INSTITUTE OF INFORMATION AND
COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES

Dimitar Georgiev Slavchev

Abstract

of PHD thesys

COMPOSITE NUMERICAL METHODS AND SCALABLE TILE ALGORITHMS

for awarding
educational and scientific degree "Doctor of Philosophy"
in scientific major: 01.01.09 "Computational Mathematics"
in professional field: 4.5 "Mathematics"

Scientific Advisor:
Prof. Svetozar Margenov

January 19, 2022

The dissertation was discussed and admitted to the defense at an extended meeting of the "Scientific Computations with Laboratory on 3D Digitization and Microstructure Analysis" department of IICT-BAS on 11.01.2022.

The dissertation is structured i n introduction, four chapters, concluding remarks, originality of the results declaration, list of publications on which the work is based, approbation of the results, appendix and bibliography. The dissertation has 140 pages, 47 figures and 7 tables, 90 cited sources and 2 appendices.

The defense of the dissertation will be held on 17.05.2022 from 14:00 in room 218 of building 25A of IICT-BAS at an open meeting of the scientific jury composed of:

1. Prof. DSc Svetoslav Marinov Markov – IMI-BAS
2. Assoc. Prof. DSc Miglena Nikolaeva Koleva – RU "Angel Kanchev"
3. Assoc. Prof. PhD Kiril Stoyanov Shterev – IMech – BAS
4. Prof. PhD Pencho Genov Marinov – IICT-BAS
5. Prof. PhD Ivan Dimov Lirkov – IICT-BAS

Reserve members:
1. Assoc. Prof. PhD Petar Penchev Rashkov – IMI-BAS
2. Assoc. Prof. PhD Stanislav Nikolaev Harizanov IICT-BAS

The defense materials will be available in room 216 of IICT-BAS, Akad. G. Bonchev Str., building 25A.

Author: Dimitar Slavchev
Title: Composite numerical methods and scalable tile algorithms
(Съставни числени методи и скалируеми блочни алгоритми)

General characteristic of the dissertation

Relemance 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 [27].

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 dynamicsand 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 [26].

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 [23]; 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 Б).

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 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 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 $i = n-1, n-2, \ldots, 1$.

The computational complexity of the Gauss method is determined by the forward elimination [27]:

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

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

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

where $L_1, L_2, \ldots, L_{n-1}$ 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:

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

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 – $O\left(n^2\right)$.

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, fir 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 it's 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

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

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

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

Generators with the notation "big" can be computed outside of the highest levels of recursive compression. U ca 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. I 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 r. 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.

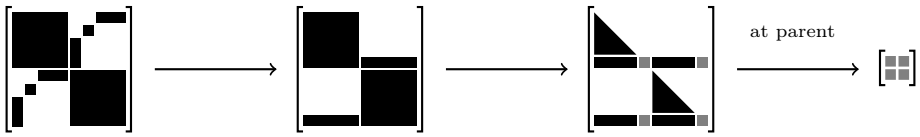


Figure 1: ULV-like factorization.

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

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

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, $\Psi_i nfty$ 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 $\overrightarrow{C} = (u,v)$ is defined by

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

satisfying the equations

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

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 $\overrightarrow{C}_\infty = (1, 0)$. 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 $\overrightarrow{C}_\infty$, we should choose Ψ_∞ such that

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

Here, we have

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

The boundary condition takes the form

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

Finally, the Kutta-Joukowsky'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 $\gamma_i = \gamma_h(s_i), i = 1, \ldots, n$ 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

$$\sum_{i=1}^{n} \gamma_i \Psi_{ji} = f_j, \qquad j = 1, 2, \ldots, n, \tag{6}$$

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 [25].

