VILNIUS UNIVERSITY

Rima
KRIAUZIENĖ

Parallel algorithms for non-classical problems with big computational costs

DOCTORAL DISSERTATION

Natural Sciences,
Informatics N 009

VILNIUS 2019

This dissertation was written between 2015 and 2019 at Vilnius University.

Academic supervisor:
Prof. Habil. Dr. Raimondas Čiegis (Vilnius Gediminas Technical University, Natural Sciences, Informatics – N 009).

Academic consultant:
Prof. Dr. Julius Žilinskas (Vilnius University, Natural Sciences, Informatics – N 009).

VILNIAUS UNIVERSITETAS

Rima
KRIAUZIENĖ

Didelės skaičiavimo apimties neklasikinių uždavinių sprendimo lygiagretieji algoritmai

DAKTARO DISERTACIJA

Gamtos mokslai,
Informatika N 009

VILNIUS 2019

Disertacija rengta 2015–2019 metais Vilniaus universitete.

Mokslinis vadovas:
prof. habil. dr. Raimondas Čiegis (Vilniaus Gedimino technikos universitetas, gamtos mokslai, informatika – N 009).

Mokslinis konsultantas:
prof. dr. Julius Žilinskas (Vilniaus universitetas, gamtos mokslai, informatika – N 009).

SUMMARY

This research focuses on parallel algorithms, which are important for solving contemporary problems. Parallel algorithms help to solve the problems of memory limitation and computational time, when we can not find the solution reasonably fast using a single core on the fastest computers.

The object of the thesis are efficient parallel algorithms for problems with big computational costs, including optimization problems.

In the first chapter of the dissertation the research area and problem, the aim and objectives of the research, the structure of the dissertation are presented.

In the second chapter of the dissertation the non-local problem with fractional powers of the Laplacian was analysed. In this dissertation, we study and compare parallel algorithms for various most recent numerical methods proposed for solving the fractional powers of elliptic problems. We consider four different numerical methods. These methods are based on the general approach: the given non-local differential problem is transformed to a local differential problem of elliptic or pseudo-parabolic type, but formulated in a higher dimensional space $\mathbb{R}^{d+1}$, if $\Omega \subset \mathbb{R}^d$. The scalability and convergence analysis of parallel algorithms for these problems was performed. Recommendations to achieve given accuracy for the provided fractional power coefficient were specified.

In the third chapter of the dissertation the detailed analysis of absorbing boundary conditions for the linear Schrödinger equation was performed. We were interested in methods based on the approximation of exact transparent boundary conditions by rational functions. Different strategies were investigated to find coefficients of approximations. In this research we compare the state-of-art methods. Recommendations for constructing absorbing boundary conditions for the one-dimensional Schrödinger equation using investigated methods were presented. The proposed methodology has shown

that it is possible to find the accurate absorbing boundary conditions for four qualitatively different tasks.

In the fourth chapter of the dissertation a three-level parallelisation scheme was proposed. The possibilities of this methodology are demonstrated for solving local optimization problems from the second part of dissertation. Comparing the three-level scheme to the classical two level scheme, the proposed scheme increases the amount of computational resources, which can be used efficiently.

To show that these ideas work for a broad scope of applications, we discussed the possibility to apply the proposed scheme to other examples. The comparison of different Nelder-Mead parallelisation methods is discussed.

The dissertation consists of Introduction, three chapters, conclusions and bibliography. The chapters are divided into sections, sections – into subsections. The scope of the dissertation is 106 pages including 10 figures and 30 tables. The list of references consists of 103 sources.

The results of this research were presented at five international and three national Lithuanian conferences, at the PhD Symposium 2018 of the 3rd NESUS Winter School. Three articles were published in periodical scientific publications in journals referred to ISI Web of science. One article was published in Conference Proceedings.

SANTRAUKA

Šis darbas skirtas lygiagretiesiems algoritmams, kurie ypač svarbūs sprendžiant šiuolaikinius uždavinius. Lygiagretieji algoritmai padeda išspręsti atminties išteklių ribojimo ir skaičiavimo laiko problemas, kai naudojant pačius greičiausius šiuolaikinius nuosekliuosius kompiuterius negalime laiku rasti atsakymo.

