

$R_{relative}$.

3.7 Concluding remarks

The experimental comparative analysis is based on the realization of HSS compression and ULV-like factorization in the software library STRUMPACK. The analysis shows the better performance of the sequential algorithms in comparison with tile LU factorization. In the same time the acquired parallel performance and speed-up with STRUMPACK are lower, which is explained by their more complex hierarchical and recursive nature.

The accuracy and computational performance of the HSS compression is highly dependent on the relative ε_{rel} and absolute ε_{abs} thresholds as well as the existence of suitable matrix *structure*. The observed relative error has similar value to the corresponding relative threshold. In order to improve the *structure* of the matrix five reordering schemes are studied. The presented numerical analysis shows that

the recursive bisection has the best results in most of the experiments.

The *structure* of the matrix obtained from the fractional diffusion problem is less suitable than the problem investigated in Chapter 2. This can be explained with the fact that the *fractional* Laplacian is strongly non-local. The examined reordering schemes significantly improve the effectiveness of STRUMPACK. Despite this the parallel performance of the HSS compression is significantly lower than the direct Gaussian elimination.

One of the advantages of the HSS compression is that when solving sequence of systems of linear algebraic equations, in which the matrix doesn't change, the lower computational complexity of the solve with the factorized matrix $O(nr)$ has advantage over the solving after the LU decomposition – $O(n^2)$. Such problem is the case with the problem studied in the next chapter – a parabolic fractional diffusion with lumped mass matrix.

Chapter 4 Finite Element Method for solving two dimensional parabolic fractional diffusion problems

The main interest in the chapter is solving systems with an already factorized matrix after applying the HSS based method. This step, after HSS compression and ULV-like factorization, has computational complexity $O(nr)$ [5]. For comparison, when applying LU factorization, the solving step takes $O(n^2)$ arithmetic operations. With the stationary elliptic problem this step is applied only once and has almost no impact on the performance of the solvers. This is changed when the numerical method is used to solve a sequence of systems of linear algebraic equations. In this case the relative weight of the solving after factorization step is increased substantially. Such problem is the finite element method discretization of a parabolic *fractional* diffusion problem examined in this chapter.

In order to generate the matrix of mass an algorithm and a software module are developed. It uses the information for the triangulation geometry $\mathcal{T} \in \Omega$. For the discretization in time we will use an implicit Euler scheme with an uniform time step and a lumped matrix of mass.

For the Numerical experiments we use settings analogous to the test example in [31] by Vabishchevich. This allows the numerical results, corresponding to the two methods to be compared.

4.1 Problem statement

We use the integral representation (9) of the *fractional* Laplacian. The following parabolic problem with unknown function $u(x, t)$, $(x, t) \in \Omega \times [0, T]$ is considered

$$\left\{ \begin{array}{l} \frac{\partial u(x, t)}{\partial t} + (-\Delta)^\alpha u(x, t) = f(x, t), \quad x \in \Omega, \quad t \in [0, T], \\ u(x, t) = 0, \quad x \in \Omega^c, \quad t \in [0, T], \\ u(x, 0) = u^0(x), \quad x \in \Omega. \end{array} \right.$$

Here $[0, T]$ is the time interval. The Dirichlet homogeneous boundary conditions are applied as in the previous problem.

We apply the same triangulation $\mathcal{T} \in \Omega$ as in the previous chapter. The following Cauchy problem is obtained

$$M_L \frac{d\mathbf{u}}{dt} + K\mathbf{u} = M_L \mathbf{f}, \quad 0 < t \leq T, \quad \mathbf{u}(0) = \mathbf{u}^0,$$

for the unknown functions $\mathbf{u} = (u_j(t)) \in \mathbb{R}^N$, $t \in [0, T]$ and right hand side $\mathbf{f} = (f_j(t)) \in \mathbb{R}^N$. Here $K = K_{ij} \in \mathbb{R}^{N \times N}$ is the stiffness matrix corresponding to the *fractional* Laplacian. It has the form defined in (12) and (14). With $M_L = \text{diag}(m_L^i) \in \mathbb{R}^N$ we denote the lumped matrix of mass, where m_L^i is the concentrated mass at node x_i . The algorithm and programming module for the computation of M_L can be found in Appendix B.

For the discretization over time we use the implicit Euler method, which in the general case has the form

$$M_L \frac{\mathbf{u}^{j+1} - \mathbf{u}^j}{\tau_j} + K\mathbf{u}^{j+1} = M_L \frac{\mathbf{f}^{j+1} + \mathbf{f}^j}{2}, \quad j = 0, \dots, m-1, \quad (15)$$

where m is the amount of time steps,

$$\sum_{j=0}^{m-1} \tau_j = T,$$

$t_0 = 0$, $t_{j+1} = t_j + \tau_j$ and $\mathbf{u}^j = \mathbf{u}(t_j)$, $\mathbf{f}^j = \mathbf{f}(t_j)$.

In this dissertation we will limit ourselves to the case of a constant time step $\tau_j = \tau$. Under this condition, each step in (15) comes down to solving the system of linear algebraic equations

$$\tilde{K}\mathbf{u}^{j+1} = \tilde{\mathbf{f}}^j, \quad (16)$$

where

$$\tilde{K} = \frac{M_L}{\tau} + K, \quad \tilde{\mathbf{f}}^j = M_L \left(\frac{\mathbf{f}^{j+1} + \mathbf{f}^j}{2} + \frac{\mathbf{u}^j}{\tau} \right).$$

The mass and stiffness matrices are symmetric and positive definite thus (16) has a unique solution. This means that in the implementation of Euler's method the factorization step is performed once, then we solve m systems with the factorized matrix.

The computational complexity of LU factorization is $O(n^3)$. Then the m steps in time require another $O(\text{left}(n^2m \text{ right}))$ arithmetic operations.

In the hierarchical method – HSS compression and ULV-like factorization have a summary complexity of $O(rn^2)$ and $O(nrm)$ arithmetic operations are needed to solve the factorized systems. Thus, determining the effectiveness of the hierarchical method is the max off-diagonal rank r .

Several reordering schemes were introduced in Chapter 3 for solving the *fractional* diffusion problem. Here we will show only results for the most effective – recursive bisection.

For the numerical experiments we will use a problem analogous to the parabolic problem studied by Vabishchevich in [31], where the spectral definition of a *fractional* Laplacian is used. The problem is solved for $(x, t) \in \Omega \times [0, T] = (-1, 1)^2 \times (0, 0.1)$. The solution is determined from the time independent right hand side

$$f(x) = \frac{(x_1 + 1)(x_2 + 1)}{4},$$

and initial condition

$$u^0(x) = 100 \left(\frac{x_1 + 1}{2} \right)^2 \left(1 - \frac{x_1 + 1}{2} \right) \left(\frac{x_2 + 1}{2} \right)^2 \left(1 - \frac{x_2 + 1}{2} \right)$$

The discretization in time uses the triangulation from Chapter 3 with time step $\tau = T/m$, $m = 256$.

The initial condition $u^0(x)$ and numerical solutions with $\alpha = 0.5$ for $t \in \{0.025, 0.05, 0.075, 0.1\}$ are shown on Figure 15. The solution obtained is qualitatively similar to the results presented in [31].

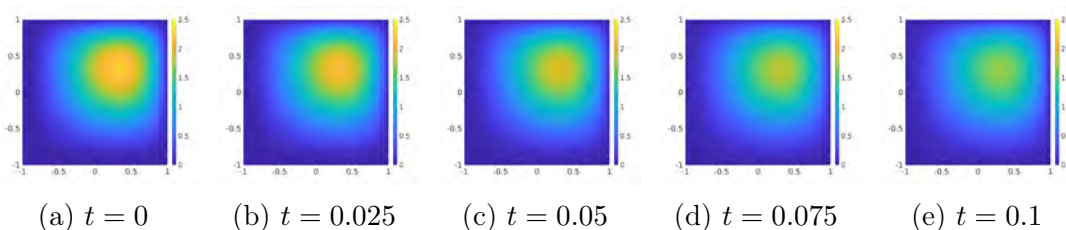


Figure 15: Numerical solutions of the model parabolic fractional diffusion problem with $\alpha = 0.5$: FEM in space and Euler's implicit method in time.

4.2 Analysis of numerical experiments on systems with shared memory

The numerical results analyzed in this chapter are obtained on computer systems with shared memory. As in previous chapters, the experiment was performed on the AVITOHOL supercomputer.

The MatLab program, published in [2], was used to generate the stiffness matrix K and the right part \mathbf{f} . For the reordering of the unknowns with recursive bisection we use the algorithm described in Section 3.4.4 and the developed code presented in Annex A.4. To calculate the lumped mass matrix M_L we use the algorithm presented in Appendix B.

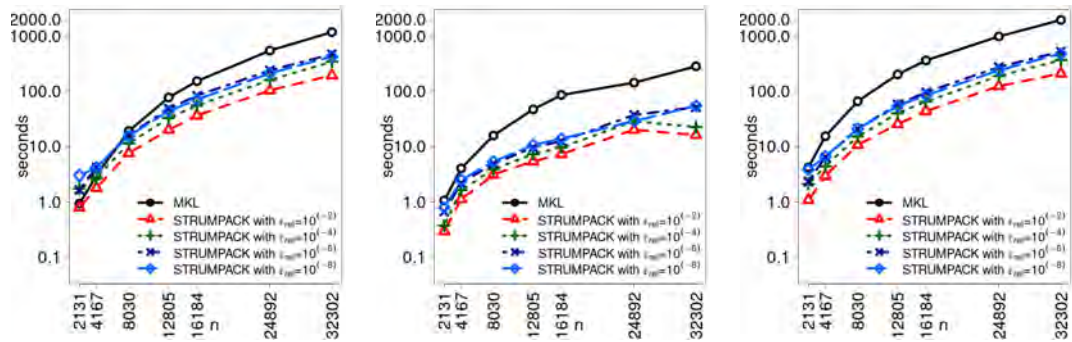
4.3 Sequential and parallel experiments.

On Figure 16 we present: the execution times of the hierarchical semi-separable compression and ULV-like factorization from STRUMPACK and LU factorization from MKL – (Figures 16a and 16d); the total time for solving the systems with the factorized matrices of each of the time steps in Euler’s method – (Figures 16b and 16e); and the total time – (Figures 16c and 16f). In almost all cases, the numerical experiments show better efficiency of the hierarchical method, and this tends to increase with the number of unknowns n . For the largest system ($n = 32\,302$) the total time with the HSS compression method from the STRUMPACK package are between ~ 2.5 and ~ 5 times better than when using the LU factorization from the MKL package. These results confirm the theoretical expectations for stronger improvement of the efficiency of the hierarchical solver for the parabolic problem, when compared to the stationary problem discussed in the previous chapter.

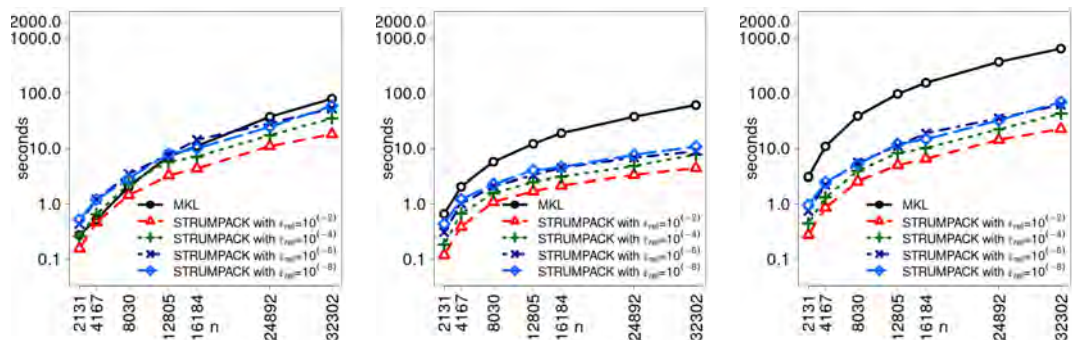
The parallel speed of the HSS based solver from STRUMPACK is shown on Figure 17. The parallel experiments with 16 threads show speed up from ~ 3 (for $n = 2\,131$) up to ~ 8 (for $n = 32\,302$). The lower parallel speed up of the hierarchical method can be explained by the more complex hierarchical and recursive structure of the compression algorithm. The parallel implementation of the solver with HSS factorized matrix also has more complex (and less balanced) structure than the tile LU factorization.

4.4 Off-diagonal rank

The maximum off-diagonal rank r , calculated during the HSS compression, is presented on Figure 18a, on Figure 18b the relation n/r is shown. This rank is a measure of the efficiency of the compression, and at the same time is detrimental in estimating the computational complexity. For the examined problem r has much smaller values than n . The compression is strong, meaning that r/n is small, with

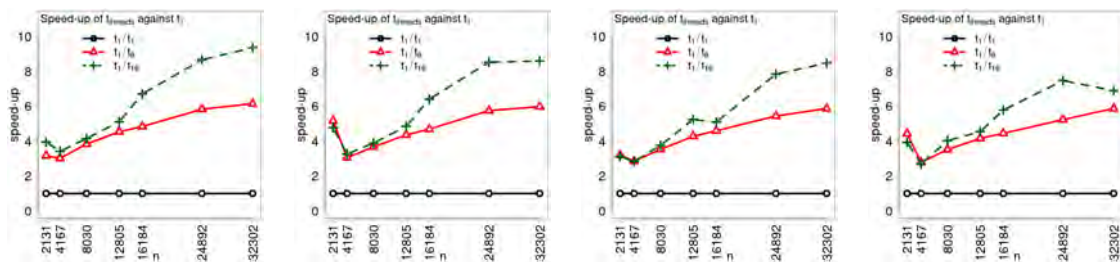


(a) Sequential times: compression and factorization (b) Sequential times: Time steps (c) Sequential times: Total



(d) Parallel times with 16 threads: compression and factorization (e) Parallel times with 16 threads: Time steps (f) Parallel times with 16 threads: Total

Figure 16: Comparison between the solution times for the parabolic problem (16) with MKL and STRUMPACK with a relative threshold $\epsilon_{rel} \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$.



(a) $\epsilon_{rel} = 10^{-2}$ (b) $\epsilon_{rel} = 10^{-4}$ (c) $\epsilon_{rel} = 10^{-6}$ (d) $\epsilon_{rel} = 10^{-8}$

Figure 17: Parallel speed up in solving the parabolic problem with the application of the HSS-based solver for $m = 256$ time steps.

the largest values of the relative threshold ε_{rel} . Thus with $\varepsilon_{\text{rel}} = 10^{-2}$ the rank r is between ~ 20 and ~ 80 times smaller than n , while with the finest threshold $\varepsilon_{\text{rel}} = 10^{-8}$ this relation is between ~ 10 and ~ 30 . This analysis shows that that the reordered with recursive bisection matrix \tilde{K} has suitable *structure* for HSS compression. This is also confirmed by the numerical experiments showcasing the advantage of the hierarchical method over Gaussian elimination (block LU factorization). It is important to note that the compression is *approximate*. Thus, the higher compression efficiency, obtained with higher threshold values ε_{rel} , is at the expense of the lower accuracy of the solution.

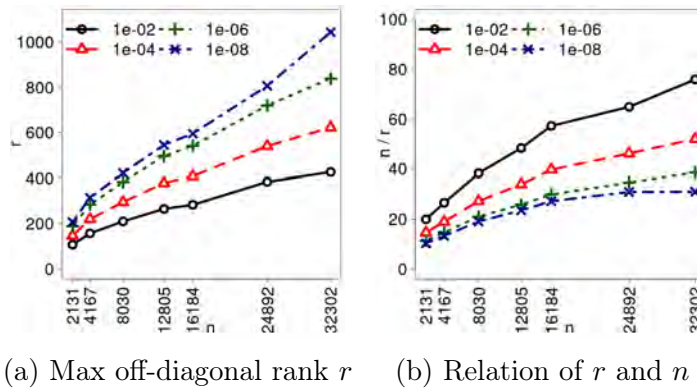


Figure 18: Visualization of the max off-diagonal rank r and relation n/r .

4.5 Analysis of the relative error of the HSS-based solver

The compressed matrix H , obtained after the HSS compression is an approximation of \tilde{K} . As with the stationary problem in the previous chapter, we will analyze the relative error R_{relative} (7).

The relative errors R_{relative} for 4 chosen values of t are presented in Table 5. As it was with the stationary problem the relative error is close to the supplied relative threshold ε_{rel} . The presented numerical results also show that the relative error doesn't increase substantially with the increase of the time interval. This confirms the stability of Euler's implicit method.

4.6 Concluding remarks

The main topic in the presented results is the analysis of the computational efficiency of a method based on Hierarchical Semi-Separable compression and ULV-like factorization and its parallel implementation in the STRUMPACK software package. On specific of the applied implicit Euler method with a constant step τ

Table 5: Relative error of the HSS-based solver.

(a) $t = 0.025$ and $t = 0.05$

n	Relative error at $t = 0.025$				Relative error at $t = 0.05$			
	10^{-2}	$R_{relative}$ for r_{tol} 10^{-4}	10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ for r_{tol} 10^{-4}	10^{-6}	10^{-8}
2131	0.00385	$7.52e-05$	$2.78e-07$	$5.53e-08$	0.00659	0.000124	$4.62e-07$	$9.72e-08$
4167	0.006	0.00013	$6.4e-07$	$8.62e-08$	0.01	0.000225	$1.03e-06$	$1.36e-07$
8030	0.0083	0.00023	$1.28e-06$	$2.17e-07$	0.0146	0.000375	$2.e-06$	$3.49e-07$
12805	0.0106	0.00028	$1.68e-06$	$4.77e-07$	0.0178	0.000484	$2.49e-06$	$7.56e-07$
16184	0.0126	0.0003	$1.75e-06$	$5.16e-07$	0.0227	0.00052	$2.77e-06$	$7.88e-07$
24892	0.0192	0.000393	$2.5e-06$	$9.69e-07$	0.0349	0.000619	$3.97e-06$	$1.49e-06$
32302	0.0234	0.000345	$2.48e-06$	$1.18e-06$	0.0437	0.000537	$3.97e-06$	$1.76e-06$

(b) $t = 0.075$ и $t = 0.1$

n	Relative error at $t = 0.075$				Relative error at $t = 0.1$			
	10^{-2}	$R_{relative}$ for r_{tol} 10^{-4}	10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ for r_{tol} 10^{-4}	10^{-6}	10^{-8}
2131	0.0088	0.000159	$5.98e-07$	$1.32e-07$	0.0108	0.000188	$7.07e-07$	$1.62e-07$
4167	0.0135	0.000298	$1.31e-06$	$1.707e-07$	0.0166	0.000361	$1.55e-06$	$1.98e-07$
8030	0.0206	0.000498	$2.52e-06$	$4.55e-07$	0.0264	0.000606	$2.93e-06$	$5.49e-07$
12805	0.0242	0.000653	$3.09e-06$	$9.74e-07$	0.0301	0.0008	$3.6e-06$	$1.16e-06$
16184	0.0324	0.000706	$3.6e-06$	$1.01e-06$	0.0419	0.000875	$4.34e-06$	$1.2e-06$
24892	0.0499	0.000807	$5.19e-06$	$1.92e-06$	0.0646	0.000973	$6.26e-06$	$2.3e-06$
32302	0.0636	0.000693	$5.25e-06$	$2.21e-06$	0.0832	0.000826	$6.41e-06$	$2.58e-06$

is that the numerical solution of the parabolic the problem is reduced to $m = T/\tau$ systems of linear algebraic equations with the same matrix \tilde{K} , only changing the right hand side at each time step. This means that the matrix \tilde{K} is factorized only once and the accent falls on the solution of m systems with a factorized matrix. Recall that this step has a computational complexity $O(n^2)$ for the Gaussian solver and $O(nr)$ for the hierarchical solver. The presented analysis shows that, for the considered problem, the rank r is significantly smaller than the number of unknowns n , which determines the advantage of the hierarchical method. As a result, both sequential and parallel solution times utilizing the solver from the STRUMPACK software package are significantly better than the execution times with the LU factorization from MKL.