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

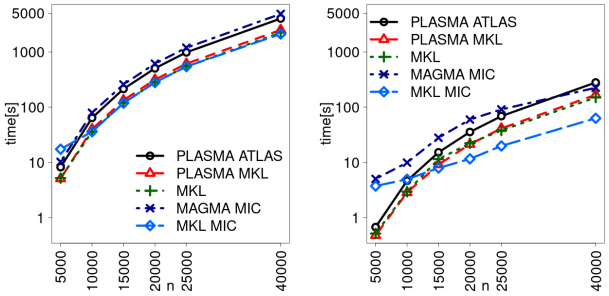
In this subsection we analyze the parallel effectiveness of the Intel Xeon Phi 7120P accelerators (MICs). The MICs are designed for massive parallelism and vectorization as required in High Performance computing. Every MIC has 61 cores and each core can run 4 threads simultaneously for a total of 244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

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



(a) Sequential (b) Parallel with 16 threads

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

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.

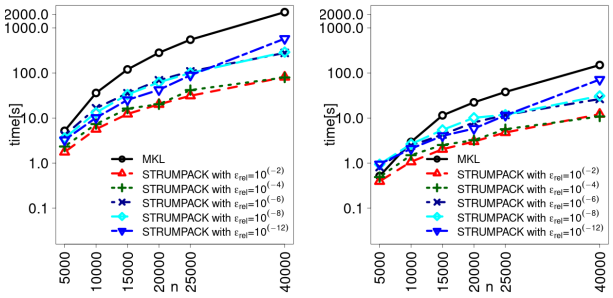
2.7 Hierarchical Semi-Separable compression

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.



(a) Sequential (b) Parallel with 16 threads

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:

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

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

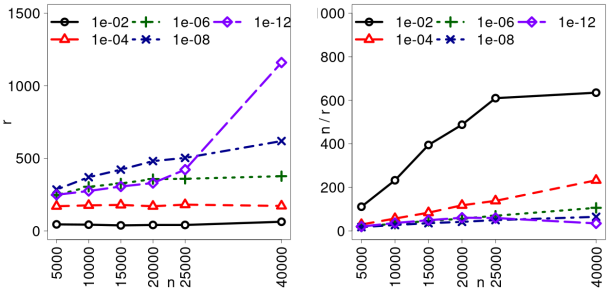
In Table 3 we show the relative errors R_{relative}, varying the size of the problem $n \in \{5\ 005, 10\ 005, 15\ 005, 20\ 005, 25\ 005, 40\ 005\}$, as well as the relative thresholds $\varepsilon_{\text{rel}} \in \{10^{-6}, 10^{-8}, 10^{-12}\}$.

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.

Table 3: Relative accuracy R_{relative}

n	ε_{rel} 10^{-6}	10^{-8}	10^{-12}	n	ε_{rel} 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



(a) Rank r (b) Relation r/n

Figure 4: Maximum off-diagonal rank r

2.8 Parallel scalability on computer systems with distributed memory

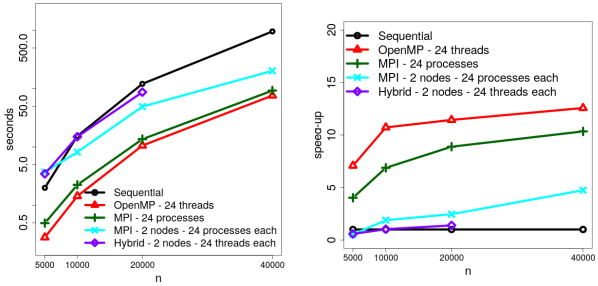
This section is concentrated on some specifics and hardships when working with HPC systems with hybrid architecture. Those machines have distributed memory on the server level and shared memory on each server. As in the previous section the comparative analysis includes the parallel libraries MKL and 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 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.

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

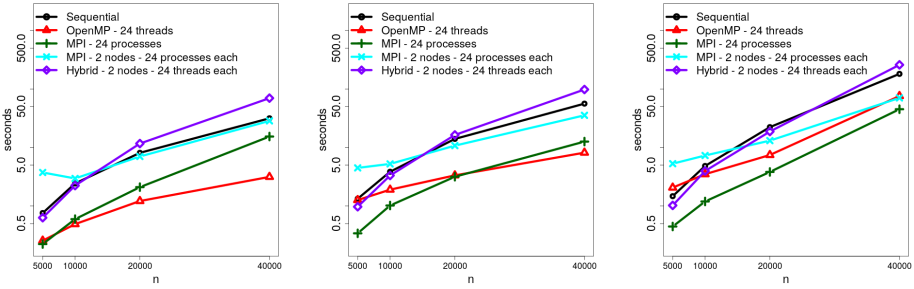


(a) MKL times (b) MKL parallel speed up

Figure 5: Parallel times and speed up with MKL.