Disertacijos tyrimo objektas – lygiagretieji algoritmai, skirti uždaviniams, susijusiems su didelėmis skaičiavimo sąnaudomis, taip pat optimizavimo uždaviniai.

Pirmajame disertacijos skyriuje aptariama tyrimo sritis ir problema, darbo tikslas ir uždaviniai, disertacijos struktūra.

Antrajame disertacijos skyriuje išnagrinėtas nelokalus uždavinys su Laplaso operatoriumi, pakeltu trupmeniniu laipsniu. Disertacijoje nagrinėjami ir lyginami lygiagretieji algoritmai, skirti skirtingiems naujausiems skaitiniams metodams, kurie literatūroje siūlomi uždaviniams su elipsiniu operatoriumi, pakeltu trupmeniniu laipsniu, spręsti. Pasirinkti keturi skirtingi naujausi skaitiniai metodai. Šie metodai pagrįsti bendra idėja: nelokalusis diferencialinis uždavinys transformuojamas į lokalųjį pseudo-parabolinio arba elipsinio tipo diferencialinį uždavinį, tačiau suformuluotą aukštesnės dimensijos $\mathbb{R}^{d+1}$ erdvėje, kai elipsinis operatorius buvo iš $\mathbb{R}^d$. Atlikta lygiagrečiųjų algoritmų išplečiamumo ir konvergavimo analizė. Pateiktos rekomendacijos, kaip pasiekti norimą tikslumą atitinkamam elipsnio operatoriaus trupmeniniam laipsniui β.

Trečiajame disertacijos skyriuje atlikta išsami tiesinės Šriodingerio lygties sugeriančių kraštinių sąlygų konstravimo analizė. Naudojami metodai paremti tikslių pralaidžių kraštinių sąlygų aproksimavimu racionaliosiomis trupmenomis. Nagrinėti ir lyginti skirtingi metodai, kurie leidžia rasti racionaliųjų funkcijų koeficientus. Pateikta rekomendacijų, kaip naudojant šiuos metodus konstruoti sugeriančias kraštines sąlygas vienmatei Šriodin-

gerio lygčiai. Pasiūlyta metodika parodė, kad galima rasti sugeriančias kraštines sąlygas keturiems kokybiškai skirtingiems uždaviniams.

Ketvirtajame disertacijos skyriuje pasiūlyta trijų lygmenų lygiagretinimo schema. Šios metodikos galimybės rodomos sprendžiant lokalaus optimizavimo uždavinį iš trečiojo disertacijos skyriaus. Pasiūlyta trijų lygmenų lygiagretinimo schema, lyginant su klasikine dviejų lygių schema, padidina skaičiavimo išteklių kiekį, kurie gali būti efektyviai naudojami. Siekiant parodyti, kad ši idėja gali būti plačiai taikoma, aptarėme galimybę taikyti siūlomą schemą ir kitiems pavyzdžiams. Šioje tyrimo dalyje atliekamas skirtingų simplekso metodų lyginimas.

Disertaciją sudaro įvadas, trys skyriai, literatūros sąrašas, publikacijų sąrašas, santrauka lietuvių kalba ir padėka. Disertacijos skyriai padalyti į poskyrius, o poskyriai – į skyrelius. Disertacijoje yra 106 puslapiai, 10 paveikslų ir 30 lentelių. Disertacijoje cituojami 103 informacijos šaltiniai.

Disertacijos rezultatai pristatyti penkiose tarptautinėse konferencijose, trijose konferencijose, vykstančiose Lietuvoje, dviejose jaunųjų mokslininkų konferencijose ir Nesus organizuojamos žiemos mokyklos doktorantų simpoziume Zagrebe. Disertacijos tema paskelbti trys straipsniai, įtraukti į Thompson Reuters ISI Web of Science duomenų bazę ir turi citavimo indeksą, vienas straipsnis paskelbtas konferencijų medžiagoje.

Contents

NOTATION

p	Number of processes
S_p	Speed-up of parallel algorithm
E_p	Efficiency of parallel algorithm
$\|v\|$	L_2 norm, $\|v\| = \|E\|_2 = \left(\int_\Omega |v(x)|^2 dx \right)^{1/2}$
$\|\psi\|_\infty$	Maximum norm, $\|\psi\|_\infty = max\{|\psi_1|, \ldots, |\psi_N|\}$

Vectors are denoted with bold letters.