The relative error $R_{relative}$ is close to the set relative threshold, growing steadily with the development of the process over time. This confirms that the hierarchical method provides good accuracy with suitably chosen ε_{rel} . The presented numerical results confirm the significant advantage of the Hierarchical Semi-Separable compression method from the STRUMPACK library.

Conclusion

In this dissertation we analyze the computational efficiency of numerical methods and algorithms for solving systems of linear algebraic equations with dense matrices. The motivation for this study are applications related to the numerical solution of elliptical and parabolic partial differential equations. Two such problems are used in the presented comparative analysis: a) boundary value problem describing laminar flow around Zhukovsky airfoils, discretized with the Boundary Elements Method; b) anomalous diffusion inside a bounded domain modeled with the *fractional* Laplacian, where the finite element method is applied for the discretization. In both cases the problems are reduced to systems of linear algebraic equations with dense matrices. It is shown that the *structure* of these matrices is suitable for applying a hierarchical method using HSS compression.

An important part of Chapter 2 is the comparative analysis of the computational complexity of software packages implementing tile LU factorization, a variant of Gaussian elimination. The general conclusion is that the MKL library has better performance than the examined alternative implementations of LU factorization.

The accent of the dissertation is the analysis of the possibility for improvement of the computational efficiency of solving systems of linear algebraic equations with dense matrices with the help of an hierarchical method utilizing HSS compression. This method is implemented in the STRUMPACK software package. In Chapters 2 and 3 the performance of the hierarchical algorithm for systems of linear algebraic equations obtained from the application of the boundary element and finite element methods, respectively, for the considered elliptic boundary problems. The analysis shows that the studied dense matrices have a suitable *structure* for the application of the hierarchical method. This means that the HSS compression finds low-rank off-diagonal blocks.

The sequential experiments affirm the computational complexity measures of the analyzed tile methods. For both problems the hierarchical solver shows better performance than the MKL Gaussian elimination algorithm – the most efficient from the studied software packages implementing LU factorization.

When applying the hierarchical method we obtain an approximate solution of the system. Its accuracy depends on the accuracy of the HSS compression. We base the analysis of the relative error R_{relative} of the numerical experiments on using the solution obtained from LU factorization as reference. For the fractional diffusion problem the relative error is observed to be close to the set threshold ε_{rel} . This can be accepted as a good characteristic for the HSS compression solver. For the flow around Zhukovsky airfoils problem the relative error for the corresponding ε_{rel} is larger, but this is compensated with better execution time performance.

It is well known that the quality of the Hierarchical Semi-Separable compression depends heavily on the *structure* of the dense matrix. With the two dimensional fractional diffusion problem the *structure* of the originally obtained dense matrix is not suitable for HSS compression. In order to improve it five reordering schemes are applied. The presented analysis shows the advantages of using Nested Dissection and Recursive Bisection.

In Chapter 4 we study the computational performance and accuracy of the hierarchical solver based on HSS compression for a parabolic problem with *fractional* in space diffusion. The discretization in time is carried out with an implicit Euler differential scheme with uniform step. With this problem finding the numerical solution of the problem is reduced to a sequence of systems of linear algebraic equations with the same matrix. In this way on every time step we solve a system that is factorized once only. Solving such systems with HSS compression has lower computational complexity – $O(nr)$ in comparison to the LU factorization's $O(n^2)$. For the examined parabolic problem the hierarchical solver's execution times are better both for the sequential and parallel experiments. At the same time, thanks to the unconditional stability of the implicit Euler method, the relative error is close to the set relative threshold ε_{rel} .

List of publication on the dissertation

The main results presented in this dissertation are published in 5 printed and 2 in press papers as follows:

- [26] D. Slavchev and S. Margenov. Performance Analysis of Intel Xeon Phi MICs and Intel Xeon CPUs for Solving Dense Systems of Linear Algebraic Equations: Case Study of Boundary Element Method for Flow Around Airfoils. In K. Georgiev, M. Todorov, and I. Georgiev, editors, *Advanced Computing in Industrial Mathematics: BGSIAM 2017*, pages 369–381, Cham, 2019. Springer International Publishing. ISBN 978-3-319-97277-0. doi: 10.1007/978-3-319-97277-0_30
- [25] D. Slavchev and S. Margenov. Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs. In G. Nikolov, N. Kolkovska, and K. Georgiev, editors, *Numerical Methods and Applications*, pages 466–473, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10692-8
- [27] D. Slavchev and S. Margenov. Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for bem discretization of flow around airfoils. In *Advanced Computing in Industrial Mathematics*, Cham, in press. Springer International Publishing

- [23] D. Slavchev. On the impact of reordering in a hierarchical semi-separable compression solver for fractional diffusion problems. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, pages 373–381, Cham, 2020. Springer International Publishing. ISBN 978-3-030-41032-2
- [24] D. Slavchev. Performance Analysis of Hierarchical Semi-separable Compression Solver for Fractional Diffusion Problems. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 333–344, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5
- [29] D. Slavchev, S. Margenov, and I.G. Georgiev. On the application of recursive bisection and nested dissection reorderings for solving fractional diffusion problems using hss compression. In *AIP Conference Proceedings*, volume 2302, page 120008, 2020. ISBN 978-0-7354-4036-4. doi: 10.1063/5.0034506
- [28] D. Slavchev and S. Margenov. Performance study of hierarchical semi-separable compression solver for parabolic problems with space-fractional diffusion. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, Cham, in press. Springer International Publishing

Approbation of the results

The results presented in this dissertation were reported in the following international conferences and workshops.

International Conferences:

- Large-Scale Scientific Computations (LSSC), Sozopol, 2017, 2019, 2021
- Annual Meeting of the Bulgarian Section of SIAM (BGSIAM), Sofia, 2017, 2018, 2019
- Numerical Methods for Scientific Computations and Advanced Applications (NMSCAA), Hisarya, 2018
- Numerical Methods and Applications (NM&A), Borovets, 2018
- Twelfth On-Line Conference of the Euro-American Consortium for Promoting the Application of Mathematics in Technical and Natural Sciences (AMi-TaNS), Albena, 2020

Workshops:

- Две години Авитохол: Иновативни Суперкомпютърни Приложения (Two years Avitohol: Innovative supercomputing applications), Panagyurishte, 2017
- Numerical Solution of Fractional Differential Equations and Applications (NSFDE&A), Sozopol, 2020

Main scientific and applied scientific contributions

1. The performance of the following software packages for solving systems linear algebraic equations with dense matrices, using block LU factorization, was studied: for General Purpose Processors (CPUs) – Intel Math Kernel Library (MKL) and the open access package Parallel Linear Algebra Software for Multicore Architectures (PLASMA); for accelerators with the Intel Many Integrated Core (MIC) architecture – MKL and the Matrix Algebra on GPU and Multicore Architectures (MAGMA) open access package. The results of the numerical experiments for systems obtained from Boundary Elements Method discretization of a boundary value problem of laminar flow around Zhukovsky airfoils are consistent with asymptotic estimates for computational complexity. The comparative analysis shows better performance and very good parallel scalability of the MKL package.
2. We studied the numerical complexity, parallel performance and relative error of an Hierarchically Semi-Separable compression (HSS). Numerical experiments are carried out with the free access library STRUctured Matrices PACKage (STRUMPACK), where a parallel solver based on HSS compression and ULV-like factorization is implemented. The comparative analysis includes two types of dense matrices, obtained from discretization with: a) Boundary Element Method of a boundary value problem of a laminar flow around Zhukovsky airfoils; b) Finite Element Method for two dimensional fractional diffusion boundary value problem. A comparative analysis of the solvers from MKL and STRUMPACK is carried out. A characterization dependent on the relative threshold of HSS compression is obtained for the cases, where the hierarchical method has better performance.
3. We show that for the flow around Zhukovsky airfoils problem with boundary element method discretization the sequential ordering of the nodes on the borders of the airfoils has suitable *structure* for HSS compression. This is not the case for the fractional diffusion boundary problem, discretized with Finite Element Method. In order to improve the effectiveness of the Hierarchical Semi-Separable compression we propose and study five methods for reordering of the unknowns. The comparative analysis shows significant

improvement of the results when nested dissection or recursive bisection are applied.

4. A method, algorithm and program implementation for the numerical solution of a parabolic in space fractional diffusion problem are developed. The discretization in time is carried out with an implicit Euler method with uniform time steps and lumped mass matrix. It is shown that for, this non-stationary problem, the computational complexity of the separate parts of the algorithm creates suitable conditions for advantageous use of the solver based on HSS compression. This is affirmed by numerical experiments. For all studied sizes in space of the discretized problem, as well as all studied relative threshold values, the STRUMPACK solver has better performance than MKL.

Bibliography

- [1] G. Acosta and J.P. Borthagaray. A Fractional Laplace Equation: Regularity of Solutions and Finite Element Approximations. In *SIAM Journal on Numerical Analysis*, volume 55, pages 472–495, 2017. doi: 10.1137/15M1033952.
- [2] G. Acosta, F.M. Bersetche, and J.P. Borthagaray. A short FE implementation for a 2d homogeneous Dirichlet problem of a fractional Laplacian. In *Computers & Mathematics with Applications*, volume 74, pages 784 – 816, 2017. doi: <https://doi.org/10.1016/j.camwa.2017.05.026>.
- [3] M. Benzi, D. Bini, D. Kressner, H. Munthe-Kaas, and C. Van Loan. *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications*. Springer, Cham, Cetraro, Italy, 2015. ISBN 978-3-319-49887-4. doi: 10.1007/978-3-319-49887-4.
- [4] K. Binder, Ch. Bennemann, J. Baschnagel, and W. Paul. Anomalous diffusion of polymers in supercooled melts near the glass transition. In A. Pełkalski and K. Sznajd-Weron, editors, *Anomalous Diffusion From Basics to Applications*, pages 124–139, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49259-7.
- [5] S. Chandrasekaran, M. Gu, and T. Pals. A fast *ulv* decomposition solver for hierarchically semiseparable representations. In *SIAM J. Matrix Anal. Appl.*, volume 28, page 603–622, USA, August 2006. Society for Industrial and Applied Mathematics. doi: 10.1137/S0895479803436652.
- [6] S. Chaturapruek, J. Breslau, D. Yazdi, T. Kolokolnikov, and S.G. McCalla. Crime modeling with Lévy flights. In *SIAM Journal on Applied Mathematics*,

- volume 73, pages 1703–1720. Society for Industrial and Applied Mathematics, 2021/12/06/ 2013. URL <http://www.jstor.org/stable/24510700>.
- [7] J. Dongarra, M. Abalenkovs, A. Abdelfattah, M. Gates, A. Haidar, J. Kurzak, P. Luszczyk, S. Tomov, I. Yamazaki, and A. YarKhan. Parallel programming models for dense linear algebra on heterogeneous systems. In *Supercomputing Frontiers and Innovations*, volume 2, 2016. URL <http://superfri.org/superfri/article/view/90>.
- [8] P Ghysels, X.S. Li, F.-H. Rouet, S. Williams, and A. Napov. An Efficient Multicore Implementation of a Novel HSS-Structured Multifrontal Solver Using Randomized Sampling. In *SIAM Journal on Scientific Computing*, volume 38, pages S358–S384, 2016. doi: 10.1137/15M1010117.
- [9] C. Gorman, G. Chavez, P. Ghysels, T. Mary, F.-H. Rouet, and X. Li. Matrix-free construction of hss representation using adaptive randomized sampling. In *ArXiv*, volume abs/1810.04125, 2018.
- [10] W. Hackbusch. A Sparse Matrix Arithmetic Based on \mathcal{H} -Matrices. Part I: Introduction to \mathcal{H} -Matrices. In *Computing*, volume 62, pages 89–108, Apr 1999. doi: 10.1007/s006070050015.
- [11] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On \mathcal{H}^2 -matrices. In H.-J. Bungartz, R.H.W. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-642-59709-1. doi: 10.1007/978-3-642-59709-1_2.
- [12] S. Harizanov, R. Lazarov, and S. Margenov. A survey on numerical methods for spectral space-fractional diffusion problems:. In *Fractional Calculus and Applied Analysis*, volume 23, pages 1605–1646, 2020. doi: doi: 10.1515/fca-2020-0080.
- [13] S. Harizanov, S. Margenov, and N. Popivanov. Spectral Fractional Laplacian with Inhomogeneous Dirichlet Data: Questions, Problems, Solutions. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 123–138, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5.
- [14] T.A.M. Langlands, B.I. Henry, and S.L. Wearne. Fractional Cable Equation Models for Anomalous Electrodiffusion in Nerve Cells: Finite Domain Solutions. In *SIAM Journal on Applied Mathematics*, volume 71, pages 1168–1203, 2011. doi: 10.1137/090775920.

- [15] P.G. Martinsson. A Fast Randomized Algorithm for Computing a Hierarchically Semiseparable Representation of a Matrix. In *SIAM Journal on Matrix Analysis and Applications*, volume 32, pages 1251–1274, 2011. doi: 10.1137/100786617.
- [16] R. Musina and A.I. Nazarov. On fractional laplacians. In *Communications in Partial Differential Equations*, volume 39, pages 1780–1790. Taylor & Francis, 2014. doi: 10.1080/03605302.2013.864304.
- [17] V. Pasheva and R. Lazarov. Boundary element method for 2D problems of ideal fluid flows with free boundaries. In *Advances in Water Resources*, volume 12, pages 37–45, 1989. doi: 10.1016/0309-1708(89)90014-6.
- [18] E. Rebrova, G. Chávez, Y. Liu, P. Ghysels, and X.S. Li. A Study of Clustering Techniques and Hierarchical Matrix Formats for Kernel Ridge Regression. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 883–892, 2018. doi: 10.1109/IPDPSW.2018.00140.
- [19] L. Rosasco, M. Belkin, and E. De Vito. On learning with integral operators. In *Journal of Machine Learning Research*, volume 11, pages 905–934, 2010. URL <http://jmlr.org/papers/v11/rosasco10a.html>.
- [20] F.-H. Rouet, X.S. Li, P. Ghysels, and A. Napov. A Distributed-Memory Package for Dense Hierarchically Semi-Separable Matrix Computations Using Randomization. In *ACM Trans. Math. Softw.*, volume 42, pages 27:1–27:35, New York, NY, USA, June 2016. ACM. doi: 10.1145/2930660.
- [21] H.D. Simon and S.-H. Teng. How Good is Recursive Bisection? In *SIAM Journal on Scientific Computing*, volume 18, pages 1436–1445, 1997. doi: 10.1137/S1064827593255135.
- [22] D. Slavchev. Parallelization of boundary element method for laplasian equation. Master’s thesis, Technical University - Sofia, 2014.
- [23] D. Slavchev. On the impact of reordering in a hierarchical semi-separable compression solver for fractional diffusion problems. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, pages 373–381, Cham, 2020. Springer International Publishing. ISBN 978-3-030-41032-2.
- [24] D. Slavchev. Performance Analysis of Hierarchical Semi-separable Compression Solver for Fractional Diffusion Problems. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*,

- pages 333–344, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5.
- [25] D. Slavchev and S. Margenov. Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs. In G. Nikolov, N. Kolkovska, and K. Georgiev, editors, *Numerical Methods and Applications*, pages 466–473, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10692-8.
- [26] D. Slavchev and S. Margenov. Performance Analysis of Intel Xeon Phi MICs and Intel Xeon CPUs for Solving Dense Systems of Linear Algebraic Equations: Case Study of Boundary Element Method for Flow Around Airfoils. In K. Georgiev, M. Todorov, and I. Georgiev, editors, *Advanced Computing in Industrial Mathematics: BGSIAM 2017*, pages 369–381, Cham, 2019. Springer International Publishing. ISBN 978-3-319-97277-0. doi: 10.1007/978-3-319-97277-0_30.
- [27] D. Slavchev and S. Margenov. Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for bem discretization of flow around airfoils. In *Advanced Computing in Industrial Mathematics*, Cham, in press. Springer International Publishing.
- [28] D. Slavchev and S. Margenov. Performance study of hierarchical semi-separable compression solver for parabolic problems with space-fractional diffusion. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, Cham, in press. Springer International Publishing.
- [29] D. Slavchev, S. Margenov, and I.G. Georgiev. On the application of recursive bisection and nested dissection reorderings for solving fractional diffusion problems using hss compression. In *AIP Conference Proceedings*, volume 2302, page 120008, 2020. ISBN 978-0-7354-4036-4. doi: 10.1063/5.0034506.
- [30] H. Taitelbaum. Diagnosis using photon diffusion: From brain oxygenation to the fat of the atlantic salmon. In A. Pełkalski and K. Sznajd-Weron, editors, *Anomalous Diffusion From Basics to Applications*, pages 160–174, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49259-7.
- [31] P.N. Vabishchevich. Splitting schemes for non-stationary problems with a rational approximation for fractional powers of the operator. In *Applied Numerical Mathematics*, volume 165, pages 414–430, 2021. doi: <https://doi.org/10.1016/j.apnum.2021.03.006>.
- [32] R.C. Whaley. Atlas (automatically tuned linear algebra software). In D. Padua, editor, *Encyclopedia of Parallel Computing*, pages 95–101,

Boston, MA, 2011. Springer US. ISBN 978-0-387-09766-4. doi: 10.1007/978-0-387-09766-4_85.

- [33] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. In *Numerical Lin. Alg. with Applic.*, volume 17, pages 953–976, 2010. doi: 10.1002/nla.691.
- [34] С. Маргенов. *Числени методи за системи с разредени матрици*. Институт по паралелна обработка на информацията, Българска академия на науките, 2007. ISBN 9780198520115.



БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

АВТОРЕФЕРАТ НА ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен “доктор” по научна специалност “Изчислителна математика“

СЪСТАВНИ ЧИСЛЕНИ МЕТОДИ И СКАЛИРУЕМИ БЛОЧНИ АЛГОРИТМИ

Димитър Георгиев Славчев

Ръководител: Проф. Светозар Маргенов

Научно жури:

**Доц. Кирил Щерев
Проф. Светослав Марков
Доц. Миглена Колева
Проф. Иван Лирков
Проф. Пенчо Маринов**



**Институт по информационни и
комуникационни технологии**

**Секция „Научни пресмятания с Лаборатория по 3D
дигитализация и микроструктурен анализ“**

Обща характеристика на дисертационния труд

Актуалност на темата

Численото решаване на задачи с голяма размерност изисква използването на високопроизводителни изчислителни компютърни системи, както и специализиран хардуер и софтуер – графични карти, ускорители, високоскоростна комуникация между сървърите на системата, софтуерни стандарти и пакети за комуникация между процесорни ядра и сървъри, софтуерни пакети имплементиращи ефективни числени методи и много други.

Съществуват различни методи за дискретизация на диференциални уравнения. Такива например са мрежовите методи като методът на крайните елементи, методът на граничните елементи и методът на крайните разлики. След тяхното прилагане, диференциалните уравнения се свеждат до системи от линейни алгебрични уравнения. Методът на Гаус е универсален подход за решаване на такива системи. В общия случай той има висока изчислителна сложност – $O(n^3)$, където n е броят на неизвестните [29].

При дискретизация на диференциалното уравнение с метода на граничните елементи, както и при прилагане на метода на крайните елементи за нелокални задачи (например изследваната в тази дисертация *аномална* (дробна) дифузия) [2], получената матрица на коравина е *плътна*. Един възможен подход за намаляване на изчислителната сложност на решението на системи с такива матрици е йерархичната компресия въведена от Хакбуш (Hackbusch) [10]. При нея се използва *структурата* на изходната матрица. Целта е както за да се редуцира заеманата памет, така и да се подобри изчислителната ефективност. Тук под *структура* на плътна матрица се разбира наличието на апроксимация на изходящата матрица с нисък ранг в извъндиагоналните блокове. Това свойство позволява представянето на извъндиагоналните блокове като произведение на по-малки матрици. Съществуват различни разновидности на йерархични матрици, в т.ч. \mathcal{H} , \mathcal{H}^2 или йерархични полусепарабелни (Hierarchically Semi-Separable) HSS матрици.

Обзор на основни резултати в областта

Огромният напредък във възможностите на съвременните високопроизводителни изчислителни системи засилва още повече ролята на ефективните числени методи и паралелните алгоритми. Суперкомпютърните симулации са определящи за развитието в редица високотехнологични области. Такива например са *in silico* молекулярната биология и проектиране на лекарствени средства, анализа на турбулентни течения, безразрушителният

контрол, обработката на тримерни изображения, динамика на флуидите и много други.

След подходяща дискретизация математическите модели обикновено се свеждат до задачи на линейната алгебра, измежду които определяща е ролята на решаването на системи от линейни алгебрични уравнения. За целта се разработват специализирани софтуерни средства.

В общия случай за решаване на системи линейни алгебрични уравнения с плътни матрици се прилагат варианти на метода на Гаус, които използват последователно изключване на неизвестните. По предположение плътната матрица е хомогенна, тъй като не се предполага, че тя има нулеви елементи. Методът на Гаус има изчислителна сложност $O(n^3)$. В настоящата дисертация е изследван алтернативен подход на базата на йерархична компресия. Целта е намаляване на изчислителната сложност. Тук под *структура* на плътна матрица се разбира наличието на извъндиагонални блокове с нисък ранг. Поточно предполага се че такова свойство е в сила за матрица, която апроксимира изходната. Наличието на подходяща *структура* е в основата на йерархичната компресия и съответните йерархични методи за решаване на системи с плътни матрици за класове от задачи на изчислителната математика. Йерархичната компресия е въведена от Хакбуш в [10], където са изследвани т.н. \mathcal{H} -матрици. Други разновидности на йерархична компресия са \mathcal{H}^2 -матриците [11] и йерархичните полусепарабелни матрици (HSS) [16]. Теоретична обосновка на методите използващи HSS компресия може да се намери в [28].

Съществена част от дисертацията е посветена на численото решаване на дробно дифузионни задачи. Дробната дифузия (наричана още аномална дифузия) описва нелокални процеси, които се наблюдават в различни физични и социални среди. За разлика от обикновената (локална) дифузия аномалната дифузия включва т.н. бързи преходи или тунелни ефекти. В литературата са публикувани разнообразни примери на математически модели на процеси и явления, които се описват с дробна дифузия. Такива например са: течения в силно нееднородни порести среди, суперпроводимост, дифузия на полимери в суперстудени среди [4]; електродифузия на йони в нервни клетки [15] и диагностика с помощта на фотонна дифузия [25]; обработка на изображения и машинно самообучение [20]; разпространение на вирусни заболявания, компютърни вируси, както и на престъпност [6]. *Дробният* оператор на Лаплас описва аномална дифузия по пространството. Съществуват различни дефиниции на *дробен* лапласиан. Важно е да отбележим, че те не са еквивалентни. Така например в [13] е анализирана разликата между интегралната и спектралната дефиниции (виж също статии [17] и [12] и литературата в тях).

Цели и задачи на дисертацията

Основни цели на дисертацията са:

- Сравнителен анализ на бързодействието и паралелната производителност на често използвани софтуерни пакети прилагачи директна гаусова елиминация за решаване на системи линейни алгебрични уравнения с плътни матрица при използване на централни процесори (CPU) и ускорители (MIC).
- Анализ на бързодействието, паралелното ускорение и точността на приблизителен метод за решаване на системи линейни алгебрични уравнения базиран на йерархична полусепарабелна компресия (HSS) от софтуерния пакет STRUMPACK за системи с подходяща структура на матрицата.
- Разработване на алгоритми за пренареждане на неизвестните при задачи породени от дискретизация с метод на крайните елементи на дробна дифузия с цел подобряване на ефективността на йерархичната полусепарабелна компресия на така изчислената матрица на коравина.
- Числено решаване на елиптични и параболични задачи от областта на аномалната дифузия описана с интегралната формулировка на дробен лапласиан и дискретизирана по пространството с метод на крайните елементи.

Методология на изследването

В дисертацията се анализира ефективността, в смисъл на бързодействие, паралелното ускорение и точността (за приблизителните решения) на блочни методи за решаване на плътни системи линейни алгебрични уравнения. За целта се използват софтуерни пакети, в които са приложени изследваните блочни методи.

При задачата, разгледана в Глава 2 се използва паралелната програма, разработена в [23], за дискретизиране на зададена задача и генериране на системата линейни алгебрични уравнения. При задачите за дробна дифузия, разгледана в Глави 3 и 4, се използва MatLab програмата разработена в [2] от Акоста и др. за генериране на системата линейни алгебрични уравнения. Разработени са програми на MatLab, които изчисляват използваните пренареждания за тази задача (Приложение А) и матрица на масата (Приложение Б). Софтуерните библиотеки със свободен достъп са компилирани върху използваните компютърни системи.

Структура на съдържанието

В Увода е дадена мотивацията за настоящата работа. Накратко са описани използваните методи и решаваните задачи.

Глава 1 има въвеждащ характер и описва използваните блочни методи за решаване на плътни системи линейни уравнения, както и оценка за изчислителните им сложности. В Раздел 1.1 е описан накратко универсалният директен метод гаусова елиминация и базираната на него LU факторизация. В Раздел 1.4 са разгледани йерархични методи за решаване на системи линейни уравнения разработени за решаване на системи със *структурирани* матрици (плътни и разреждени). Описани са и предимствата на разглеждания метод базиран на HSS компресия – по-ниска оценка на изчислителната сложност за задачи с подходяща *структура* на матрицата.

В Глава 2 са представени числени резултати за обтичането на крилни профили на Жуковски. Получената система с *плътна* матрица се използва за бенчмарк при сравнителният анализ на използваните блочни алгоритми.

В Глава 3 е разгледана задача за двумерна *аномална* дифузия моделирана с *дробния* оператор на Лаплас. За дискретизация по пространството се прилага метод на крайните елементи.

В Глава 4 е разгледана параболична задача за двумерна по пространството *аномална* дифузия.

В Заключение са представени обобщаващи бележки за основните резултати получени в дисертацията. Формулирани са научните и научно-приложните приноси. Даден е списък на публикуваните статии и на изнесените доклади на научни форуми върху които се базира тази работа.

Глава 1 Методи за решаване на системи линейни уравнения с плътни матрици

Много задачи от изчислителната практика се решават числено чрез свеждане до система от линейни алгебрични уравнения. Така например при прилагане на *метода на граничните елементи* или при дискретизация с крайни елементи на уравнения с *дробна* степен на оператора на Лаплас (*дробна* дифузия), се получава система с плътна матрица.

1.1 Преки методи

Методът на Гаус е универсален метод за решаване на системи от линейни алгебрични уравнения. Той е основа на повечето преки методи. Така например

LU факторизацията (още наречена декомпозиция) се базира на последователно изключване на неизвестните по метода на Гаус. LU факторизацията е базов метод реализиран във високопроизводителните софтуерни библиотеки на изчислителната линейна алгебра.

1.2 Метод на Гаус

Методът на Гаус за решаване на системата от линейни алгебрични уравнения $Ax = b$ включва прав и обратен ход: (i) Чрез еквивалентни преобразувания матрицата на системата се привежда в горна триъгълна (прав ход, елиминация); (ii) Рекурсивно в обратна последователност се изключват извъндиагоналните елементи в i -тия ред на матрицата за $i = n - 1, n - 2, \dots, 1$ (обратен ход, заместване).

Нека означим с \tilde{A} разширената матрица $\tilde{A} = (A|b)$. Тогава на първата стъпка на правия ход се умножава първия ред на \tilde{A} по $-\frac{a_{i1}}{a_{11}}$ и се прибавя към реда с номер i , за всяко $i = 2, \dots, n$. Правият ход завършва след изпълняване на $n - 1$ такива стъпки. В резултат получаваме горната триъгълна матрица $\tilde{A}^{(n-1)}$.

На обратния ход горната триъгълна матрица $A^{(n-1)}$ се преобразува в диагонална матрица, като за целта се изпълняват $n - 1$ стъпки. На първата стъпка последния ред на разширената матрица се умножава по $\frac{a_{nn}^{(i-1)}}{a_{i-1,n}^{(n-2)}}$ и се събира с i -тия ред за $i = 1, \dots, n - 1$. На втората стъпка се взема $(n - 1)$ -вия ред, като след нейното изпълнение се анулират извъндиагоналните елементи в $(n - 1)$ -вия стълб. Така, след $n - 1$ стъпки на обратния ход, матрицата от коефициенти се свежда до диагонална матрица. Решението на системата се получава след разделяне на i -тия ред със съответния диагонален елемент.

Така в трансформирания вектор $b^{(n-1, n-1)} = x$ получаваме решението на системата. Изчислителната сложност на метода на Гаус се определя от правия ход [29]:

$$\mathcal{N}^{\text{Гаус}} \sim \frac{2n^3}{3} = O(n^3).$$

1.3 LU факторизация

LU факторизацията е изразяване на матрицата A като произведение на две триъгълни матрици $A = LU$. Тук L е долна триъгълна матрица с единици по главния диагонал, а U е горна триъгълна матрица. Тази факторизация се изчислява с помощта на модифициран метод на Гаус и се използва във високопроизводителните библиотеки (LAPACK, MKL, ACML, PLASMA, ATLAS, и други) за решаване на системи линейни уравнения.

Правият ход на метода на Гаус може да се запише във вида

$$\underbrace{L_{n-1}L_{n-2}\dots L_2L_1}_{\tilde{L}}A = U,$$

където L_1, L_2, \dots, L_{n-1} са долни триъгълни матрици с единици по главния диагонал. Непосредствено се проверява, че \tilde{L} и $L = \tilde{L}^{-1}$ са също долни триъгълни матрици с единици по главния диагонал. Така получаваме:

$$\tilde{L}A = U \iff A = LU, \quad L = \tilde{L}^{-1}.$$

След факторизацията на A , системата от линейни алгебрични уравнения се свежда до решаване на две системи с триъгълни матрици. Полагаме

$$L \underbrace{Ux}_y = b,$$

след което се:

1. Решава системата $Ly = b$ с право заместване;
2. Решава системата $Ux = y$ с обратно заместване.

Изчислителната сложност на факторизацията е $O\left(\frac{2}{3}n^3\right)$, докато правото и обратното заместване са със сложност $O(n^2)$.

1.4 Йерархични матрици. Методи за решаване на системи линейни уравнения с помощта на йерархична полусепарабелна компресия

Йерархичните матрици се използват за апроксимация на разреждени по данни (data-sparse) матрици. Под разреждени по данни се разбират матрици, които имат *структура*, позволяваща апроксимация с помощта на компресирани матрици, които се записват с помощта на по-малък брой елементи. В общия случай, разредените по данни матрици не удовлетворяват условието да имат $O(n)$ ненулеви елемента. Хакбуш въвежда понятието „йерархични матрици“ в [10], като разработва теория и алгоритми за работа с т.н. \mathcal{H} -матрици.

Методите използващи йерархични матрици са част от по-общата група от методи за решаване на системи чрез т.н. структурирани матрици. В [3] е направен обзор на съществуващите методи използващи такива матрици, включително и йерархични полусепарабелни матрици. В настоящата дисертация е изследвана ефективността на алгоритми на базата на този клас методи.

STRUMPACK (STRUctured Matrices PACKage) е паралелна софтуерна библиотека, която използва йерархична полусепарабелна компресия за решаване на системи от линейни алгебрични уравнения с плътни матрици [21]. Алгоритъмът включва три стъпки:

1. *Йерархична полусепарабелна компресия (апроксимация) на матрицата на системата.* При наличието на определени предположения изчислителната сложност на тази стъпка е $O(r^2n)$, където r е максималният ранг на извъндиагоналните блокове на апроксимиращата матрица изчислен в процеса на компресия. В общият случай сложността е $O(rn^2)$
2. *ULV-подобна факторизация.* На тази стъпка се факторизира компресираната матрица. За целта се прилага вариант на метода на Гаус, подобен на използвания при представянето на LU факторизацията. Най-напред се изключват $O(r)$ неизвестни, след което се изключват останалите $O(n - r)$. Изчислителната сложност на тази стъпка е $O(r^2n)$.
3. *Решение.* Тази стъпка използва компресираната и факторизирана матрица от коефициенти на системата и дясната страна за намиране на решението. Изчислителната сложност на тази стъпка е $O(rn)$.

Така общата изчислителна сложност на метода е $O(r^2n)$. Както ще видим по-късно, тази оценка е в сила при определени предположения.

1.5 Йерархична полусепарабелна компресия

В този раздел ще разгледаме накратко йерархичните полусепарабелни матрици (Hierarchically Semi-Separable – HSS). Те са въведени от Мартинсон в [16]. В [21] са описани алгоритмите използвани в STRUMPACK за решаване на системи от линейни уравнения с плътни матрици. Йерархичната компресия може да се приложи върху всяка неособена матрица, но е *ефективна* само, ако изходящата матрица A има подходяща *структура* – т.е. извъндиагоналните ѝ блокове имат нисък ранг. Под *ефективна* компресия разбираме апроксимация на матрицата, което води до съществено намаляване на изчислителната сложност на операциите с компресираната матрицата, както и на паметта необходима за нейното съхраняване.

Означаваме HSS компресираната апроксимация на матрицата A с H . Алгоритъмът може да се опише по следния начин:

1. Разделяме матрицата A на четири блока. Предполагаме, че извъндиагоналните блокове имат нисък ранг (и могат да се декомпозират по

сингулярни стойности (Singular Value Decomposition – SVD) или друга факторизация, която изчислява ранг):

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} = \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix}.$$

Матриците U , B и V се наричат *генератори*. Ако извъндиагоналните блокове имат нисък ранг, матриците U са „високи и тънки“, B са малки и квадратни (или близки до квадратни) и матриците V са „ниски и широки“. Съотношението на броя на колоните и редовете зависи от ранга на извъндиагоналните блокове. D са непроменените диагонални блокове на изходящата матрица A . Означението „big“ ще бъде обяснено по-долу в точка 3.

2. Предполагаме, че диагоналните блокове D също имат извъндиагонални блокове с нисък ранг. Те се компресират по аналогичен начин, като процесът продължава рекурсивно. Второто ниво на рекурсивна компресия има вида:

$$A = \begin{bmatrix} \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix} & U_3^{\text{big}} B_{3,6} V_6^{\text{big}*} \\ U_6^{\text{big}} B_{6,3} V_3^{\text{big}*} & \begin{bmatrix} D_4 & U_4^{\text{big}} B_{4,5} V_5^{\text{big}*} \\ U_5^{\text{big}} B_{5,4} V_4^{\text{big}*} & D_5 \end{bmatrix} \end{bmatrix}$$

3. Съществува рекурсивна зависимост между *генераторите* на различните нива компресия. Това обяснява и използването на означенията „big“. В сила са следните зависимости:

$$U_3^{\text{big}} = \begin{bmatrix} U_1^{\text{big}} & 0 \\ 0 & U_2^{\text{big}} \end{bmatrix} U_3 \quad \text{и} \quad V_3^{\text{big}} = \begin{bmatrix} V_1^{\text{big}} & 0 \\ 0 & V_2^{\text{big}} \end{bmatrix} V_3 \quad (1)$$

Третото ниво на рекурсивна HSS компресия се записва във вида:

$$A = \begin{bmatrix} \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix} & \begin{bmatrix} U_1^{\text{big}} & 0 \\ 0 & U_2^{\text{big}} \end{bmatrix} U_3 B_{3,6} V_6^* \begin{bmatrix} V_4^{\text{big}*} & 0 \\ 0 & V_5^{\text{big}*} \end{bmatrix} \\ \begin{bmatrix} U_4^{\text{big}} & 0 \\ 0 & U_5^{\text{big}} \end{bmatrix} U_6 B_{6,3} V_3^* \begin{bmatrix} V_1^{\text{big}*} & 0 \\ 0 & V_2^{\text{big}*} \end{bmatrix} & \begin{bmatrix} D_4 & U_4^{\text{big}} B_{4,5} V_5^{\text{big}*} \\ U_5^{\text{big}} B_{5,4} V_4^{\text{big}*} & D_5 \end{bmatrix} \end{bmatrix} \quad (2)$$

Генераторите с означения „big“ могат да не се изчисляват извън най-високите нива на рекурсивната компресия. U се изчислява от U_τ и от U^{big} в по-високите нива на компресия. В последното ниво на компресия $U = U^{\text{big}}$.