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.



(a) STRUMPACK times with $\varepsilon_{rel} = 10^{-2}$ (b) STRUMPACK times with $\varepsilon_{rel} = 10^{-6}$ (c) STRUMPACK times with $\varepsilon_{rel} = 10^{-8}$

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 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

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

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

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

Here $\Omega \subset \mathbb{R}$ is a bounded domain, Ω^c is the compliment of Ω in 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

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

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

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

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 $\{\varphi_1, \ldots, \varphi_N\} \subset \mathbb{V}_h$ be the Lagrangian nodal basis, corresponding to the internal nodes $x_1, \ldots, x_N$. Then $\varphi_i(x_j) = \delta_i^j$. Let $T \in \mathcal{T}$ is a 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

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

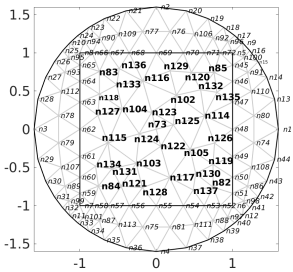
Under these conditions for any $\alpha \in (0, 1)$ the discreet analog of the variational problem (10) can be written as

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

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 $\overline{\Omega}$ and the compliment B^c is sufficient. This introduces the additional triangulation $\tilde{\mathcal{T}}_A$ over $V \backslash \Omega$, 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

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

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

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

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

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.

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.

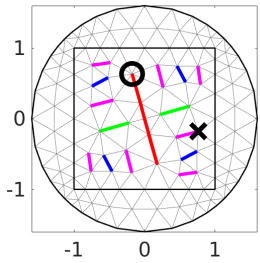
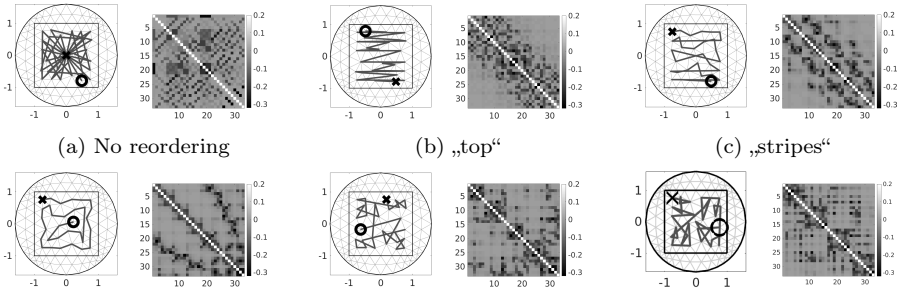


Figure 8: Reordering with nested dissection. "X" marks the first node, "O" marks the last.

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].



(a) No reordering (b) „top" (c) „stripes"

(d) „snake" (e) Nested Dissection (f) Recursive bisection

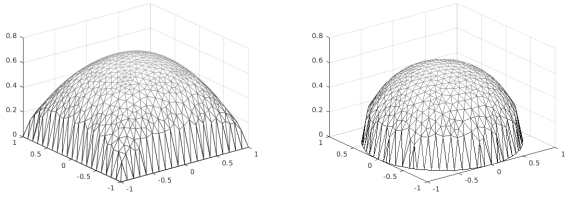
Figure 9: Reordering of the nodes in a square domain Ω and the structure of the corresponding matrix. Dark gray lines represent the ordering of the nodes from the first (marked with"X") and the last (marked with "O").

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.