Abbreviations

$1D$	1-Dimensional
$2D$	2-Dimensional
$3D$	3-Dimensional
CG	Conjugate Gradients
FDM	Finite difference method
FFT	Fast Fourier transformation
FS	Fourier symbol
FVM	Finite volume method
NM	Nelder-Mead
MG	Multigrid
PCG	Preconditioned conjugate gradient
PDE	Partial differential equation
R	Reflection coefficient

Chapter 1

INTRODUCTION

1.1 Research Area and Relevance of the Problem

Nowadays, we need to solve problems with big computational costs, when we can not find a solution reasonably fast using a single core on the fastest computers. Another limitation of calculations is high memory requirements, that can not be fulfilled using a single shared memory computer. Thus, parallel computations are necessary.

In this dissertation, two problems are investigated. The first problem is described by fractional powers of elliptic operators [12,72,73]. This non-local problem has an important property: the increasment of the problem size greatly increases the numerical costs of computation. Parallel computing makes the application of such non-local models more feasible and attractive. However, the efficient parallel computations require the application of appropriate parallel algorithms and a detailed theoretical analysis. We investigate and compare the parallel numerical algorithms for different state-of-the-art [13, 45, 72, 90] numerical methods proposed to solve the non-local problems described by elliptic operators of fractional powers [12, 13, 72, 73]. In this research, the non-local problem is transformed to some local (classical) differential problem of elliptic or pseudo-parabolic type, formulated in a space of higher dimension $\mathbb{R}^{d+1}$, if $\Omega \subset \mathbb{R}^d$. We investigate the weak and strong scalability of the developed parallel algorithms. Two- and three-dimensional test problems are solved and the results of extensive convergence tests are presented. The main aim of this part of dissertation is to determine, which parallel algorithms can be recommended to achieve certain accuracy for the given fractional power coefficient.

The second non-local problem of this dissertation deals with the construction of absorbing boundary conditions for the one-dimensional Schrö-

dinger equation [5,7,66,88]. A simple standard boundary conditions are formulated on the boundaries of the restricted domain (e.g. the homogeneous Dirichlet boundary conditions), the solution after reaching the boundary will be reflected back into the domain and will pollute the results of subsequent simulations. Thus it is a challenge to construct appropriate local boundary conditions and to avoid the negative long memory computational effects included into the definition of the exact non-local transparent boundary conditions. We are interested in methods based on the approximation of the exact transparent boundary conditions by rational functions. Different strategies are investigated for the optimal selection of the coefficients of rational functions, including the Padé approximation, the L_2 norm approximations of the Fourier symbol, L_2 minimization of a reflection coefficient, techniques, based on minimization in two different norms for the chosen benchmark problems with known exact solutions and coupled adaptive strategy. The formulated minimization problems are considered as black box problems of global optimization. Since the objective functions of the attacked global optimization problems are computationally expensive, experiments take a considerable amount of time. Parallel computing should be applied to make experiments faster. The proposed minimisation technique showed that it is possible to find optimal values of coefficients that suit both test problems with the errors that are small enough for many modelling purposes.

We propose a general methodology for parallelisation of algorithms that address the considered problems. The optimization problems that are solved during the approximation of absorbing boundary conditions for the Schrödinger equation fits this methodology very well. The strategy of parallelisation has three-levels. Different parallelisation levels of this scheme give new parallelisation possibilities; at the same time they also and create different computational challenges. The important part of the proposed three-level scheme is based on a assumption that there exists parallel alternatives to the original sequential optimization algorithm on the first level. The first level of parallelisation template becomes a part of a new parallel algorithm and the degree of the first level parallelism can be selected dynamically during the computations. The parallelisation technologies on the first level vary depending on a problem, there are no concrete methods that would work in all cases. However, we are presenting a general abstract procedure of application of this approach. On the second level, a set of computational tasks with different computational sizes is defined. These tasks can be solved independently, which means this part can be done in parallel. The third level is defined by parallel algorithms used to solve tasks from the second

level. We optimise the workload distribution, for this purpose we propose a greedy workload balancing heuristic and test it on benchmark problems.