В общия случай равенство 2 не е точно, а приблизително. Това означава, че в резултат на HSS компресията получаваме апроксимация на A , която означаваме с H , т.е. $H \approx A$.

За целта се избира подходящ трешхолд (прагова стойност) ε , който е необходим при изчисляването на *генераторите*. Когато се използва по-голям трешхолд, се получават по-малки *генератори* и съответно компресираната матрица заема по-малко памет и позволява по ефективни операции с нея, но това е за сметка на точността. При избор на по-малък трешхолд е точно обратното.

Както ще покажем в следващите глави, подреждането на неизвестните при асемблиране на матрицата A влияе съществено на ефективността на HSS компресията. Ако матрицата A се пренареди произволно, това с голяма вероятност може да унищожи каквато и да е подходяща за този метод *структура*.

За определени класове задачи е възможно така да се пренаредят неизвестните, че да се подобри съществено *структурата* на матрицата на системата. Така например в [19] са разгледани няколко метода за клъстеризация при използването на хребетообразна регресия с ядро (Kernel Ridge Regression). В Глава 3 ще предложим и анализираме няколко метода за пренареждане на неизвестните за система от линейни алгебрични уравнения, получена при дискретизация на елиптична задача с *дробна* степен на оператора на Лаплас (дробно-дифузионна задача).

1.6 Компресия със случайни извадки

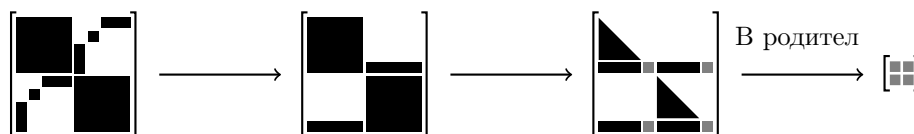
Алгоритъмът за HSS компресия в STRUMPACK се основава на използването на случайни извадки (randomized sampling), който прилага умножаване на множество от случайни вектори с изходната матрица A . Този метод е предложен от Мартинсон в [16]. В алгоритъма не се изисква да разполагаме с матрицата A в явен вид. Вместо това е нужна само функция за умножение на A с вектор. Премуществата на този подход, както и адаптивен алгоритъм за случайни извадки, са разгледани от Горман и др. в [9]. Използването на случайни извадки е полезно също така при интегрирането на HSS ядра в солвъри за задачи с разреждени матрици [8].

В общия случай изчислителната сложност на умножението на плътна матрица с вектор е $O(n^2)$. Това води до сложност на HSS компресирането $O(rn^2)$. За определени класове задачи r е много по-малко от n . Така например за двумерни задачи на Поасон (МКЕ) r е константа, а за тримерни задачи на Хелмхолц (МГЕ) расте бавно при нарастване на n . Ако разполагаме с бърз алгоритъм за умножение на компресираната матрица с вектор, сложността на компресията може да се намали до $O(r^2n)$.

1.7 ULV-подобна факторизация и решение

Компресираната матрица H в HSS форма може да се факторизира със специален вид LU факторизация, наречена ULV факторизация [5]. Тази факторизация използва ортогонални трансформации за последователно изключване на първите $n - r$ неизвестни. Останалите r неизвестни се изключват с LU факторизация. В STRUMPACK е реализирана ULV-подобна факторизация, при която вместо ортогонални трансформации се използва HSS структурата на компресираната матрицата H .

Процесът на ULV-подобна факторизация е илюстриран на Фигура 1.



Фигура 1: ULV-подобна факторизация.

След прилагане на ULV-подобната факторизация системата от линейни алгебрични уравнения $Ax = b$ се свежда до решаване на две системи с триъгълни матрици. Изчислителната сложност на тази последна стъпка е $O(rn)$ [21].

Глава 2 Метод на граничните елементи за числено решаване на двумерна задача за обтичане на крилни профили

В тази глава се разглежда числен метод за компютърна симулация на ламинарен поток около крилни профили на Жуковски. В дисертацията е приложен методът описан в [18], като е разработена програмна реализация за обтичане на каскада от крилни профили от идеален флуид. Методът е базиран на колокация на сплайни с на части линейна интерполация. В [23] е представен паралелен код на програмния език C за решаване на разглежданата задача.

След дискретизиране на интегралните уравнения по метода на граничните елементи се получава система от линейни уравнения с плътна матрица. Резултатите от прилагане на изследваните в дисертацията методи и алгоритми се сравняват с резултати получени по метода на последователно изключване (метод на Гаус) реализиран в няколко популярни софтуерни пакета. Върху CPU процесори в сравнителния анализ на производителността използваме Intel Math Kernel Library (MKL) и Parallel Linear Algebra for Scalable Multi-core

Architectures (PLASMA), докато за Intel Xeon Phi копроцесорите (накратко наричани МІС от името на архитектурата Many Integrated Core) МКL производителността се сравнява и с Matrix Algebra on GPU and Multicore Architectures (MAGMA) за МІС архитектура (наричан за кратко MAGMA МІС).

2.1 Постановка на задачата

2.2 Метод на граничните елементи за пресмятане на токовата функция на идеален флуид в неограничена двумерна област

Нека $\Omega \subset \mathbb{R}^2$ е неограничена многосвързана област с достатъчно гладка вътрешна граница S . За разглежданата задача функцията на тока Ψ удовлетворява уравнението на Лаплас

$$\nabla^2 \Psi \equiv \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = 0 \quad (3)$$

в $\Omega \subset \mathbb{R}^2$ и може да се представи във вида

$$\Psi(P) = -\frac{1}{4\pi} \int_S \gamma(\sigma) \ln(r^2(P, Q)) d\sigma_Q + \Psi_\infty(P) + C_0, \quad P \in \Omega. \quad (4)$$

където $r^2(P, Q) = (x - \xi)^2 + (y - \eta)^2$, $P = (x, y)$, $Q = (\xi, \eta)$ и $d\sigma_Q$ е мярка върху S . Първият член на дясната страна съответства на прост слой с плътност $\gamma(\sigma)$, $\Psi_\infty(P)$ е хармонична функция, добавена, за да бъдат удовлетворени условията на външната граница, т.е. при $P \rightarrow \infty$. При тези предположения за полето на скоростите $\vec{C} = (u, v)$,

$$u = \frac{\partial \Psi}{\partial y}, \quad v = \frac{\partial \Psi}{\partial x}$$

са в сила уравненията

$$u = \frac{1}{2\pi} \int_S \gamma(\sigma) \frac{y - \eta}{r^2} d\sigma, \quad v = -\frac{1}{2\pi} \int_S \gamma(\sigma) \frac{x - \xi}{r^2} d\sigma.$$

2.3 Обтичане на крилни профили

В този раздел ще разглеждаме задачата за обтичане на крилни профили на Жуковски. Приемаме, че флуидното течение в безкрайност е с хомогенна скорост $\vec{C}_\infty = (1, 0)$. Тук с S са означени контурите на крилните профили. За разглежданата задача токовата функция Ψ удовлетворява уравнението

на Лаплас (3). Крилните профили S са непроницаеми. Следователно е в сила граничното условие $\Psi|_S = K = \text{const}$. За да удовлетворим условията в \vec{C}_∞ , избираме такава Ψ_∞ , че

$$\vec{C}_\infty = \left(\frac{\partial \Psi_\infty}{\partial y}, -\frac{\partial \Psi_\infty}{\partial x} \right).$$

В конкретната задача използваме

$$\Psi_\infty(P) = \gamma_\infty(P).$$

Така интегралното уравнение (4) се записва във вида

$$\gamma(P) - \frac{1}{4\pi} \int_S \gamma(\sigma) \ln(r^2(P, Q)) d\sigma_Q + C = 0. \quad (5)$$

За получаване на единствено решение на интегралното уравнение се използва условието на Рунге-Кута $\gamma(A) = 0$, където A са върховете на острите ъгли на крилните профили.

2.4 Дискретизация

За численото решаване на интегралното уравнение (5) прилагаме метода на граничните елементи. Така непрекъснатата задача се свежда до система от линейни алгебрични уравнения, която можем да запишем във вида:

$$(A\gamma) s = f(s).$$

Приближеното решение се търси във вида

$$\gamma_h(S) = \sum_{i=1}^n \gamma_i \phi_i(s).$$

Тук $\{\phi_i(s)\}_{i=1}^n$ е Лагранжевият базис на пространството от на части линейни функции по границата S , върху която е дефинирана мрежата S_h . С $\gamma_i = \gamma_h(s_i)$, $i = 1, \dots, n$ са означени неизвестните стойности на приближеното решение във възлите на мрежата. За реализация на метода на граничните елементи прилагаме метод на колокацията с колокационни точки в средите на елементите от S_h . Следвайки [23] получаваме системата от линейни уравнения

$$\sum_{i=1}^n \gamma_i \Psi_{ji} = f_j, \quad j = 1, 2, \dots, n, \quad (6)$$

където $\Psi_{ji} = \Psi_i(s_j)$, $f(s_j) = f_j$, $\Psi_i(s) = (A\phi_i)(s)$.

В Раздел 2.5 са представени числени експерименти за задачата за обтичане на пет крилни профила разположени вертикално един над друг.

2.5 Анализ на числени експерименти върху компютърни системи с обща памет

Формирането на матрицата D има сложност $O(n^2)$. Сложност $O(n)$ има и изчисляването на коефициентите на челно съпротивление и подемна сила. Фокусът на изследване в тази работа е върху най-тежката изчислителна част – решаването на системата линейни алгебрични уравнения, което с метод на Гаус има сложност $O(n^3)$ [14].

2.6 LU факторизация

2.6.1 CPU процесори с обща памет

PLASMA (Parallel Linear Algebra Software for Multicore Architectures) [7] е софтуерен пакет за решаване на системи линейни уравнения с плътни матрици, който имплементира функциите на стандарта LAPACK.

Ефективността на PLASMA е базирана на силно оптимизирания пакет BLAS (Basic Linear Algebra Subprograms), в който са реализирани основните операции в линейната алгебра – умножение на вектори, матрици и вектори с матрици. За това ниво на изчисления използваме MKL BLAS и ATLAS (Automatically Tuned Linear Algebra Software) BLAS [27].

В Таблица 1 са представени резултатите от проведените числени експерименти за решаване на линейните системи получени при прилагане на метода на граничните елементи за дискретизация на задачата за обтичане на крилни профили. Сравнени последователните и паралелни времена за PLASMA + ATLAS, PLASMA + MKL и MKL за $n = 5\,000$ и $n = 40\,000$, варирайки броя на нишките.

Таблица 1: Последователни и паралелни времена за решаване на системата върху CPU процесори с обща памет

Софтуер		Plasma + ATLAS		PLASMA + MKL		MKL	
Нишки	n	време [s]	ускорение	време [s]	ускорение	време [s]	ускорение
1	5 000	8.42	1.00	5.03	1.00	5.30	1.00
16	5 000	0.67	12.57	0.47	10.69	0.47	11.26
32	5 000	0.88	9.59	0.65	7.76	0.65	8.12
1	40 000	4008.76	1.00	2497.12	1.00	2233.93	1.00
16	40 000	282.94	14.17	166.41	15.01	147.64	15.13
32	40 000	325.17	12.33	169.58	14.73	148.59	15.03

Резултатите показват добро ускорение за всички тествани библиотеки до 16 нишки. Ускорението достига до 15 и е близо до теоретичния максимум 16.

Паралелната ефективност на PLASMA с MKL и MKL е близка, като достига 94% за най-голямата задача ($n = 40\,000$). И двата варианта превъзхождат PLASMA с ATLAS повече от 1.5 пъти.

Таблица 2: Последователни и паралелни времена и ускорения за решаване на системата върху MIC копроцесорите

Пакет		MAGMA MIC		MKL	
Нишки	n	време [s]	ускорение	време [s]	ускорение
1	5 000	11.70	1.00	17.64	1.00
60	5 000	5.81	2.01	2.86	6.16
120	5 000	6.76	1.73	3.55	4.97
240	5 000	5.39	2.17	3.80	4.64
1	40 000	4896.49	1.00	2101.93	1.00
60	40 000	665.53	7.36	154.23	13.63
120	40 000	432.89	11.31	93.80	22.41
240	40 000	208.48	23.49	64.43	32.62

2.6.2 MIC ускорители

В този раздел анализираме паралелната ефективност на Intel Xeon Phi 7120P копроцесори (MICs). MIC-овете са проектирани за масивни паралелни и векторни изчисления необходими при високопроизводителните изчисления. Използваният модел MIC има 61 ядра, като всяко ядро може да изпълнява едновременно инструкции от 4 нишки, т.е. могат да се използват максимум 244 нишки. Използваме Offload режим, при който едно от ядрата се запазва за комуникация с CPU процесора и можем да използваме до 60 ядра (240 нишки).

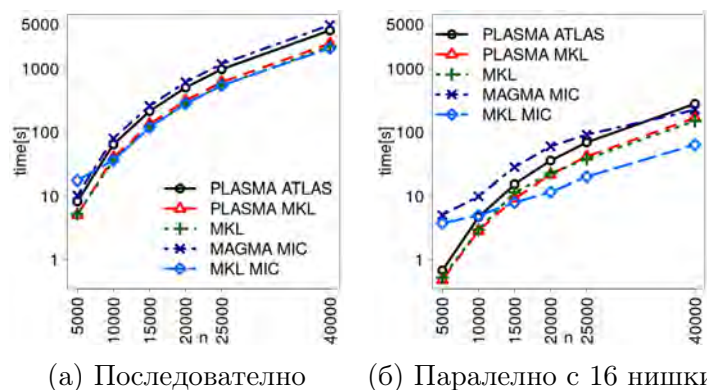
В Таблица 2 са представени резултати от числени експерименти за решаване на системите при $n = 5\,000$ и $n = 40\,000$, варирайки броя на нишките от 1 до 240. Резултатите показват много по-добра производителност на MKL в сравнение с MAGMA MIC. Това може да се дължи на по-добрата комуникация между нишките в MKL. За по-голямата задача постигнатото паралелно ускорение е 32.

На Фигура 2 е сравнена производителността на използваните софтуерни пакети за CPU и MIC архитектурата.

Производителността на PLASMA с MKL за решаване на системите върху CPU е по-добра от тази на MAGMA MIC. Това може да се дължи на по-добра комуникация между нишките.

2.7 Йерархична полусепарабелна компресия

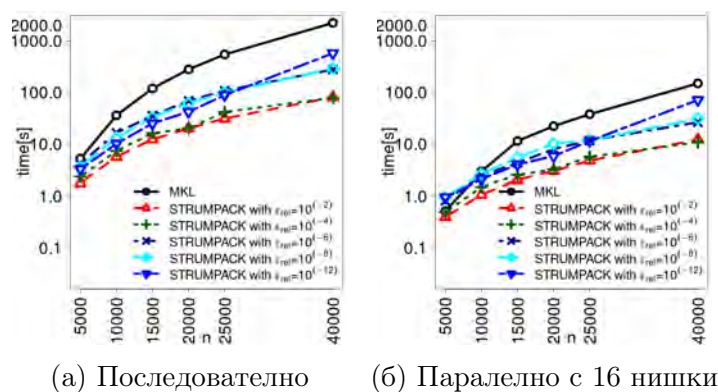
Йерархичната полусепарабелна компресия имплементирана в софтуерния пакет STRUMPACK е приблизителна. Това означава, че компресираната матрица H апроксимира изходящата матрица A . Потребителят задава два прага на грешката – абсолютен ε_{abs} и относителен ε_{rel} [9]. В представените в дисертацията експерименти абсолютният праг е фиксиран на $\varepsilon_{\text{abs}} = 10^{-8}$, като варираме относителния праг $\varepsilon_{\text{rel}} = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ и 10^{-12} .



Фигура 2: Сравнение на производителността на софтуерните библиотеки за CPU и MIC.

2.7.1 Сравнителен анализ на йерархична и LU факторизация върху CPU с обща памет

В този раздел анализираме производителността на метода на полусепарабелна йерархична (HSS) компресия и неговата софтуерна реализация STRUMPACK в сравнение с метода на Гаус и неговата най-добра (на базата на анализа, изложен по-горе) реализация MKL, където алгоритъмът използва блочна LU факторизация. На Фигура 3 са представени последователните (Фигура 3а) и паралелните (Фигура 3б) времена за решаване на системата линейни уравнения. Резултатите потвърждават по-добрата изчислителна сложност на HSS компресията ($O(n^2r)$) в сравнение с LU факторизацията ($O(n^3)$). Ясно се вижда и въздействието на избрания относителен праг на грешката ε_{rel} .



Фигура 3: Производителност на STRUMPACK сравнена с MKL

За последователните експерименти STRUMPACK е много по-ефективен от най-добрия пряк солвър, който използваме – MKL. STRUMPACK показва

по-малки паралелни ускорения, съответно от ~ 2 до ~ 5 . Това се дължи на по-сложната рекурсивна структура на HSS компресията.

2.7.2 Анализ на грешката за HSS-базирания солвър

Нека припомним отново, че HSS компресията е приблизителна, т.е. компресираната матрица H е приближение на матрицата A . Полученото с помощта на HSS компресия решение на системата линейни алгебрични уравнения е апроксимация на точното. За оценка на грешката приемаме за референтно решението по метода на Гаус с пряк солвър на базата на LU факторизация (виж Раздел 2.6). Тук анализираме относителната грешка R_{relative} дефинирана както следва:

$$R_{\text{relative}} = \frac{\|x^{\text{Gauss}} - x^{\text{HSS}}\|_{l_2}}{\|x^{\text{Gauss}}\|_{l_2}} = \frac{\sqrt{\sum_{i=1}^n (x_i^{\text{Gauss}} - x_i^{\text{HSS}})^2}}{\sqrt{\sum_{i=1}^n (x_i^{\text{Gauss}})^2}}, \quad (7)$$

където x^{Gauss} е решението получено с гаусовия солвър от пакета MKL, което се използва като референтно, а x^{HSS} е решението получено при използване на йерархична полусепарабелна компресия.

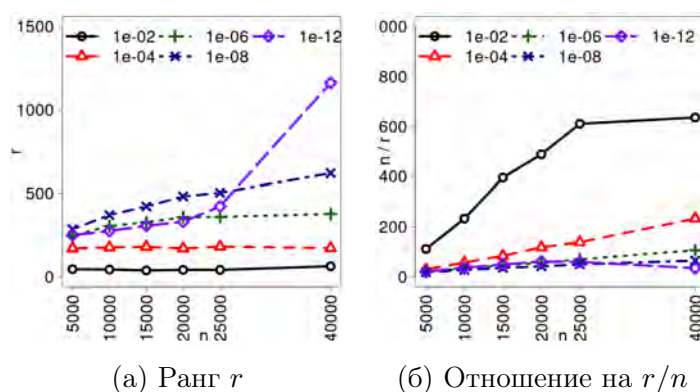
В Таблица 3 са показани относителните грешки R_{relative} , варирайки размерността на задачата $n \in \{5\ 005, 10\ 005, 15\ 005, 20\ 005, 25\ 005, 40\ 005\}$, както и праговете на относителна грешка $\varepsilon_{\text{rel}} \in \{10^{-6}, 10^{-8}, 10^{-12}\}$.