(a) Square domain (b) Circle domain

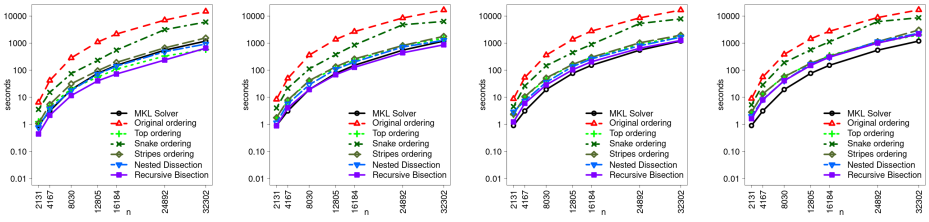
Figure 10: Solution of a *fractional* diffusion problem in square and circle domains.

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 $\varepsilon_{\text{rel}} = 10^{-2}$, 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.

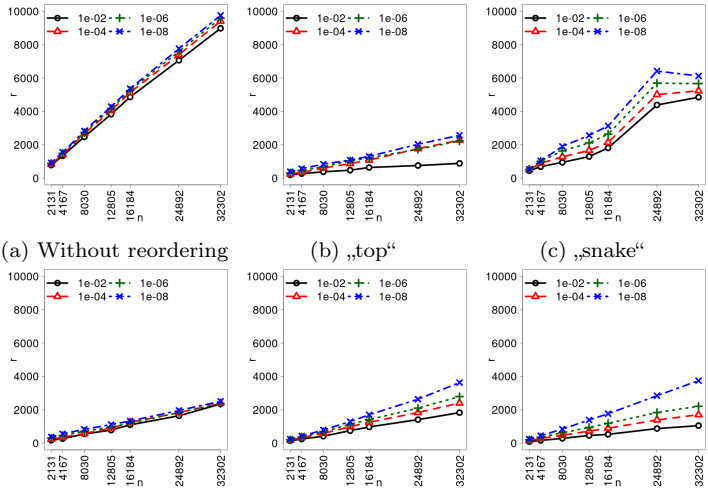


(a) $\varepsilon_{rel} = 10^{-2}$ (b) $\varepsilon_{rel} = 10^{-4}$ (c) $\varepsilon_{rel} = 10^{-6}$ (d) $\varepsilon_{rel} = 10^{-8}$

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.



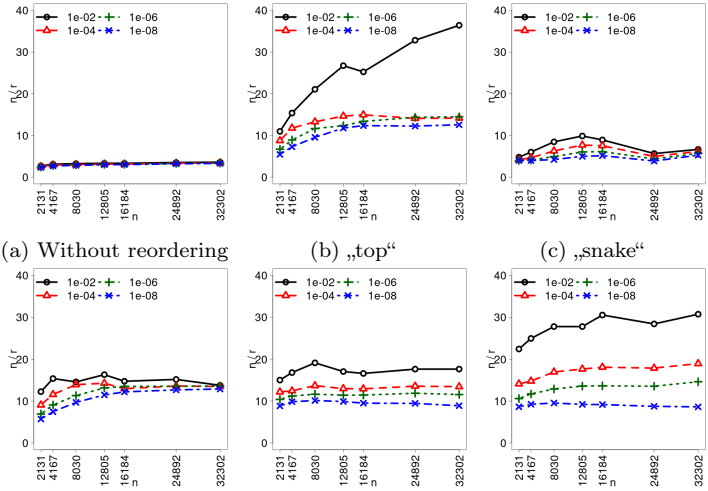
(a) Without reordering (b) „top“ (c) „snake“

(d) „stripes“ (e) Nested dissection (f) Recursive bisection

Figure 12: Maximum off-diagonal rank r.