To show that these ideas work for a broad scope of applications, we discuss the possibility to apply the proposed three level parallelisation scheme to other examples.

1.2 The object of the thesis

The object of the thesis is to create and analyse efficient parallel algorithms for problems with big computational costs including optimization problems.

1.3 The Aim and Objectives of the Research

The aim of the research is to investigate efficient parallel algorithms for problems with the fractional powers of elliptic operators and optimization problems with big computational costs.

Thesis tasks are:

- Tasks for constructing parallel algorithms for problems with fractional powers of elliptic operators:
  - the review the parallel algorithms for problems with fractional powers of elliptic operators;
  - propose and analyse efficient parallel algorithms for problems with fractional powers of elliptic operators;
  - perform scalability analysis of parallel algorithms;
  - formulate conclusions and recommendations for problems with fractional powers of elliptic operators.
- Tasks for construction of absorbing boundary conditions for the one-dimensional Schrödinger equation:
  - to review of literature for constructing absorbing boundary conditions for the Schrödinger equation;
  - propose and investigate methods for finding coefficients for boundary conditions;
  - perform calculations with different initial boundary conditions for the Schrödinger problem;

  - formulate conclusions and recommendations for constructing absorbing boundary conditions for the one-dimensional Schrödinger equation using proposed methods.

- Tasks for three-level parallelisation scheme:
  - propose a three-level parallelisation scheme for solving optimizations problems;
  - perform the detailed analysis of this scheme;
  - apply the three-level scheme for optimization problems;
  - formulate conclusions and recommendations for using all possibilities of this scheme.

1.4 Research Methodology

In the dissertation we use numerical schemes (finite volume method (FVM), the finite difference method (FDM)) from mathematical modelling theory, the methods of convergence analysis from the theory of algorithms, scalability analysis from the theory of parallel computing, multigrid, domain decompasition method, simplex method from the optimization theory.

1.5 Novelty of the Thesis

- In this dissertation, parallel algorithms for various most recent numerical methods for solving the fractional powers of elliptic operators problems were investigated and compared. Scalability and convergence analysis of parallel algorithms was performed. Recommendations to achieve a given accuracy for the provided fractional power β coefficient were the specified.

- Recommendations to construct absorbing boundary conditions for the one-dimensional Schrödinger equation using investigated methods were presented. The proposed methodology showed that it is possible to find the accurate absorbing boundary conditions for four qualitatively different problems.

- The proposed three-level parallelisation scheme comparing to the classical two level algorithm increases the amount of computational resources and lets us to achieve additional speed-up. It was shown, that the proposed methodology lets us use available resources efficiently.

1.6 Practical Value of the Research Findings

- The amount of calculations for solving a non-local problem with fractional power of elliptic operators are large and even a single solution of this problem without parallel calculations is difficult. In this dissertation, scalability analysis of parallel algorithms was performed. Different methods were compared using parallel algorithms. The recommendations to achieve a given accuracy for the provided fractional power coefficient β help to select the appropriate method.

- The three-level parallelisation scheme and recommendations for usage of this scheme were proposed. This scheme can be applied to problems efficiently in the case when a big number of computational resources is available efficiently.

1.7 Defended Statements

1. For the given cases of test problems with fractional powers of elliptic operators the recommended parallel algorithms are optimal in terms of computational time in order to achieve certain accuracy for the provided fractional power coefficients.

2. Using the proposed methodology there were found absorbing boundary conditions for four qualitatively different tasks in a specified seven parameters space.

3. The proposed three-level scheme improves the degree of parallelism and improves the amount of available computational resources, which gave to achieve additional speed-up.

1.8 Presentation and Approbation of the Results

The results of this research were presented at five international and three national Lithuanian conferences, at the PhD Symposium 2018 of the 3rd NESUS Winter School. Three articles were published in periodical scientific publications in journals referred to ISI Web of science. One article was published in Conference Proceedings. The detailed list of publications can be found in the "List of Publications by the Author on the Topic of the Dissertation" theme.