Таблица 3: Относителна грешка R_{relative}

n	ε_{rel}			n	ε_{rel}		
	10^{-6}	10^{-8}	10^{-12}		10^{-6}	10^{-8}	10^{-12}
5005	1.1	0.085	0.00019	15005	0.29	0.23	0.00097
20005	0.28	0.34	0.0038	25005	0.3	1.48	0.013
10005	0.75	0.17	0.00075	40005	0.37	1.59	0.027

Точността и изчислителната ефективност на йерархичния метод зависят от максималния ранг на извъндиагоналните блокове r , който се определя от избраните прагове на грешката и зависи от структурата на изходната матрица A . По-големият ранг r съответства на по-малка относителна грешка R_{relative} , както и на по-дълго време за решаване на системата.

Анализът на представените резултати показва, че за постигане на висока точност на метода може да е необходим много малък праг съответстващ на голям ранг r . В такъв случай методът на HSS компресия може да не бъде достатъчно ефективен.

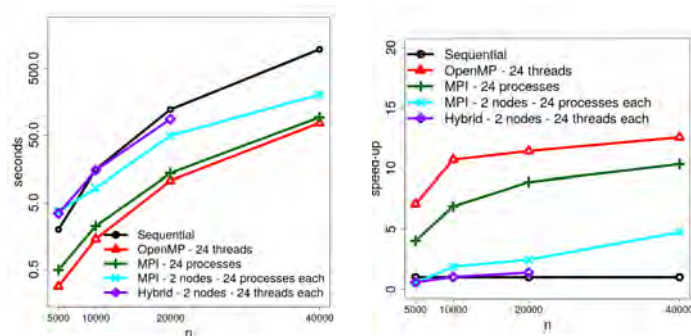
Фигура 4: Максимален извъндиагонален ранг r

2.8 Паралелна скалируемост върху компютърни системи с разпределена памет

Този раздел е посветен на анализ на някои специфични особености и трудности при работа с изчислителни системи с хибридна архитектура. Те имат разпределена памет на нивото на сървърите, от които са изградени и обща памет в рамките на всеки сървър. Представените резултати са публикувани в [24]. Както и в предишния раздел, сравнителният анализ включва паралелните библиотеки MKL и STRUMPACK. За решаване на системата от линейните алгебрични уравнения, получена при дискретизация на задачата за обтичане на профили на Жуковски са използвани един или два сървъра, свързани с етернет. Анализирани са следните варианти на изпълнение на MKL и STRUMPACK: Последователно; Паралелизация с 24 OpenMP нишки; Паралелизация с 24 MPI процеса на 1 сървър; Паралелизация с 48 MPI процеса на 2 сървъра; Хибридна паралелизация с 2 MPI процеса на 2 сървъра, всеки с 24 OpenMP нишки.

2.9 LU факторизация

На Фигура 5 са показани времената за решение на системата от линейни алгебрични уравнения с MKL. Най-добро време се получава с OpenMP при използване на един сървър. Паралелното ускорение на един сървър е най-добро за OpenMP, следвано от MPI (Фигура 5б). Това се обяснява с относително по-бавните етернет комуникации между процесите при използване на повече от един сървър.



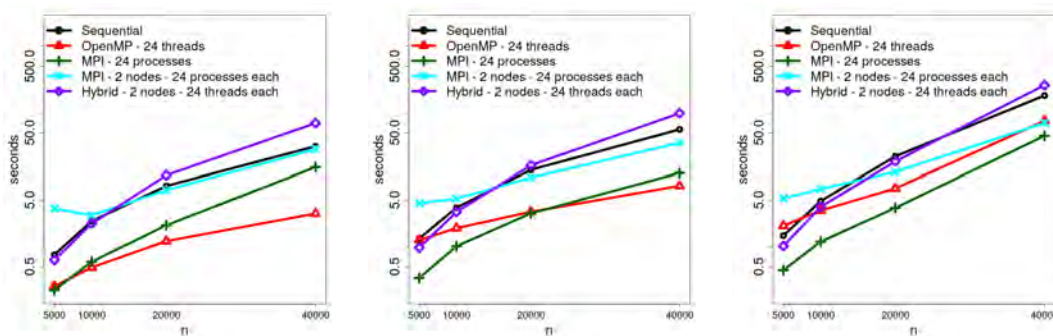
(а) MKL времена (б) MKL паралелни ускорения

Фигура 5: Паралелни времена и ускорения за решаване на системите с MKL.

2.10 HSS компресия

При решаване на системата с използвана HSS компресия се получават по-малки паралелни ускорения отколкото при прекия Гаусов солвър (виж Фигура 6). Това може да се обясни с рекурсивната структура на HSS компресията.

При числените експерименти с най-ниския праг на относителна грешка ($\epsilon_{rel} = 10^{-8}$) се получава най-добро време за изпълнение при използване на MPI върху един сървър. Това най-вероятно се дължи на факта, че OpenMP паралелизацията в STRUMPACK е направена по-късно в разработката на пакета от MPI.



(а) STRUMPACK времена при $\epsilon_{rel} = 10^{-2}$ (б) STRUMPACK времена при $\epsilon_{rel} = 10^{-6}$ (в) STRUMPACK времена при $\epsilon_{rel} = 10^{-8}$

Фигура 6: Времена и паралелни ускорения.

Поведението на паралелното ускорение се променя съществено при използване на два сървъра. Това се дължи на (относително) бавната връзка между тях – 1000 Mb Ethernet. Можем да направим извода, че ефективността мо-

же значително да се подобри при по-бърза комуникационна среда (например InfiniBand), както и при увеличаване на размера на решаваните системи линейни алгебрични уравнения.

2.11 Заключение бележки

Централно място в представените резултати заема изследването на метода на йерархична полусепарабелна компресия (HSS компресия). Експерименталният сравнителен анализ е на базата на неговата реализация в софтуерния пакет STRUMPACK. Той показва по-добро бързодействие от преките Гаусови солвъри, използващи блочна LU факторизация. В същото време получените паралелни ускорения със STRUMPACK са по-малки, което се обуславя от по-сложната йерархична структура на алгоритъма.

Точността и изчислителната ефективност на HSS компресията зависят от праговете на относителна и абсолютна грешка. Това са параметри, които се избират от потребителя. Представеният анализ показва как да получим най-добра ефективност при зададена точност.

Глава 3 Метод на крайните елементи за числено решаване на двумерна стационарна задача за дробна дифузия

Дробните елиптични оператори по пространството на степен $\alpha \in (0, 1)$ описват процеси на *аномална* дифузия. Свързаните с тях гранични задачи са нелокални и, в общия случай, численото решение на такива задачи е изчислително скъп процес. Такъв тип нелокални модели се прилага например в обработката на изображения, финансовата математика, електро-магнитостатиката, перидинамиката, моделирането на течения в порести среди и много други.

Представените числени експерименти са за моделни задачи в квадратна и кръгла област, като авторефератът се ограничава с резултати само за първата. Дробният лапласиан се дефинира с помощта на потенциал на Риц (Riesz). Теоретичната постановка на задачата и разработеният специализиран метод на крайните елементи за нейното числено решаване са представени в статията на Акоста и съавтори [1]. В [2] авторите описват алгоритмичната реализация на метода.

В Глава 2 бяха анализирани няколко софтуерни пакета реализиращи методи от тип гаусова елиминация. В тази глава е използван само най-ефективният от тях: Intel's Math kernel Library (MKL). Анализирана е производителността на алгоритъма базиран на йерархична полусепарабелна компресия (HSS),

реализиран в пакета STRUctured Matrix PACKage (STRUMPACK). Изследвани са няколко метода за пренареждане на неизвестните с цел подобряване ефективността на HSS компресията.

3.1 Постановка на задачата

Дробният лапласиан може да се представи във вида

$$(-\Delta)^\alpha u(x) = C(d, \alpha) \text{P.V.} \int_{\mathbb{R}^n} \frac{u(x) - u(y)}{|x - y|^{d+2\alpha}}, \quad (8)$$

където P.V. означава главна стойност, d е размерността, $\alpha \in (0, 1)$, а $C(d, \alpha)$ е нормализиращата константа

$$C(d, \alpha) = \frac{2^{2\alpha} \alpha \Gamma\left(\alpha + \frac{d}{2}\right)}{\pi^{d/2} \Gamma(1 - \alpha)},$$

където Γ е гамма функцията.

В тази глава е използвана интегралната дефиниция на дробен лапласиан, в съответствие с възприетата постановка на задачата в статията на Акоста [2]. Така разглеждаме следната гранична задача за *дробния* оператор на Лаплас

$$\begin{cases} (-\Delta)^\alpha u(x) = f(x), & x \in \Omega \\ u(x) = 0, & x \in \Omega^c. \end{cases} \quad (9)$$

Тук $\Omega \subset \mathbb{R}^d$ е ограничена отворена област, Ω^c е допълнението на Ω в \mathbb{R}^d и $f(x)$, $x \in \Omega$, е дясна част с достатъчна гладкост.

Вариационната формулировка на (9) се получава, като уравнението се умножи с тестова функция и се интегрира по части. Така за слабото решение получаваме уравнението: търси се $u \in H^\alpha(\Omega)$, такова че

$$\frac{C(d, \alpha)}{2} \langle u, v \rangle_{H^\alpha \mathbb{R}^d} = \int_{\Omega} f v, \quad v \in \tilde{H}^\alpha(\Omega). \quad (10)$$

Скаларното произведение на u и v е дефинирано в Хилбертовото пространство $H^\alpha(\Omega)$ с норма $\|\cdot\|_{H^\alpha(\Omega)} = \|\cdot\|_{L^2(\Omega)} + |\cdot|_{H^\alpha(\Omega)}$. Тук $|\cdot|_{H^\alpha(\Omega)}$ е полунормата на Ароншайн-Слободецкий. $\langle u, v \rangle_{H^\alpha \mathbb{R}^d}$ може да се запише във вида

$$\langle u, v \rangle_{H^\alpha \mathbb{R}^d} = \iint_{\mathbb{R}^d \times \mathbb{R}^d} \frac{(u(x) - u(y))(v(x) - v(y))}{|x - y|^{d+2\alpha}} dx dy.$$

Ще обърнем внимание на факта, че интегрирането е върху цялото пространство \mathbb{R}^d .

Коректността на вариационната постановка на задачата (10), както и съществуването и единствеността на решение в $\tilde{H}^\alpha(\Omega)$ следват от лемата на Лакс-Милграм (Lax-Milgram).

3.2 Постановка на метода на крайните елементи

Нека \mathcal{T} е допустима триангулация на областта Ω съставена от $N_{\mathcal{T}}$ триъгълни крайни елементи. Разглеждаме крайноелементното пространство \mathbb{V}_h от непрекъснати на части линейни функции върху \mathcal{T} . Нека $\{\varphi_1, \dots, \varphi_N\} \subset \mathbb{V}_h$ е лагранжев възлов базис съответстващ на върховете на триъгълниците от $N_{\mathcal{T}}$ означени с x_1, \dots, x_N , които не са на границата $\partial\Omega$. Тогава $\varphi_i(x_j) = \delta_i^j$. Нека $T \in \mathcal{T}$ е даден елемент от триангулацията и нека означим с h_T и ρ_T съответно диаметъра и радиуса на вписаната в T окръжност, като $h = \max_{T \in \mathcal{T}} h_T$. Разглеждаме регулярни по форма триангулации, за които съществува $\tau > 0$ независимо от \mathcal{T} , такова че

$$h_T \leq \tau \rho_T, \quad \forall T \in \mathcal{T}.$$

При тези предположения за всяко $\alpha \in (0, 1)$ дискретният аналог на вариационната задача (10) има вида

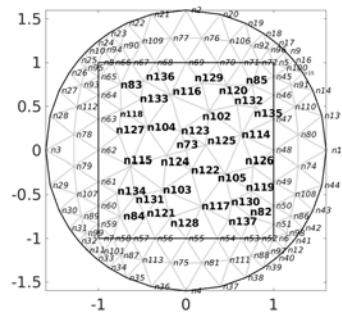
$$\frac{C(d, \alpha)}{2} \langle u_h, v_h \rangle_{H^\alpha(\mathbb{R}^n)} = \int_{\Omega} f v_h, \quad v_h \in \mathbb{V}_h. \quad (11)$$

Тук с $u_h = \sum_j u_j \varphi_j$ е означено численото решение по метода на крайните елементи (МКЕ) на стационарната задача за дробна дифузия (9). Решаването на дискретната вариационна задача се свежда до система от линейни алгебрични уравнения във вида $KU = F$, където $U = (u_j) \in \mathbb{R}^N$ са неизвестните възлови стойности. Матрицата на коравина K е симетрична и положително определена, и следователно системата има единствено решение.

В приложения в [1] вариант на МКЕ, интегрирането се редуцира до кръгова област $B \supset \Omega$, такава че разстоянието между границата $\bar{\Omega}$ и допълнението B^c е достатъчно голямо. Добавя се допълнителната триангулация $\tilde{\mathcal{T}}_A$ върху $B \setminus \Omega$, такава че общата триангулация $\tilde{\mathcal{T}} = \mathcal{T} \cup \tilde{\mathcal{T}}_A$ върху B е допустима. На Фигура 7 е показан пример на такава триангулация за квадратна област Ω . Възлите отбелязани с удебелен шрифт съответстват на неизвестните u_j .

Нека означим с $M_{\tilde{\mathcal{T}}}$ броя на елементите в триангулацията върху кръговата област B . Тогава елементите на матрицата на коравина K могат да се запишат във вида

$$K_{ij} = \frac{C(d, \alpha)}{2} \sum_{\ell=1}^{N_{\tilde{\mathcal{T}}}} \left(\sum_{m=1}^{N_{\tilde{\mathcal{T}}}} I_{\ell, m}^{i, j} + 2J_{\ell}^{i, j} \right), \quad \ell, m \in [1, \dots, N_{\tilde{\mathcal{T}}}], \quad (12)$$



Фигура 7: Допустима триангулация за квадратна област Ω с обвиващ кръг B .

където за въведените интеграли I и J са в сила формулите

$$I_{\ell,m}^{i,j} = \int_{T_\ell} \int_{T_m} \frac{(\varphi(x) - \varphi_i(y))(\varphi_j(x) - \varphi_j(y))}{|x - y|^{2+2\alpha}} dx dy \quad (13)$$

$$J_\ell^{i,j} = \int_{T_\ell} \int_{B^c} \frac{\varphi_i(x)\varphi_j(x)}{|x - y|^{2+2\alpha}} dy dx. \quad (14)$$

3.3 Пренареждане на неизвестните

Както беше отбелязано в Раздел 1.5 за определени класове задачи е възможно неизвестните да бъдат пренаредени така, че да се подобри съществено *структурата* на матрицата на системата. Представеният в тази глава анализ показва, че за матриците породени от задачата за дробна дифузия пренареждането е необходимо.

На Фигура 9а са показани оригиналните номерации на възлите в триангулацията. Те се получават от MatLab функцията `initmesh` при генериране на крайноелементните мрежи. *Структурата* на матрицата съответстваща на това подреждане не е подходяща за йерархичния солвър от пакета STRUMPACK.

3.3.1 Пренареждане по Y координата – „top“

При това пренареждане съответстващите на възлите от триангулацията на Ω неизвестни се сортират по тяхната Y координата. Новата номерация и структурата на получената матрица са визуализирани на Фигура 9б.

3.3.2 Пренареждане по линии – „stripes“

При това пренареждане новата номерация на възлите от мрежата е по хоризонтални линии. Пример за това пренареждане и структурата на получената матрица са показани на Фигура 9в.

3.3.3 Пренареждане по спирала – „snake“

Този алгоритъм пренарежда неизвестните по спирала наподобяваща навита змия. Това пренареждане и структурата на получената матрица са визуализирани на Фигура 9г.

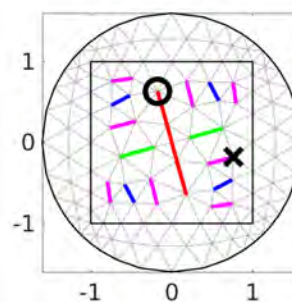
3.3.4 Пренареждане по метода на вложените сечения

Методът на вложените сечения (Nested Dissection) използва подхода „разделяй и владей“ за разделяне на графа представящ структурата на ненулевите елементи на зададена разредена симетрична матрица.

Алгоритъмът включва три стъпки:

1. Конструиране на неориентиран граф, съответстващ на триангулацията в който върхове са възлите от мрежата а ребрата са страните на съответните триъгълници.
2. Рекурсивно разделяне на графа, използвайки разделители (такива малки подмножества от върхове, че когато се премахнат графът се разделя на два подграфа). Тази процедура продължава рекурсивно.
3. Пренареждане на възлите в съответствие с рекурсивната структура: първо по подграфи и след това по разделители.

Разделителите за тестовата задача в квадратна област са показани на Фигура 8. Пренареждането и получената структура на матрицата са визуализирани на Фигура 9г.



3.3.5 Пренареждане по метода на рекурсивна бисекция

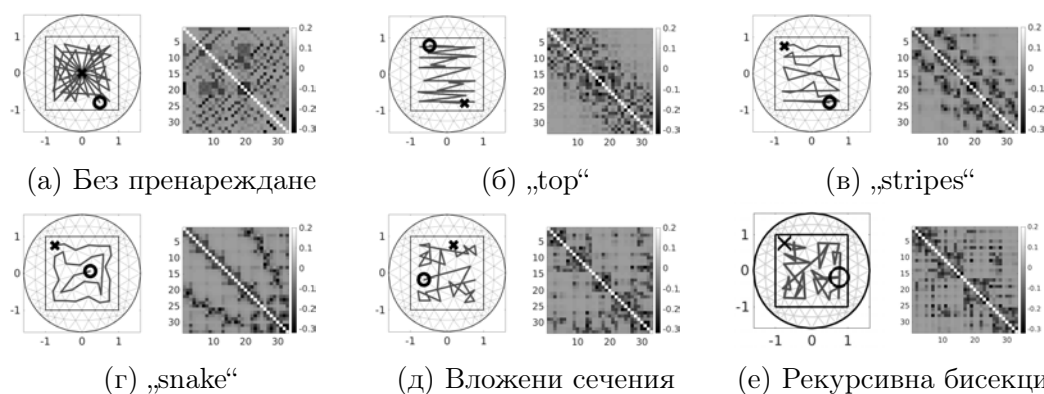
Разглеждаме рекурсивната бисекция като алтернативен подход за разделяне на графа на две части [22]. За разлика от метода на вложените сечения, в този случай графът се разделя чрез премахване на множество от ребра. Процесът продължава рекурсивно, като всеки от двата подграфа се разделя на две по същият начин и така нататък. Частите на всяко ниво на рекурсивното разделяне се номерират последователно.