3.6.2 Parallel times and speed-up

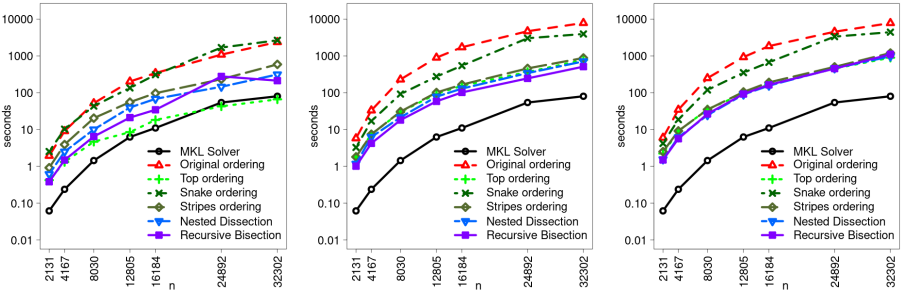
In this subsection we analyze the parallel speed up. The numerical experiments are carried out with increasing problem size $n \in [2\ 131, 32\ 302]$. The parallel results obtained with 16 threads are presented in Figure 14. The graphs have similar behavior an for the larger values of n the acceleration 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.



(a) Without reordering (b) „top“ (c) „snake“

(d) „stripes“ (e) Nested dissection (f) Recursive bisection

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



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

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

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

Table 4: Relative error for square domain.

	No reorder				„top“			
n	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	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	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	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	5.342-08	0.287	0.00108	7.943e-06	4.421-08
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	9.161-06	1.088e-07
	Nested dissection				Recursive bisection			
n	$R_{relative}$ for STRUMPACK with r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	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

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 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 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. Th 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 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 [24] 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

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

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 $M_L = \operatorname{diag}\left(m_L^i\right) \in \mathbb{R}^N$ 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 Б.

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

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

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

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

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 $O\left(n^3\right)$. Then the m steps in time require another $O\ left(n^2 m\ right)$ 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 [24], 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

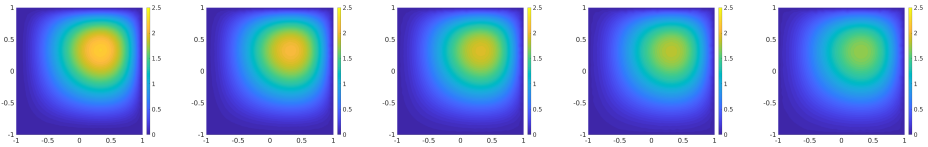
$$f(x) = \frac{(x_1+1)(x_2+1)}{4},$$

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 [24] .



(a) $t = 0$ (b) $t = 0.025$ (c) $t = 0.05$ (d) $t = 0.075$ (e) $t = 0.1$

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

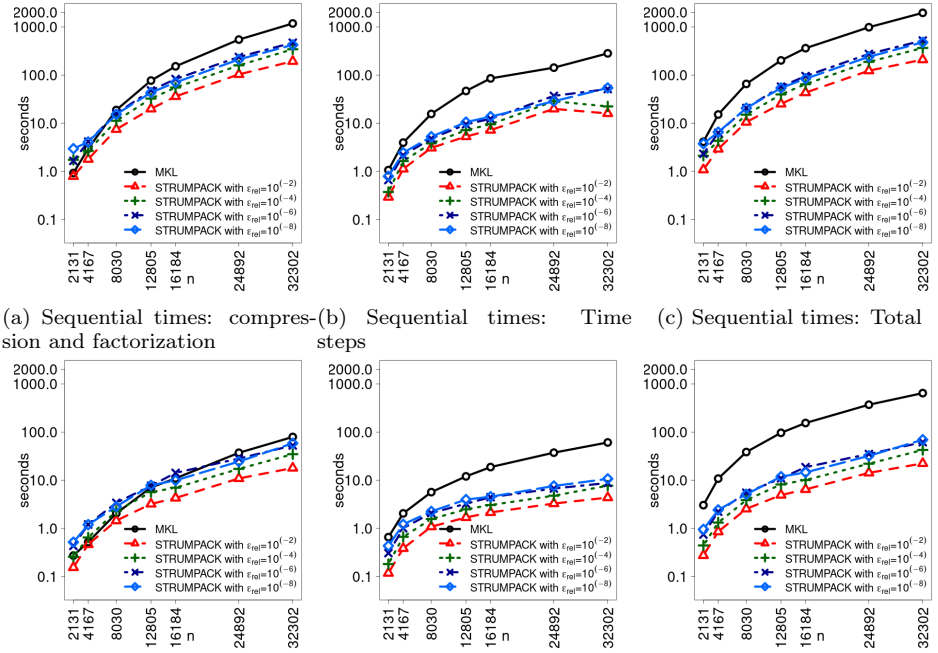
The numerical results analyzed in this chapter are obtained on computer systems with shared memory. As in previous chapters, the experiment was performed on the AVITOHOL supercomputer.