The results of this research were presented at the following conferences

- MMA2019: 24nd international conference, May 28-31, 2019, Tallinn, Estonia.
- DAMSS: 10th international workshop on data analysis methods for software systems, November 29–December 1, 2018, Druskininkai.
- DAMSS: 9th international workshop on data analysis methods for software systems, November 30–December 2, 2017, Druskininkai.
- MMA2018: 23nd international conference, May 29-June 1, 2018, Sigulda, Latvia.
- DAMSS: 8th Data Analysis Methods for Software Systems, December 1-3, 2016, Druskininkai, Lithuania.
- MMA2017: 22nd International Conference Mathematical Modelling and Analysis, May 30-June 2, 2017, Druskininkai, Lithuania.
- 12th International Conference on Parallel Processing and Applied Mathematics [PPAM 2017], September 10-13, 2017, Lublin, Poland.
- MMA2016: 21st International Conference Mathematical Modelling and Analysis, June 1-4, 2016, Tartu.

Schools

- 3rd NESUS Winter School and the PhD Symposium 2018, 22nd-25th January 2018, Zagreb, Croatia. Topic of speech was "Numerical analysis and optimization of parallel algorithms for problems with big computational costs." http://nesusws.irb.hr/images/BookofAbstracts.pdf
- NESUS Winter School & PhD Symposium 2016, February 8–11, 2016, West University of Timisoara, Romania.

Contributing talks were given at the seminars at the Institute of Data Science and Digital Technologies in Vilnius University and at the Department of Mathematical Modelling of Vilnius Gediminas Technical University.

1.9 Structure of the Dissertation and Main Results

The dissertation consists of Introduction, three chapters, conclusions and bibliography. The chapters are divided into sections, sections – into subsections. The scope of the dissertation is 106 pages including 10 figures and 30 tables. The list of references consists of 103 sources.

Chapter 2

Parallel algorithms for problems of fractional powers of elliptic operators*

In this chapter we investigate the parallel numerical algorithms for four different state-of-the-art numerical methods for solving the non-local problems described by fractional powers of elliptic operators. These methods transform the non-local problem into some local differential problems of elliptic or pseudo-parabolic types, formulated in a space of higher dimension $\mathbb{R}^{d+1}$, if $\mathbb{R}^d$. The selected four basic methods lead to different properties of the constructed parallel algorithms. The proposed parallel algorithms for these numerical methods are based on the domain decomposition and master-slave methods.

The scalability and convergence analysis of parallel algorithms for these problems was performed. Recommendations to achieve certain accuracy for the provided fractional power coefficient were provided.

Parts of this chapter are published in [23–25].

*1. Čiegis, R.; Starikovičius, V.; Margenov, S; and Kriauzienė. Parallel solvers for fractional power diffusion problems. *Concurrecy Comput.: Pract. Exper.*, 2017, Vol. 29,, iss. 24, p. 1–12. DOI: 10.1002/cpe.4216. 2. Čiegis, R.; Starikovičius, V.; Margenov, S; and Kriauzienė. A comparison of accuracy and efficiency of parallel solvers for fractional power diffusion problems. In *Parallel Processing and Applied Mathematics, (PPAM2017, Lublin, Poland, September 9–13, 2017) Proceedings, part I*, volume 10777 of *Lecture Notes in Computer Science*, pages 79–89, Berlin, Heidelberg, 2018. Springer. 3. Čiegis, R.; Starikovičius, V.; Margenov, S; and Kriauzienė. Scalability analysis of different parallel solvers for 3D fractional power diffusion problems. *Concurrecy Comput.: Pract. Exper.*, 2019. DOI: 10.1002/cpe.5163

2.1 Introduction to this Chapter

Fractional derivatives are more and more often used for the mathematical modeling of various problems in physics [46, 76], geophysics [17], chemistry [100], biology [62], image processing [45,98] and finance [83]. Fractional-order derivatives are able to describe anomalous behaviors of various materials, which are in between ideal solids and Newtonian fluids, such as granular materials, colloids, polymers, emulsions, sediments, biological materials, multiphase fluids, et al. [52, 100]. The behavior of these materials often does not obey to the standard gradient laws, such as, Fick's law of diffusion, Fourier's law of heat conduction, Newtonian viscosity, Darcy law of fluid flow through a porous medium et. al. [46]. The fractional-order models appear to be more adequate than the standard models in the description of the long range interactions, memory and hereditary properties of different substances [28, 61, 76].