Фигура 8: Пренареждане с метод на вложените сечения. С „X“ е означен първия връх, а с „O“ последния.

Рекурсивната бисекция се използва за балансиране на натоварването при решаване на задачи върху изчислителни системи с разпределена памет [22].

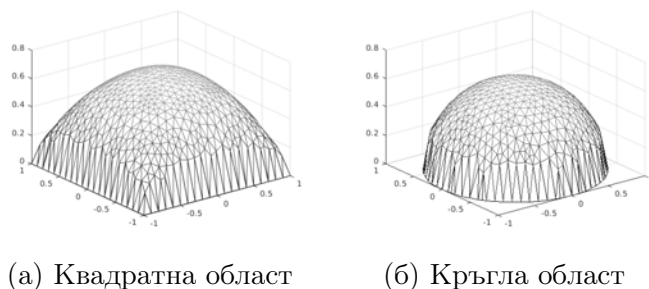
3.4 Анализ на числени експерименти върху компютърни системи с обща памет

Експерименталните резултати анализирани в този раздел са получени върху един сървър от суперкомпютъра AVITONOL. Оригиналната номерация на възлите (неизвестните) се получава в резултат от генерирането на мрежата реализирано в програмата публикувана в статия [2]. С цел подобряване ефективността на йерархичния солвър от пакета STRUMPACK са анализирани няколко подхода за пренареждане на неизвестните.



Фигура 9: Пренареждане на върховете в квадратна област Ω (горе) и структура на съответната матрица K . Тъмносивите линии показват подреждането на възлите от първия (маркиран с „X“) до последния (маркиран с „O“).

Числените експерименти са за задачи с *дробен* лапласиан със степен $\alpha = 0.5$. Визуализация на численото решение на задачата за квадратна и кръгла област е показана на Фигура 10. В автореферата са представени резултати за квадратната област.



Фигура 10: Решение на задачата за *дробна* дифузия за квадратна и кръгла област.

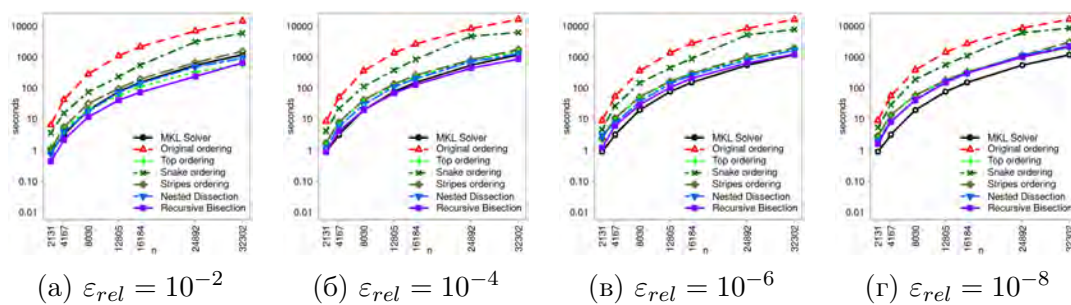
В този раздел анализираме производителността на двата разглеждани метода на базата на тяхната реализация в следните софтуерни пакети за изчислителни системи с обща памет: LU факторизация от MKL и HSS компресия от STRUMPACK.

3.5 Квадратна област

На Фигура 11 са представени последователните времена (при използване на една нишка) за решаване на системата линейни алгебрични уравнения. В пове-

чето случаи йерархичният солвър показва най-добри времена при рекурсивната бисекция, следвана от метод на вложените сечения, „top“ и „stripes“. Експериментите показват, че MKL има по-добра производителност от STRUMPACK за всички стойности на относителният праг с изключение на $\varepsilon_{rel} = 10^{-2}$, където резултатите за рекурсивната бисекция, „stripes“ и „top“ пренарежданията са по-добри.

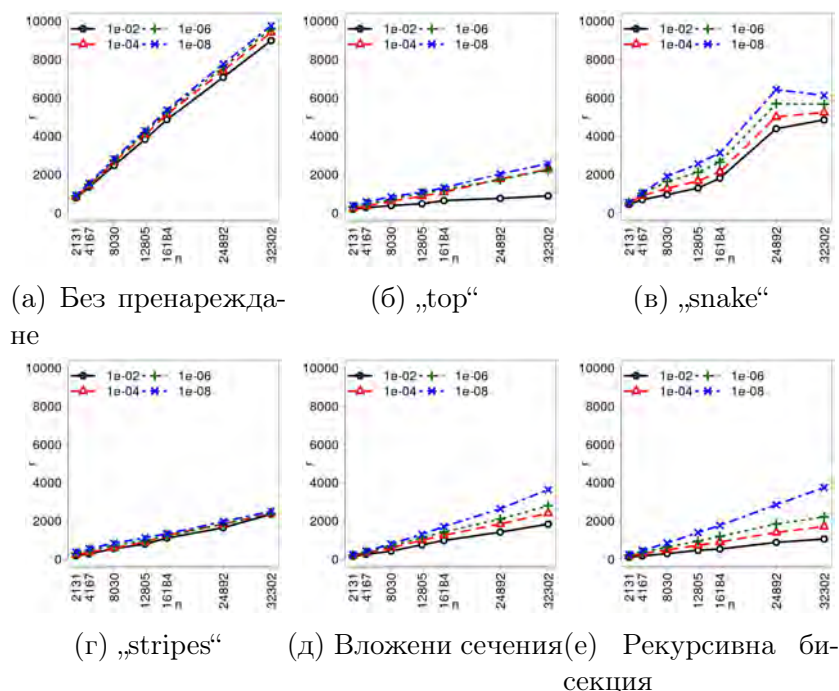
Можем да направим също така извода, че като изключим варианта без пренареждане и пренареждането от тип „snake“, STRUMPACK показва относително близка до MKL производителност.



Фигура 11: Сравнителен анализ на времената за решаване на системата линейни алгебрични уравнения с MKL и STRUMPACK с пренареждания „top“, „snake“, „stripes“, вложени сечения (Nested Dissection) и рекурсивна бисекция (Recursive Bisection) за случая на квадратна област.

3.5.1 Извъндиагонален ранг

На Фигура 12 е представен извъндиагоналният ранг r изчислен в процеса на HSS компресията, като на Фигура 13 са показани отношенията n/r , където n е броят на неизвестните. Стойността на r е мярка за ефективността на компресията за дадената матрица. Колкото по-малък е рангът, толкова по-ефективна е компресията. При оригиналното подреждане стойностите на r са най-големи, като варират между $1/3$ и $1/4$ от броя на неизвестните. Следващ по големина ранг се получава при пренареждането „snake“. Рангът за останалите пренареждания има относително близки стойности, като за повечето от разгледаните примери за рекурсивната бисекция се получава най-висока ефективност на йерархичната полусепарабелна компресия. Ролята на ранга се потвърждава и от анализирания в предходния раздел резултати показани на Фигура 11.



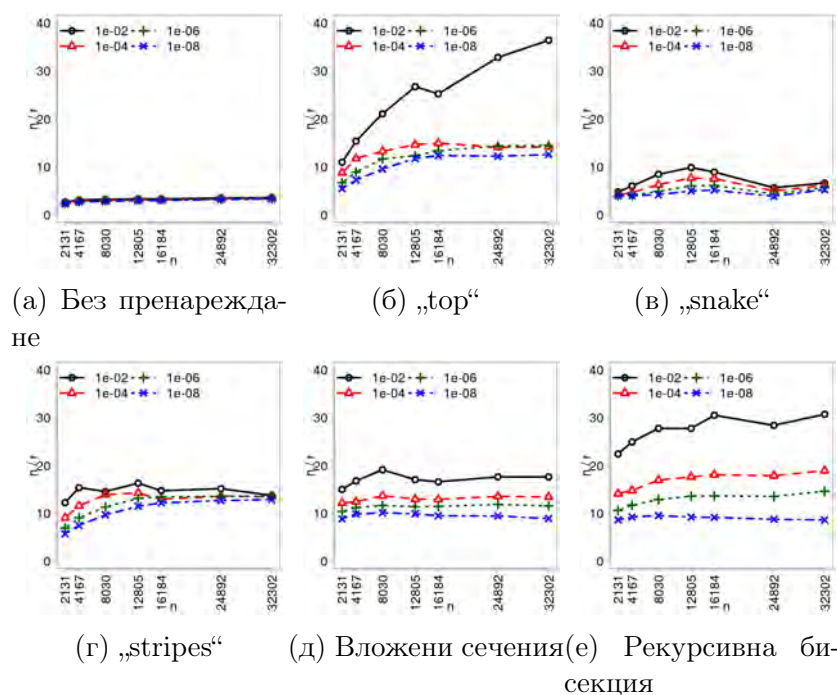
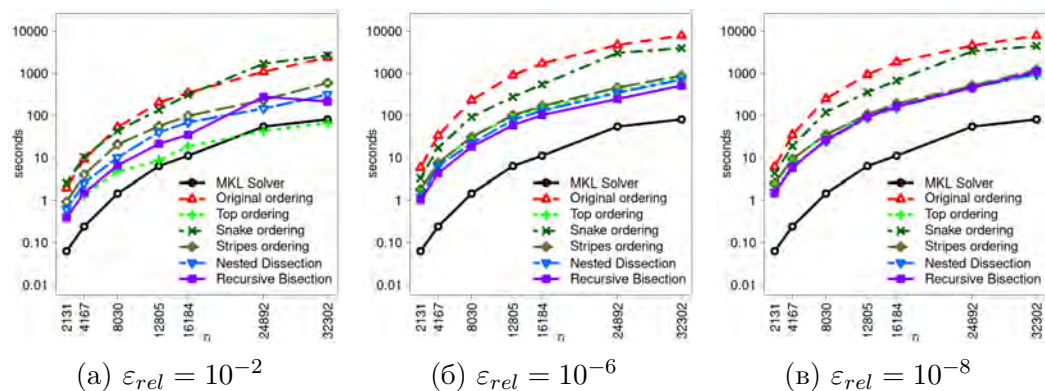
Фигура 12: Максимален извъндиагонален ранг r .

3.5.2 Паралелни времена и ускорения

В този раздел анализираме паралелната ефективност. Проведени са числени експерименти за матрици с нарастваща размерност $n \in [2\ 131, 32\ 302]$. На Фигура 14 са представени резултати получени при използване на 16 нишки. Виждаме, че графиките имат подобно поведение, като за по-големите задачи (по-големите стойности на n) ускоренията стават близки до линейни. Това е в съответствие с теоретичните оценки. Отбелязваме, че за $\varepsilon_{\text{rel}} = 10^{-2}$ и пренареждане „top“ йерархична HSS компресия е с по-добри времена за двете най-големи системи, т.е. при $n = 24\ 892$ и $n = 32\ 302$. В останалите случаи директният гаусов солвър MKL е по-бърз. Това се дължи и на по-сложната рекурсивна *структура* на HSS компресията. Можем да очакваме, че при по-големи стойности на n , STRUMPACK може да покаже по-добри времена от MKL и за по-малки стойности на прага на относителна грешка ε_{rel} .

3.5.3 Анализ на грешката за HSS-базирания солвър

При прилагане на HSS солвъра от пакета STRUMPACK получаваме *приближение* на решението на системата. Това се дължи на факта, че компресираната матрица H е апроксимация на изходната. Както и в предходната глава,

Фигура 13: Отношение на броя на неизвестни n и максималния ранг r .

Фигура 14: Сравнение на паралелните времена за решаване на системата линейни алгебрични уравнения с MKL и STRUMPACK с пренареждания „top“, „snake“, „stripes“, вложени сечения (Nested Dissection) и рекурсивна бисекция (Recursive Bisection) за квадратна област.

ще анализираме относителната грешка на метода (7). В този анализ приемаме решението получено с директния гаусов солвър на MKL за референтно.

В Таблица 4 са представени относителните грешки за оригиналното подреждане и разгледаните пренареждания на неизвестните за квадратна област.

Таблица 4: Относителна грешка за квадратна област.

(а) Без пренареждане пренареждане „top“.

n	Без пренареждане				„top“			
	10^{-2}	$R_{relative}$ 10^{-4}	за r_{tol} 10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ 10^{-4}	за r_{tol} 10^{-6}	10^{-8}
2131	0.0125	0.000124	1.633e-06	1.088e-08	0.122	0.00014	1.469e-06	1.003e-08
4167	0.0244	0.000211	2.195e-06	1.414e-08	0.194	0.000345	1.868e-06	1.95e-08
8030	0.0435	0.000422	7.515e-06	5.049e-08	0.237	0.00681	5.732e-06	3.446e-08
12805	6.74	0.0212	6.258e-06	5.04e-08	0.329	0.00976	4.3e-06	6.554e-08
16184	0.136	0.000818	1.307e-05	1.035e-07	0.335	0.00117	5.96e-06	6.16e-08
24892	0.115	0.000955	1.791e-05	1.659e-07	0.45	0.00192	4.920e-06	9.5e-08
32302	0.145	0.0011	3.409e-05	1.533e-07	0.479	0.00246	9.788e-06	9.09e-08

(б) Пренареждания „snake“ и „stripes“.

n	„snake“				„stripes“			
	10^{-2}	$R_{relative}$ 10^{-4}	за r_{tol} 10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ 10^{-4}	за r_{tol} 10^{-6}	10^{-8}
2131	0.0786	0.000229	1.353e-06	1.144e-08	0.111	0.000324	2.01e-06	1.283e-08
4167	0.172	0.000355	2.212e-06	2.357e-08	0.193	0.000623	2.178e-06	4.491e-08
8030	0.206	0.00107	2.737e-06	5.342e-08	0.287	0.00108	7.943e-06	4.421e-08
12805	0.324	0.00286	6.105e-06	1.995e-07	0.382	0.00117	6.793e-06	7.977e-08
16184	0.397	0.00363	8.919e-06	9.394e-08	0.393	0.00163	5.446e-06	7.988e-08
24892	0.513	0.00753	1.32e-05	1.174e-07	0.478	0.00176	7.613e-06	1.211e-07
32302	0.587	0.00647	3.129e-05	1.774e-07	0.497	0.00231	9.161e-06	1.088e-07

(в) Пренареждания по методите на вложените сечение и рекурсивната бисекция.

n	Вложени сечения				Рекурсивна бисекция			
	$R_{relative}$ 10^{-2}	for STRUMPACK 10^{-4}	with r_{tol} 10^{-6}	10^{-8}	$R_{relative}$ 10^{-2}	for STRUMPACK 10^{-4}	with r_{tol} 10^{-6}	10^{-8}
2131	0.145	0.000403	2.837e-06	3.022e-08	0.0892	0.000338	2.694e-06	2.373e-08
4167	0.247	0.00138	2.866e-06	6.963e-08	0.223	0.000987	3.28e-06	5.245e-08
8030	0.378	0.00221	6.636e-06	7.277e-08	0.373	0.00172	6.402e-06	7.322e-08
12805	0.469	0.00342	8.211e-06	1.159e-07	0.424	0.000929	7.06e-05	1.043e-07
16184	0.499	0.00268	8.668e-06	1.542e-07	0.487	0.00217	8.736e-06	1.745e-07
24892	0.583	0.00318	1.04e-05	1.536e-07	0.536	0.0033	1.738e-05	2.e-07
32302	0.615	0.00539	1.432e-05	2.159e-07	0.612	0.0031	1.419e-05	3.386e-07

За повечето експерименти относителната грешка е близка до зададения праг на относителна грешка ε_{rel} .

Относителната грешка зависи от ефективността на компресията. Това означава, че когато изчисленият ранг r на извъндиагоналните блокове е по-малък изходящата матрица се компресира по-ефективно, което води до по-малко изчислително време, но и до по-висока относителна грешка $R_{relative}$ поради по-голямата компресия.

3.6 Заключениелни бележки

Експерименталният сравнителен анализ се базира на реализацията на HSS компресия и ULV-подобна факторизация в софтуерния пакет STRUMPACK. Анализът показва добро бързодействие на последователния алгоритъм в сравнение с прекия гаусов солвър, използващ блочна LU факторизация. В същото време, получените паралелни производителности и ускорения при използване на пакета STRUMPACK са по-малки, което се обяснява с по-сложната йерархична и рекурсивна структура на компресията.

Точността и изчислителната ефективност на HSS компресията съществено зависят от праговете на относителна грешка ε_{rel} и абсолютна грешка ε_{abs} , както и от наличието на подходяща *структура* на матрицата. При експериментите се наблюдава относителна грешка R_{relative} на численото решение близка до ε_{rel} . За подобряване на *структурата* на матрицата са предложени пет начина за пренареждане на неизвестните, които увеличават съществено ефективността на компресията. Представеният анализ показва, че при повечето експерименти ефективността на HSS компресията е най-добра при пренареждане с рекурсивната бисекция.

Структурата на матрицата получена при дискретизация на дробно дифузионната задача е по-малко подходяща за HSS компресия, в сравнение със задачата разгледана в Глава 2. Това може да се обясни с факта, че *дробният* лапласиан е силно нелокален. Предложените пренареждания на неизвестните значително подобряват ефективността на STRUMPACK. Въпреки това паралелното бързодействие на HSS компресията отстъпва на прекия солвър използващ блочна LU факторизация.

Едно от предимствата на HSS компресия е, че при решаване на серия от системи линейни алгебрични уравнения, при които матрицата не се изменя пониската изчислителна сложност на решаване с факторизирана матрица след HSS компресия и ULV-подобна факторизация $O(nr)$ ще има предимство пред решаването с факторизирана матрица след LU факторизация – $O(n^2)$. Такъв например е случаят на параболична задача, при дискретизацията на която се използва матрица на масата с диагонална концентрация (lumped mass matrix) разгледана в следващата глава.

Глава 4 Метод на крайните елементи за решаване на двумерна параболична задача за дробна дифузия

Водещ интерес за изследванията в тази глава представлява решаването на системи с факторизираната матрица при прилагане на йерархичния метод. Тази стъпка, след HSS компресия и ULV-подобна факторизация, има изчислителната сложност $O(nr)$ [5]. За сравнение при използването на LU факторизация (гаусова елиминация) решаването на системи с факторизираната матрица изисква $O(n^2)$ аритметични операции. При стационарната задача тази стъпка се изпълнява само веднъж и почти не влияе на бързодействието на солвърите. Това се променя, когато численият метод включва решаване на последователност от системи линейни алгебрични уравнения с една и съща матрица. В този случай относителната тежест на времето за решаване на системи с факторизираната матрица се увеличава съществено. Такъв е разглежданият в настоящата глава метод за решаване на параболична задача с *дробна* дифузия.