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 $\sim$2.5 and $\sim$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.



(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

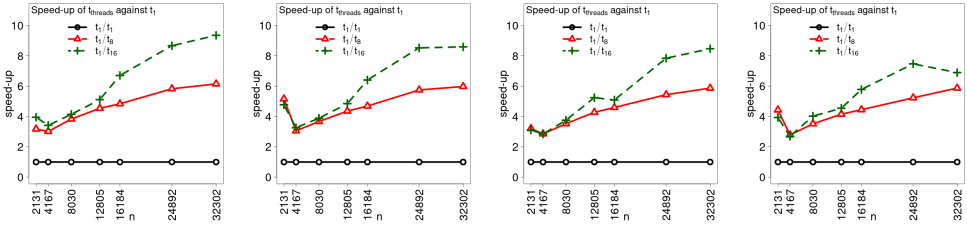
(f) Parallel times with 16 threads: Total

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

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 $\sim$3 (for $n = 2\ 131$) up to $\sim$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

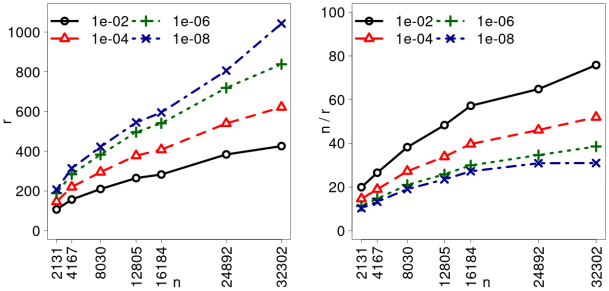
The maximum off-diagonal rank r, calculated during the HSS compression, is presented on Figure 18a, on Figure 18b the relation n/r is shown. This rank is a measure of the efficiency of the compression, and at the same time is detrimental in estimating the computational complexity. For the examined problem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



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

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

is strong, meaning that r/n is small, with the largest values of the relative threshold ε_{rel}. Thus with $\varepsilon_{\text{rel}} = 10^{-2}$ the rank r is between $\sim$20 and $\sim$80 times smaller than n, while with the finest threshold $\varepsilon_{\text{rel}} = 10^{-8}$ this relation is between $\sim$10 and $\sim$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.



(a) Max off-diagonal rank r (b) Relation of r and n

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.

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

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

n	Relative error at $t = 0.025$ $R_{relative}$ for r_{tol}				Relative error at $t = 0.05$ $R_{relative}$ for r_{tol}			
	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	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 $t = 0.075$ $R_{relative}$ for r_{tol}				Relative error at $t = 0.1$ $R_{relative}$ for r_{tol}			
	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	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	$1.31e{-06}$	$1.707{-07}$	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 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 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:

- 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

- Dimitar Slavchev and Svetozar Margenov. *Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs.* In Geno Nikolov, Natalia Kolkovska, and Krassimir 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

- Dimitar Slavchev. *On the impact of reordering in a hierarchical semi- separable compression solver for fractional diffusion problems.* In Ivan Lirkov and Svetozar 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 Ivan Georgiev, Hristo Kostadinov, and Elena 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

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 (AMiTaNS), 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 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 form 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 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. Pękalski 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 Lèvy 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] 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] H. Taitelbaum. Diagnosis using photon diffusion: From brain oxygenation to the fat of the atlantic salmon. In A. Pękalski 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.

[24] 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.

[25] 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.

[26] 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.

[27] С. Маргенов. *Числени методи за системи с разредени матрици.* Институт по паралелна обработка на информацията, Българска академия на науките, 2007. ISBN 9780198520115.