In spite of the increasing popularity of the fractional-order models, there is a significant difference in the number of applications of time-fractional models compared to the space-fractional ones. This can be partially explained by the fact that mathematical definitions of the space-fractional derivatives, notably of the multi-dimensional fractional Laplacian, are much more complicated [31, 76]. However, more important is that the space-fractional models are computationally very expensive when applied to the higher dimensional cases. Non-local nature of the fractional models leads to a challenging numerical discretisation. Numerical techniques for the multidimensional space-fractional models remain far from mature [43, 99]. Because of these difficulties significantly less results has been reported on the space-fractional derivative modeling in the literature [28, 36].

Nowadays, application of parallel computing technologies presents a natural approach to make the space-fractional derivative modelling more feasible and attractive. However, efficient parallel computations require application of appropriate parallel algorithms. The permanent development of the parallel computing systems with all their diversity requires a constant attention, which must be paid to a development and selection of proper parallel algorithms for the solution of various problems.

There are several different definitions of fractional power of elliptic operators. Fractional powers of elliptic operators are usually defined through integral expressions with singular kernels [12, 31, 76]. Standard numerical solution methods lead to the solution of systems of linear equations with dense matrices, what is computationally very expensive. We use a differ-

ent definition based on the spectral decomposition of the elliptic operator and the following solution approach. The non-local problem with fractional power of the Laplacian is transformed to a local differential problem of elliptic or pseudo-parabolic type, but formulated in a higher dimensional space $\mathbb{R}^{d+1}$, if $\Omega \subset \mathbb{R}^d$. An important advantage of this approach is that the obtained differential models are widely used in many applications; thus, the related numerical solution methods are well developed. Numerous efficient numerical software packages are available, which are subject to a long-time development and permanent improvements.

Four different transformations are considered. The first one transforms the non-local problem with fractional power of elliptic operator to a local elliptic problem with a singular diffusion coefficient [9, 12, 65, 72, 73]. For the second one, the non-local problem is transformed to a pseudo-parabolic problem [32, 90, 91]. In the third approach [12, 13], quadrature formulas are applied to the integral representation of a sought solution. The last method [44, 45] is based on best uniform rational approximation (BURA) of a scalar function of the form t^γ, $\gamma \in (0,1)$, $t \in [0,1]$. This method conceptually differs from the previous three, as the original dimensionality of the problem is preserved.

We have developed different parallel algorithms for these numerical methods. These solvers are based on the domain decomposition and master-slave methods. We use the finite volume method to approximate the differential problems. Open source parallel multigrid libraries are employed for the numerical solution of arising systems of linear equations.

It is important to note that the selected four basic methods lead to different properties of the constructed parallel algorithms. The main objective of our research is to analyse and compare their scalability, efficiency, and accuracy.

2.2 Definitions of Fractional Power of Elliptic Operators

There are different definitions of fractional power of elliptic operators [12]. The integral definition of fractional Laplacian in a bounded domain $\Omega \subset \mathbb{R}^d$ is introduced by the Riesz potential:

$$(-\Delta)^\beta u(x) = C(d,\beta) P.V. \int_{\mathbb{R}^d} \frac{u(x) - u(x')}{|x - x'|^{d+2\beta}} dx' = f(x), \quad u = 0 \text{ in } \Omega^c = \mathbb{R}^d \setminus \Omega, \tag{2.1}$$

where P.V. stands for the Cauchy principle value and $C(d,\beta)$ is a normalization constant. In study, [1] error bounds in the energy norm and numerical experiments (in 2D) are presented for the integral definition of fractional Laplacian, demonstrating an accuracy of the order $O(h^{\frac{1}{2}}\ln h)$ for solutions obtained by means of linear elements on a quasi uniform mesh. The numerical solution of problems involving such a non-local operator is rather complicated. There are at least two major reasons for that: highly singular kernels and an unbounded region of integration. The standard numerical methods lead to systems of linear equations with dense matrices. This is computationally expensive.