За генериране на матрицата на масата е разработен алгоритъм и софтуерен модул, който използва информацията за геометрията на триангулацията $\mathcal{T} \in \Omega$. За дискретизация по времето ще използваме неявна диференчна схема на Ойлер с постоянна стъпка и диагонална концентрация на матрицата на масата (lumped mass matrix).

За числените експерименти в настоящата глава използваме аналог на тестовия пример от статията на Вабишчевич [26]. Това дава възможност да направим сравнение на числените резултати, съответстващи на двете различни дефиниции на дробния лапласиан.

4.1 Постановка на задачата

Използваме интегралното представяне (9) на дробният лапласиан. Разглеждаме следната параболична задача за неизвестната функция $u(x, t)$, $(x, t) \in \Omega \times [0, T]$

$$\left\{ \begin{array}{l} \frac{\partial u(x, t)}{\partial t} + (-\Delta)^\alpha u(x, t) = f(x, t), \quad x \in \Omega, \quad t \in [0, T], \\ u(x, t) = 0, \quad x \in \Omega^c, \quad t \in [0, T], \\ u(x, 0) = u^0(x), \quad x \in \Omega. \end{array} \right.$$

Тук $[0, T]$ е времевият интервал. Хомогенните гранични условия на Дирихле се налагат както при стационарната задача разглеждана в предходната глава.

Използваме същата допустима триангулация $\mathcal{T} \in \Omega$ като в предходната глава. Така получаваме задачата на Коши

$$M_L \frac{d\mathbf{u}}{dt} + K\mathbf{u} = M_L \mathbf{f}, \quad 0 < t \leq T, \quad \mathbf{u}(0) = \mathbf{u}^0,$$

за неизвестните функции $\mathbf{u} = (u_j(t)) \in \mathbb{R}^N$, $t \in [0, T]$ и дясна част $\mathbf{f} = (f_j(t)) \in \mathbb{R}^N$. Тук $K = K_{ij} \in \mathbb{R}^{N \times N}$ е матрицата на коравина, съответстваща на *дробния* лапласиан. Тя има вида дефиниран в уравнения (12) и (14). С $M_L = \text{diag}(m_L^i) \in \mathbb{R}^N$ е означена матрицата на масата с диагонална концентрация (lumped mass matrix), където m_L^i е концентрираната маса във възела x_i . Алгоритъмът и програмният модул за изчисляване на матрицата M_L са представени в Приложение Б.

За дискретизация по времето използваме неявния метод на Ойлер, който в общия случай има вида

$$M_L \frac{\mathbf{u}^{j+1} - \mathbf{u}^j}{\tau_j} + K\mathbf{u}^{j+1} = M_L \frac{\mathbf{f}^{j+1} + \mathbf{f}^j}{2}, \quad j = 0, \dots, m-1, \quad (15)$$

където m е броят на стъпките по времето,

$$\sum_{j=0}^{m-1} \tau_j = T,$$

$t_0 = 0$, $t_{j+1} = t_j + \tau_j$ и $\mathbf{u}^j = \mathbf{u}(t_j)$, $\mathbf{f}^j = \mathbf{f}(t_j)$.

В настоящата дисертация ще се ограничим до случая на постоянна стъпка по времето $\tau_j = \tau$. При това условие реализацията на всяка стъпка по времето на (15) се свежда до решаване на системата линейни алгебрични уравнения

$$\tilde{K}\mathbf{u}^{j+1} = \tilde{\mathbf{f}}^j, \quad (16)$$

където

$$\tilde{K} = \frac{M_L}{\tau} + K, \quad \tilde{\mathbf{f}}^j = M_L \left(\frac{\mathbf{f}^{j+1} + \mathbf{f}^j}{2} + \frac{\mathbf{u}^j}{\tau} \right).$$

Матрицата на коравина и матрицата на масата са симетрични и положително определени, и следователно (16) има единствено решение. Матрицата \tilde{K} не зависи от j , т.е. тя е една и съща за всички стъпки по времето. Това означава, че при реализацията на метода на Ойлер факторизацията се изпълнява един път, след което решаваме m системи с факторизираната матрица.

Изчислителната сложност на LU факторизацията е $O(n^3)$, като след това реализацията m на брой стъпки по времето изисква още $O(n^2m)$ аритметични операции.

При йерархичния метод – HSS компресията и ULV-подобната факторизация имат обща изчислителна сложност $O(n^2r)$, като след това за решаване на системите с факторизираната матрица са необходими още $O(nrm)$ аритметични операции. Така, определящ за ефективността на йерархичния метод е извъндиагоналният ранг r .

В Глава 3 бяха предложени и анализирани няколко метода за пренареждане на неизвестните за стационарната задача с дробна дифузия. Така тук ще се ограничим до анализ на числени резултати при най-ефективното пренареждане – рекурсивната бисекция.

За числените експерименти ще използваме аналог на параболичната задача от статията на Вабишчевич [26], в която се използва спектралната дефиниция на *дробния* лапласиан. Задачата се решава за $(x, t) \in \Omega \times [0, T] = (-1, 1)^2 \times (0, 0.1)$. Решението се определя от независеща от времето дясна част

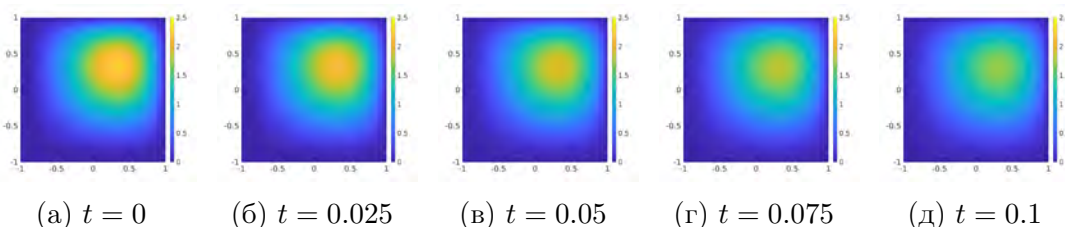
$$f(x) = \frac{(x_1 + 1)(x_2 + 1)}{4},$$

и начално условие

$$u^0(x) = 100 \left(\frac{x_1 + 1}{2} \right)^2 \left(1 - \frac{x_1 + 1}{2} \right) \left(\frac{x_2 + 1}{2} \right)^2 \left(1 - \frac{x_2 + 1}{2} \right)$$

За дискретизация по пространството са използвани триангулациите от Глава 3, при стъпка по времето $\tau = T/m$, $m = 256$.

На Фигура 15 са показани началното условие $u^0(x)$ и числените решения при $\alpha = 0.5$ за $t \in \{0.025, 0.05, 0.075, 0.1\}$. Получените резултати са качествено подобни на резултатите, представени в [26].



Фигура 15: Числени решения на моделната параболична задача с дробна дифузия със степен $\alpha = 0.5$: МКЕ по пространството и неявен метод на Ойлер по времето.

4.2 Анализ на числени експерименти върху компютърни системи с обща памет

Анализираните в тази глава числени резултати са получени върху компютърни системи с обща памет. Както и в предходните глави, този тип експерименти

са проведени на суперкомпютър AVITONOL.

За генериране на матрицата на коравина K и дясната част \mathbf{f} е използвана програма на MatLab, която е публикувана в [2]. За пренареждане на неизвестните по метода на рекурсивната бисекция е използван алгоритъмът описан в Раздел 3.3.5 и разработеният код, представен в Приложение А.4. За изчисляване на матрицата на масата с диагонална концентрация M_L е реализиран алгоритъмът представен в Приложение Б.

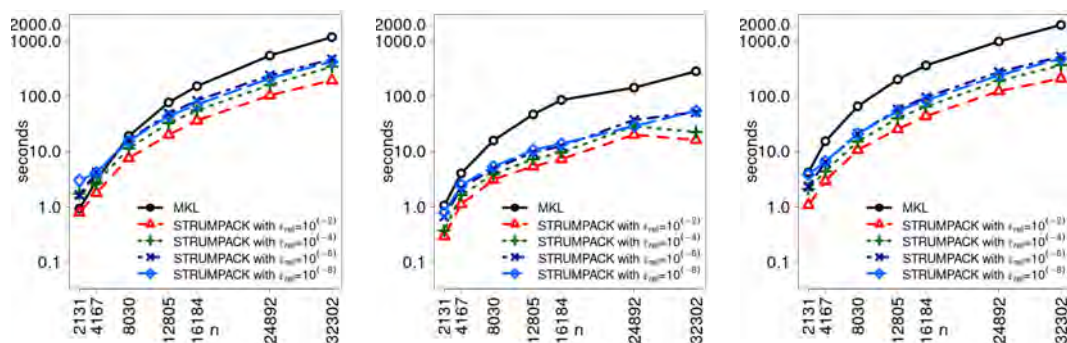
4.3 Последователни и паралелни експерименти.

На Фигура 16 са представени: времената за изпълнение на йерархичната полусепарабелна компресия и ULV-подобна факторизация при използване на софтуерния пакет STRUMPACK и на LU факторизацията при използване на MKL – (Фигури 16а и 16г); общото време за решаване на системите с факторизираните матрици на всяка от стъпките по времето в метода на Ойлер – (Фигури 16б и 16д); и цялото време за решаване на задачата – (Фигури 16в и 16е). В почти всички случаи резултатите от числените експерименти показват по-добрата ефективност на йерархичния метод, като тенденцията е тя да се повишава с увеличаване на броя на неизвестните n . За най-голямата система ($n = 32\ 302$) времената за решаване на цялата задача с прилагане на HSS компресия от пакета STRUMPACK са между ~ 2.5 и ~ 5 пъти по-добри отколкото при използване на LU факторизация от пакета MKL. Тези резултати потвърждават теоретичните очаквания за по-силно подобряване на ефективността на йерархичния солвър при параболичната задача, в сравнение със стационарната задача, разгледана в предходната глава.

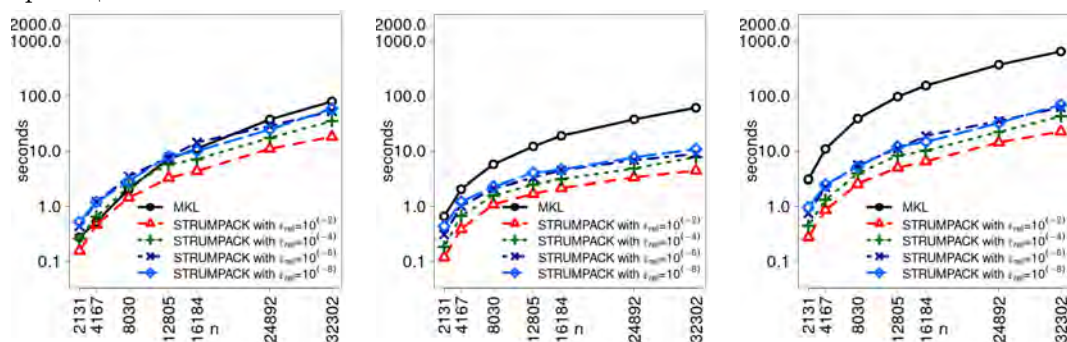
На Фигура 17 са показани паралелните ускорения при решаване на параболичната задача с прилагане на солвъра от пакета STRUMPACK, използващ HSS компресия. При паралелните експерименти с 16 нишки получаваме ускорения от ~ 3 (за $n = 2\ 131$) до ~ 8 (за $n = 32\ 302$). По-ниското паралелно ускорение на йерархичния метод може да се обясни с по-сложната йерархична и рекурсивна структура на алгоритъма за компресия. Паралелната реализация на решаването на системи с HSS факторизираната матрица също има по-сложна (и по-малко балансирана) структура от блочната LU факторизация.

4.4 Извъндиагонален ранг

Изчисленият в процеса на HSS компресията максимален извъндиагонален ранг r е представен на Фигура 18а, като на Фигура 18б е показано отношението n/r . Рангът е мярка за ефективността на компресията, като също така е определящ в оценките за изчислителната сложност. За разглежданата задача

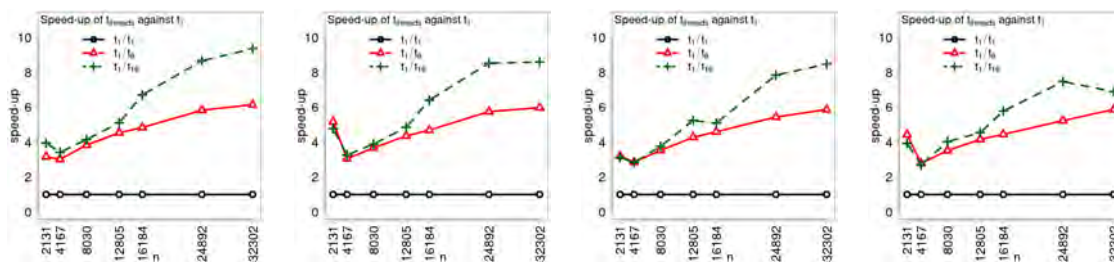


(а) Последователни времена: компресия и факторизация (б) Последователни времена: стъпки по времето (в) Последователни времена за цялата задача



(г) Паралелни времена при 16 нишки: компресия и факторизация (д) Паралелни времена при 16 нишки: стъпки по времето (е) Паралелни времена при 16 нишки за цялата задача

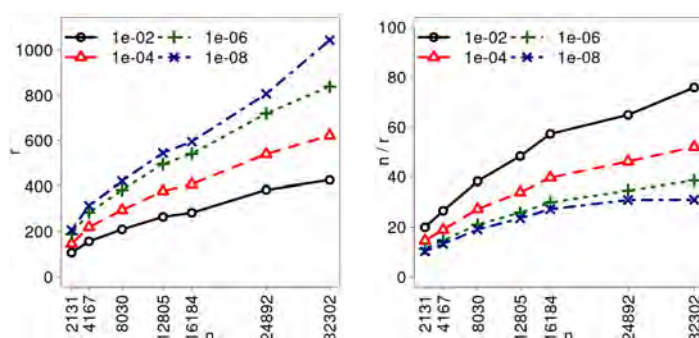
Фигура 16: Сравнение между времената за решаване на параболната задача (16) при използване на MKL и STRUMPACK при праг на относителната грешка $\epsilon_{rel} \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$.



(а) $\epsilon_{rel} = 10^{-2}$ (б) $\epsilon_{rel} = 10^{-4}$ (в) $\epsilon_{rel} = 10^{-6}$ (г) $\epsilon_{rel} = 10^{-8}$

Фигура 17: Паралелни ускорения при решаване на параболната задача с прилагане на HSS базирания солвър за $m = 256$ стъпки по времето.

r има значително по-малки стойности от n . Компресията е най-силна, т.е. отношението r/n е най-малко, при най-голяма стойност на прага на относителна грешка. Така, при $\varepsilon_{\text{rel}} = 10^{-2}$ рангът r е между ~ 20 и ~ 80 пъти по-малък от n , докато при най-финия праг $\varepsilon_{\text{rel}} = 10^{-8}$ това отношение е между ~ 10 и ~ 30 . Този анализ показва, че пренаредената с рекурсивна бисекция матрица \tilde{K} има подходяща структура за прилагане на HSS компресия. Това се потвърждава от числените експерименти, показващи предимство на йерархичния метод в сравнение с гаусовата елиминация (блочната LU факторизация). Тук е важно да припомним, че компресията е *приблизителна*. Така, по-високата ефективност на компресията, която получаваме при по-големи стойности на прага ε_{rel} е за сметка на по-малка точност на решението.



(а) Максимален извъндиагонален ранг r (б) Отношение на r и броя на неизвестните n

Фигура 18: Визуализация на максималния извъндиагонален ранг r и отношение n/r .

4.5 Анализ на грешката за HSS-базирания солвър

Матрицата H получена след HSS компресия е приближение на \tilde{K} . Както при стационарната задача разгледана в Глава 3, ще анализираме относителната грешка R_{relative} (7).

Получените относителни грешки R_{relative} за избраните 4 стойности на времето t са представени в Таблица 5. Както и при стационарната задача, относителната грешка е близка по стойност до зададения относителен праг ε_{rel} . Представените числени резултати показват също така, че относителната грешка не нараства съществено с увеличаване на времевия интервал. Това потвърждава устойчивостта на неявния метод на Ойлер.

Таблица 5: Относителна грешка на HSS базираният солвър.

(а) $t = 0.025$ и $t = 0.05$

n	Относителна грешка в $t = 0.025$				Относителна грешка в $t = 0.05$			
	10^{-2}	$R_{relative}$ за r_{tol} 10^{-4}	10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ за r_{tol} 10^{-4}	10^{-6}	10^{-8}
2131	0.00385	$7.52e-05$	$2.78e-07$	$5.53e-08$	0.00659	0.000124	$4.62e-07$	$9.72e-08$
4167	0.006	0.00013	$6.4e-07$	$8.62e-08$	0.01	0.000225	$1.03e-06$	$1.36e-07$
8030	0.0083	0.00023	$1.28e-06$	$2.17e-07$	0.0146	0.000375	$2.e-06$	$3.49e-07$
12805	0.0106	0.00028	$1.68e-06$	$4.77e-07$	0.0178	0.000484	$2.49e-06$	$7.56e-07$
16184	0.0126	0.0003	$1.75e-06$	$5.16e-07$	0.0227	0.00052	$2.77e-06$	$7.88e-07$
24892	0.0192	0.000393	$2.5e-06$	$9.69e-07$	0.0349	0.000619	$3.97e-06$	$1.49e-06$
32302	0.0234	0.000345	$2.48e-06$	$1.18e-06$	0.0437	0.000537	$3.97e-06$	$1.76e-06$

(б) $t = 0.075$ и $t = 0.1$

n	Относителна грешка в $t = 0.075$				Относителна грешка в $t = 0.1$			
	10^{-2}	$R_{relative}$ за r_{tol} 10^{-4}	10^{-6}	10^{-8}	10^{-2}	$R_{relative}$ за r_{tol} 10^{-4}	10^{-6}	10^{-8}
2131	0.0088	0.000159	$5.98e-07$	$1.32e-07$	0.0108	0.000188	$7.07e-07$	$1.62e-07$
4167	0.0135	0.000298	$1.31e-06$	$1.707e-07$	0.0166	0.000361	$1.55e-06$	$1.98e-07$
8030	0.0206	0.000498	$2.52e-06$	$4.55e-07$	0.0264	0.000606	$2.93e-06$	$5.49e-07$
12805	0.0242	0.000653	$3.09e-06$	$9.74e-07$	0.0301	0.0008	$3.6e-06$	$1.16e-06$
16184	0.0324	0.000706	$3.6e-06$	$1.01e-06$	0.0419	0.000875	$4.34e-06$	$1.2e-06$
24892	0.0499	0.000807	$5.19e-06$	$1.92e-06$	0.0646	0.000973	$6.26e-06$	$2.3e-06$
32302	0.0636	0.000693	$5.25e-06$	$2.21e-06$	0.0832	0.000826	$6.41e-06$	$2.58e-06$

4.6 Заключение и бележки

Водеща тема в представените резултати е анализът на изчислителната ефективност на метода базиран на йерархична полусепарабелна компресия и ULV-подобна факторизация и неговата паралелна реализация в софтуерния пакет STRUMPACK. Специфика на приложения невявния метод на Ойлер с постоянна стъпка τ е, че численото решаване на параболичната задача се свежда до $m = T/\tau$ системи линейни алгебрични уравнения с една и съща матрица \tilde{K} и променящи се на всяка стъпка по времето десни части. Това означава, че матрицата \tilde{K} се факторизира еднократно и акцентът пада върху решаването на m системи с известна факторизирана матрица. Ще припомним, че тази стъпка има изчислителна сложност $O(n^2)$ за гаусовия солвър и $O(nr)$ за йерархичния солвър. Представеният анализ показва, че за разглежданата задача рангът r е съществено по-малък от броя на неизвестните n , което обуславя преимуществото на йерархичния метод. В резултат на това, както последователните, така и паралелните времена за решаване на параболичната задача с използване на солвъра от софтуерния пакет STRUMPACK са съществено по-добри в сравнение с времената при използване на блочна LU факторизация от пакета MKL.