Another definition is based on the spectral decomposition of the elliptic operator. Let Ω be a bounded domain in R^d. In order to find $u \in V$ we define the bilinear form [13]:

$$a(u,v) := \int_\Omega (\mathbf{a}(x)\nabla u(x)\cdot\nabla v(x) + q(x)u(x)v(x))\,dx, \quad \forall v \in V, \qquad (2.2)$$

where $V := \{v \in H_0^1(\Omega): \quad v(x) = 0 \ \text{ on } \ \Gamma_D\}$, $\Gamma = \partial\Omega$, and $\Gamma = \bar{\Gamma}_D \cup \bar{\Gamma}_N$. We assume that Γ_D has a positive measure, $q(x) \geq 0$ in Ω, and $\mathbf{a}(x)$ is a symmetric and positive definite $d \times d$ tensor product matrix, uniformly bounded in Ω. Let $\tau : L^2(\Omega) \to V$, where for $f \in L^2(\Omega)$ the function $u = \tau f \in V$ is the unique solution to $a(u,\psi) = (f,\psi), \forall \psi \in V$, and (v,u), for $u, v \in L^2(\Omega)$ is the inner product in $L^2(\Omega)$. Let $\mathcal{L} = \tau^{-1}$. Then, fractional power of elliptic operator $\mathcal{L}^\beta$, $0 < \beta < 1$, is introduced through its spectral decomposition, that is,

$$\mathcal{L}^\beta u = f, \quad \mathcal{L}^\beta u(x) = \sum_{i=1}^{\infty} \lambda_i^\beta c_i \psi_i(x), \quad \text{where} \quad u(x) = \sum_{i=1}^{\infty} c_i \psi_i(x), \qquad (2.3)$$

where $\{\psi_i(x)\}_{i=1}^\infty$ are the eigenfunctions of $\mathcal{L}$, orthonormal in L_2-inner product and $\{\lambda_i\}_{i=1}^\infty$ are the corresponding positive real eigenvalues.

Bonito and Pasciak [13] showed that $H^\beta = \{v \in L^2 : \sum\limits_{i=1}^{\infty} \lambda_i^{2\beta} |\,(v,\psi_j)\,|^2 < \infty\}$ is a Hilbert space under the inner product $a_\beta(v,w) := \left(\mathcal{L}^{\beta/2}v, \mathcal{L}^{\beta/2}w\right)$ for all $v, w \in H^\beta$. The weak formulation of the elliptic problem is as follows: find $u \in H^\beta$ such that

$$a_\beta(u,v) = (f,v), \quad \forall v \in H^\beta.$$

Thus, a unique solution of this problem is $u = \tau^\beta f = \sum\limits_{i=1}^{\infty} \lambda_i^{-\beta}(f,\psi_i)\psi_i$.

Let us assume that linear elements are used to obtain the finite element

method's approximation $U_h \in V_h$, where h is the mesh size and $V_h \subset H_0^1(\Omega)$ is the space of continuous piece-wise linear functions over the mesh. In the case of full regularity, the best possible convergence rate for $f \in L^2(\Omega)$ is, cf., [13]

$$\|u - U_h\|_{L^2(\Omega)} \leq Ch^{2\beta} |\ln h| \ \|f\|_{L^2(\Omega)}. \tag{2.4}$$

This estimate illustrates how the accuracy of the numerical method is reduced, depending on power $\beta \in (0, 1)$. This property requires solving the fractional diffusion problems on fine meshes.

The research about the relations between these two fractional powers of elliptic operators definitions is still ongoing. However, it is known [12] that they are not equivalent and may produce different solutions over bounded domains.

The direct implementation of this approach is very expensive. It requires computing all eigenvectors and eigenvalues of large matrices. The given spectral algorithm can be used for practical computations if the fractional power of the Laplace operator is solved in a rectangular domain, when the basis functions are known in advance and the fast Fourier transform (FFT) techniques can be applied. In such cases the computational complexity of solving fractional power elliptic problems is the same as for solving classical elliptic problems. However, we are interested in more general methods, which can be applied in general domains for general elliptic operators.

2.3 State of the Art in Numerical Solution Methods

The existing most advanced numerical methods are based on the following general approach. The non-local differential problem $\mathcal{L}^\beta u = f$ is transformed to a local differential problem, which formulated in a higher dimensional space $\mathbb{R}^{d+1}$. The introduction of an additional dimension considerably increase computational complexity of fractional-in-space diffusion problems compared to the classical diffusion problems. Also the proposed algorithms require more memory. This makes parallel computations to be necessary in order to solve such problems.

Four different numerical methods are used, they are denoted by (M1)–(M4).