Относителната грешка $R_{relative}$ е близка до зададения относителен праг,

като нараства устойчиво с развитие на процеса по времето. Това потвърждава, че йерархичният метод осигурява добра точност на решението при подходящо избран праг ε_{rel} . Представените резултати показват съществено преимущество на алгоритъма използващ йерархична полусепарабелна компресия и неговата реализация в паралелния пакет STRUMPACK.

Заклучение

В дисертацията е анализирана изчислителната ефективност на блочни числени методи и алгоритми за решаване на системи линейни алгебрични уравнения с плътни матрици. Мотивация на това изследване са приложения свързани с числено решаване на елиптични и параболични частни диференциални уравнения. Две такива задачи са използвани в представения сравнителен анализ: а) гранична задача описваща обтичане на крилни профили на Жуковски, дискретизирана с метод на граничните елементи; б) аномална дифузия в ограничена област моделирана с *дробен* лапласиан, където за дискретизация е приложен метод на крайните елементи. И в двата случая непрекъснатите задачи се свеждат до системи линейни алгебрични уравнения с плътни матрици. Показано е, че *структурата* на тези матрици е подходяща за прилагане на йерархичен метод използващ HSS компресия.

Важна част от Глава 2 е сравнителният анализ на изчислителната ефективност на софтуерни пакета имплементиращи блочна LU факторизация, като вариант на гаусова елиминация. Общият извод е, че софтуерният пакет MKL има по-добро бързодействие от анализирания алтернативни паралелни реализации на блочна LU факторизация.

Основен акцент в дисертацията е анализът на възможността за подобряване на изчислителната ефективност на решаването на системи линейни алгебрични уравнения с плътни матрици с помощта на блочен йерархичен метод използващ HSS компресия. Този метод е реализиран в софтуерния пакет със свободен достъп STRUMPACK. В Глави 2 и 3 е анализирана производителността на йерархичния алгоритъм за системи линейни алгебрични уравнения, получени при прилагане съответно на метод на граничните елементи и метод на крайните елементи за разглежданите елиптични гранични задачи. Анализът показва, че тези плътни матрици имат подходяща *структура* за прилагане на йерархичния метод. Това означава, че в процеса на HSS се получават извъндиагонални блокове с нисък ранг.

Последователните експерименти потвърждават оценките на изчислителната сложност на анализирания блочни методи. И за двете задачи йерархичният солвър има по-добро бързодействие от гаусовия солвър от пакета MKL – най-ефективният от анализирания в настоящата работа софтуерни средства

използващи LU факторизация.

При прилагане на йерархичния метод се намира приближено решение на системата, като неговата точност зависи от това каква е точността на HSS компресията. На базата на проведените числени експерименти е анализирана относителната грешка R_{relative} , като решението получено с прекия гаусов солвър се приема за референтно. За задачата с дробна дифузия се наблюдават относителни грешки близки до зададения праг ε_{rel} . Това може да бъде прието като добра характеристика на алгоритъма за HSS компресия. За задачата за обтичане на крилни профили на Жуковски относителните грешки за съответните прагове ε_{rel} са по-големи, което обаче се компенсира от по-доброто бързодействие.

Известно е, че качеството на йерархичната полусепарабелна компресия силно зависи от *структурата* на плътната матрица. При двумерната дробно дифузионна задача *структурата* на получената плътна матрица не е подходяща за HSS компресия. За нейното подобряване са предложени и пет метода за пренареждане. Представеният анализ показва преимущества на методите на вложените сечения и на рекурсивната бисекция.

В Глава 4 е изследвана изчислителната ефективност и точност на йерархичния солвър базиран на HSS компресия за параболична задача с *дробна* дифузия по пространството. За дискретизация по времето е използвана неявна диференчна схема на Ойлер с постоянна стъпка. При тази постановка на задачата намирането на численото решение се свежда до решаване на последователност от системи линейни алгебрични уравнения с една и съща матрица. Така на всяка стъпка по времето се решава система с матрицата на прехода, която е факторизирана еднократно. Решаването на такива системи с помощта на HSS компресия има по-ниска изчислителна сложност – $O(nr)$, в сравнение с използваната в метода на Гаус LU факторизация – $O(n^2)$. За разгледаната параболична задача времената на йерархичния солвър са по-добри, както при последователните, така и при паралелните експерименти. В същото време, благодарение на безусловната устойчивост на неявния метод на Ойлер, относителната грешка на решението се запазва близка до зададения относителен праг ε_{rel} .

Списък на публикациите по дисертацията

Основна част от представените в дисертацията резултати са публикувани в 5 излезли от печат и 2 приети за печат статии, както следва:

- D. Slavchev and S. Margenov. *Performance Analysis of Intel Xeon Phi MICs and Intel Xeon CPUs for Solving Dense Systems of Linear Algebraic Equations: Case Study of Boundary Element Method for Flow Around*

Airfoils, pages 369–381. Springer International Publishing, Cham, 2019. ISBN 978-3-319-97277-0. doi: 10.1007/978-3-319-97277-0_30

- D. Slavchev and S. Margenov. *Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs*. In G. Nikolov, N. Kolkovska, and K. Georgiev, editors, Numerical Methods and Applications, pages 466–473, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10692-8
- D. Slavchev and S. Margenov. *Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for BEM discretization of flow around airfoils*. Springer International Publishing, Cham. in press
- D. Slavchev. *On the impact of reordering in a hierarchical semi-separable compression solver for fractional diffusion problems*. In I. Lirkov and S. Margenov, editors, Large-Scale Scientific Computing, pages 373–381, Cham, 2020. Springer International Publishing. ISBN 978-3-030-41032-2
- D. Slavchev. *Performance Analysis of Hierarchical Semi-separable Compression Solver for Fractional Diffusion Problems*. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, Advanced Computing in Industrial Mathematics, pages 333–344, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5
- D. Slavchev, S. Margenov, and I. G. Georgiev. *On the application of recursive bisection and nested dissection reorderings for solving fractional diffusion problems using HSS compression*. AIP Conference Proceedings, 2302 (1):120008, 2020. doi: 10.1063/5.0034506

Апробация на резултатите

Резултатите представени в дисертацията са докладвани в редица международни конференции и уъркшоупи.

Международни конференции:

- Large-Scale Scientific Computations (LSSC), Созопол, 2017, 2019, 2021
- Annual Meeting of the Bulgarian Section of SIAM (BGSIAM), София, 2017, 2018, 2019
- Numerical Methods for Scientific Computations and Advanced Applications (NMSCAA), Хисаря, 2018

- Numerical Methods and Applications (NM&A), Боровец, 2018
- Twelfth On-Line Conference of the Euro-American Consortium for Promoting the Application of Mathematics in Technical and Natural Sciences (AMiTaNS), Албена, 2020

Уъркошопи:

- Две години Авитохол: Иновативни Суперкомпютърни Приложения, Панагюрище, 2017
- Numerical Solution of Fractional Differential Equations and Applications (NSFDE&A), Созопол, 2020

Основни научни и научно-приложни приноси

1. Изследвана е производителността на следните софтуерни пакети за решаване на линейни системи с плътни матрици, с помощта на блочна LU факторизация: за процесори с общо предназначение (CPU) – пакетът на Intel Math Kernel Library (MKL) и библиотеката със свободен достъп Parallel Linear Algebra Software for Multicore Architectures (PLASMA); за ускорители с архитектурата Many Integrated Core (MIC) на Intel – MKL и пакетът със свободен достъп Matrix Algebra on GPU and Multicore Architectures (MAGMA). Резултатите от числените експерименти за системи получени при дискретизация с метод на граничните елементи за гранична задача за ламинарен поток около крилни профили на Жуковски са в съответствие с асимптотичните оценки на изчислителната сложност. Сравнителният анализ показва по-добро бързодействие и много добра паралелна скалируемост на пакета MKL.
2. Изследвана е изчислителната сложност, паралелната ефективност и относителната грешка на метод на йерархична полусепарабелна компресия (HSS). Числените експерименти са проведени с пакета със свободен достъп STRUctured Matrices PACKage (STRUMPACK), в който е реализиран паралелен солвър на базата HSS компресия и ULV-подобна факторизация. Сравнителният анализ включва два типа плътни матрици, които са получени при дискретизация с: а) метод на граничните елементи на гранична задача за ламинарен поток около крилни профили на Жуковски; б) метод на крайните елементи за двумерна дробно дифузионна гранична задача. Направен е сравнителен анализ на солвърите от пакетите MKL и STRUMPACK. Получена е характеристика в зависимост от прага на относителна грешка при HSS компресията на случаите, в които йерархичният метод има по-добро бързодействие.

3. Показано е, че за задачата за обтичане на профили на Жуковски при дискретизация по метода на граничните елементи, последователната номерация на възлите по границата на профилите води до матрица с подходяща за HSS компресия *структура*. Това не е така за дробно дифузионната гранична задача, дискретизирана с метод на крайните елементи. С цел подобряване на ефективността на йерархичната полусепарабелна компресия са предложени и изследвани пет метода за пренареждане на неизвестните. За три от тях са разработени собствени алгоритми и програмни реализации. Сравнителният анализ показва съществено подобрене на резултатите при прилагане методите на вложените сечения и на рекурсивната бисекция.
4. Разработен е метод, алгоритъм и програмна реализация за числено решаване на параболично уравнение с дробно дифузионен оператор по пространството. За дискретизация по времето е приложен неявен метод на Ойлер с постоянна стъпка по времето и диагонална концентрация на масата. Доказано е, че за тази нестационарна задача, изчислителната сложност на отделните части на алгоритъма създава условия за предимство на йерархичния метод на базата на HSS компресия. Това е потвърдено от проведените числени експерименти. Така за всички размерности на дискретната задача по пространството, както и при всички варианти на праг на относителна грешка, вариантът на програмата използваща солвъра от пакета STRUMPACK има по-добро бързодействие от този използващ MKL.

Литература

- [1] G. Acosta and J.P. Borthagaray. A Fractional Laplace Equation: Regularity of Solutions and Finite Element Approximations. In *SIAM Journal on Numerical Analysis*, volume 55, pages 472–495, 2017. doi: 10.1137/15M1033952.
- [2] G. Acosta, F.M. Bersetche, and J.P. Borthagaray. A short FE implementation for a 2d homogeneous Dirichlet problem of a fractional Laplacian. In *Computers & Mathematics with Applications*, volume 74, pages 784 – 816, 2017. doi: <https://doi.org/10.1016/j.camwa.2017.05.026>.
- [3] M. Benzi, D. Bini, D. Kressner, H. Munthe-Kaas, and C. Van Loan. *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications*. Springer, Cham, Cetraro, Italy, 2015. ISBN 978-3-319-49887-4. doi: 10.1007/978-3-319-49887-4.

- [4] K. Binder, Ch. Bennemann, J. Baschnagel, and W. Paul. Anomalous diffusion of polymers in supercooled melts near the glass transition. In A. Pełkalski and K. Sznajd-Weron, editors, *Anomalous Diffusion From Basics to Applications*, pages 124–139, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49259-7.
- [5] S. Chandrasekaran, M. Gu, and T. Pals. A fast *ulv* decomposition solver for hierarchically semiseparable representations. In *SIAM J. Matrix Anal. Appl.*, volume 28, page 603–622, USA, August 2006. Society for Industrial and Applied Mathematics. doi: 10.1137/S0895479803436652.
- [6] S. Chaturapruek, J. Breslau, D. Yazdi, T. Kolokolnikov, and S.G. McCalla. Crime modeling with Lévy flights. In *SIAM Journal on Applied Mathematics*, volume 73, pages 1703–1720. Society for Industrial and Applied Mathematics, 2021/12/06/ 2013. URL <http://www.jstor.org/stable/24510700>.
- [7] J. Dongarra, M. Abalenkovs, A. Abdelfattah, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, and A. YarKhan. Parallel programming models for dense linear algebra on heterogeneous systems. In *Supercomputing Frontiers and Innovations*, volume 2, 2016. URL <http://superfri.org/superfri/article/view/90>.
- [8] P Ghysels, X.S. Li, F.-H. Rouet, S. Williams, and A. Napov. An Efficient Multicore Implementation of a Novel HSS-Structured Multifrontal Solver Using Randomized Sampling. In *SIAM Journal on Scientific Computing*, volume 38, pages S358–S384, 2016. doi: 10.1137/15M1010117.
- [9] C. Gorman, G. Chavez, P. Ghysels, T. Mary, F.-H. Rouet, and X. Li. Matrix-free construction of hss representation using adaptive randomized sampling. In *ArXiv*, volume abs/1810.04125, 2018.
- [10] W. Hackbusch. A Sparse Matrix Arithmetic Based on \mathcal{H} -Matrices. Part I: Introduction to \mathcal{H} -Matrices. In *Computing*, volume 62, pages 89–108, Apr 1999. doi: 10.1007/s006070050015.
- [11] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On \mathcal{H}^2 -matrices. In H.-J. Bungartz, R.H.W. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-642-59709-1. doi: 10.1007/978-3-642-59709-1_2.
- [12] S. Harizanov, R. Lazarov, and S. Margenov. A survey on numerical methods for spectral space-fractional diffusion problems:. In *Fractional Calculus and Applied Analysis*, volume 23, pages 1605–1646, 2020. doi: doi:10.1515/fca-2020-0080.

- [13] S. Harizanov, S. Margenov, and N. Popivanov. Spectral Fractional Laplacian with Inhomogeneous Dirichlet Data: Questions, Problems, Solutions. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 123–138, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5.
- [14] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 1991. ISBN 0-534-13014-3.
- [15] T.A.M. Langlands, B.I. Henry, and S.L. Wearne. Fractional Cable Equation Models for Anomalous Electrodiffusion in Nerve Cells: Finite Domain Solutions. In *SIAM Journal on Applied Mathematics*, volume 71, pages 1168–1203, 2011. doi: 10.1137/090775920.
- [16] P.G. Martinsson. A Fast Randomized Algorithm for Computing a Hierarchically Semiseparable Representation of a Matrix. In *SIAM Journal on Matrix Analysis and Applications*, volume 32, pages 1251–1274, 2011. doi: 10.1137/100786617.
- [17] R. Musina and A.I. Nazarov. On fractional laplacians. In *Communications in Partial Differential Equations*, volume 39, pages 1780–1790. Taylor & Francis, 2014. doi: 10.1080/03605302.2013.864304.
- [18] V. Pasheva and R. Lazarov. Boundary element method for 2D problems of ideal fluid flows with free boundaries. In *Advances in Water Resources*, volume 12, pages 37–45, 1989. doi: 10.1016/0309-1708(89)90014-6.
- [19] E. Rebrova, G. Chávez, Y. Liu, P. Ghysels, and X.S. Li. A Study of Clustering Techniques and Hierarchical Matrix Formats for Kernel Ridge Regression. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 883–892, 2018. doi: 10.1109/IPDPSW.2018.00140.
- [20] L. Rosasco, M. Belkin, and E. De Vito. On learning with integral operators. In *Journal of Machine Learning Research*, volume 11, pages 905–934, 2010. URL <http://jmlr.org/papers/v11/rosasco10a.html>.
- [21] F.-H. Rouet, X.S. Li, P. Ghysels, and A. Napov. A Distributed-Memory Package for Dense Hierarchically Semi-Separable Matrix Computations Using Randomization. In *ACM Trans. Math. Softw.*, volume 42, pages 27:1–27:35, New York, NY, USA, June 2016. ACM. doi: 10.1145/2930660.

- [22] H.D. Simon and S.-H. Teng. How Good is Recursive Bisection? In *SIAM Journal on Scientific Computing*, volume 18, pages 1436–1445, 1997. doi: 10.1137/S1064827593255135.
- [23] D. Slavchev. Parallelization of boundary element method for laplasian equation. Master’s thesis, Technical University - Sofia, 2014.
- [24] D. Slavchev and S. Margenov. Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for bem discretization of flow around airfoils. In *Advanced Computing in Industrial Mathematics*, Cham, in press. Springer International Publishing.
- [25] H. Taitelbaum. Diagnosis using photon diffusion: From brain oxygenation to the fat of the atlantic salmon. In A. Pełkalski and K. Sznajd-Weron, editors, *Anomalous Diffusion From Basics to Applications*, pages 160–174, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49259-7.
- [26] P.N. Vabishchevich. Splitting schemes for non-stationary problems with a rational approximation for fractional powers of the operator. In *Applied Numerical Mathematics*, volume 165, pages 414–430, 2021. doi: <https://doi.org/10.1016/j.apnum.2021.03.006>.
- [27] R.C. Whaley. Atlas (automatically tuned linear algebra software). In D. Padua, editor, *Encyclopedia of Parallel Computing*, pages 95–101, Boston, MA, 2011. Springer US. ISBN 978-0-387-09766-4. doi: 10.1007/978-0-387-09766-4_85.
- [28] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. In *Numerical Lin. Alg. with Applic.*, volume 17, pages 953–976, 2010. doi: 10.1002/nla.691.
- [29] С. Маргенов. *Числени методи за системи с разреждени матрици*. Институт по паралелна обработка на информацията, Българска академия на науките, 2007. ISBN 9780198520115.

Abstracts of Dissertations

Number 4, 2022

INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES

БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Брой 4, 2022

Автореферати на дисертации