M1 Extension to a mixed boundary value problem in the semi-infinite cylinder $\Omega \times [0, \infty)$ [9, 12, 65, 72, 73]. The semi-infinite cylinder is

then truncated by finite cylinder to allow numerical solution of elliptic problem in a bounded domain $C_T = \Omega \times [0, T] \subset \mathbb{R}^{d+1}$.

M2 Transformation to a pseudo-parabolic problem [32,90,91]. Here the extended dimension is defined by the pseudo-time. The obtained pseudo-parabolic problem is solved numerically. This approach is further applied to problems with fractional order boundary conditions [57].

M3 Integral representation of a sought solution of the non-local problem [12, 13]. Different quadrature formulas can be applied to evaluate numerically the related integrals.

M4 Alternative methods for solving algebraic systems $L_h^\beta U_h = f_h$ (discrete problems). These methods are based on the best uniform rational approximations of the function $t^{1-\beta}$ for the BURA method [45] and of the function t^β for the R-BURA method [44] in the interval $[0, 1]$. Computationally, a limited number of linear systems need to be solved independently, the matrices of these systems are the positive diagonal shifts of the (original, non-fractional) discrete elliptic operator.

All methods (M1)–(M4) are applicable to fractional diffusion problems in computational domains with general geometry. For methods (M1)–(M3), the computational domain is of higher $(d + 1)$ dimension.

2.4 PDE Approximations of the Fractional Model

We formulate PDE models to approximate problems involving fractional powers of elliptic operators. These approximations allow us to construct efficient numerical solution algorithms for the original non-local problem. The formulated PDEs are approximated by finite volume schemes.

2.4.1 Extension to a Mixed Boundary Value Problem in the Semi-Infinite Cylinder $C = \Omega \times (0, \infty) \subset \mathbb{R}^{d+1}$ (M1)

The non-local problem (2.3) is equivalent to the following classical local linear problem in the extended space $\mathbb{R}^{d+1}$ [72, 73]:

$$-\frac{\partial}{\partial y}\left(y^{\alpha}\frac{\partial U}{\partial y}\right)+y^{\alpha}\mathcal{L}U=0,\quad (x,y)\in C,\ C=\Omega\times(0,\infty)\ \ \alpha=1-2\beta, \tag{2.5}$$

$$-y^{\alpha}\frac{\partial U}{\partial y}=d_{\beta}f,\quad (x,0)\in\bar{\Omega}\times\{0\},$$

$$U=0,\quad (x,y)\in C_B=\partial C\setminus\bar{\Omega}\times\{0\},$$

where d_β is a positive normalization constant that depends only on β. Then the solution of problem (2.3) is obtained by $u(x)=U(x,0)$.

In order to construct a finite volume approximation of 2.5, the semi-infinite cylinder is approximated by a truncated cylinder $C_Y=\Omega\times\{0,Y\}$ with a sufficiently large Y. A uniform mesh Ω_h is introduced in Ω and anisotropic mesh $\omega_h=\{y_j=(j/M)^{\gamma}Y, j=0,\dots,M\}$ is used to compensate the singular behavior of the solution as $y\to 0$, where $\gamma>3/(2\beta)$ [72, 73]. In this way we obtain the mesh $C_{Y,h}=\Omega_h\times\omega_h$ for discretization of the extended problem.

Using the finite volume method and applying the standard notations of the finite differences [18, 19, 27, 101] we define the discrete problem, which approximates (2.5):

$$\begin{aligned}
&-\Big(y^{\alpha}_{j+1/2}\frac{\mathbf{U}_{h,j+1}-\mathbf{U}_{h,j}}{H_{j+1/2}}-y^{\alpha}_{j-1/2}\frac{\mathbf{U}_{h,j}-\mathbf{U}_{h,j-1}}{H_{j-1/2}}\Big) \\
&+\frac{y^{\alpha+1}_{j+1/2}-y^{\alpha+1}_{j-1/2}}{\alpha+1}L_h\mathbf{U}_h=0,\ (X_h,y_j)\in C_{Y,h}, \\
&-y^{\alpha}_{1/2}\frac{\mathbf{U}_{h,1}-\mathbf{U}_{h,0}}{H_{1/2}}+\frac{y^{\alpha+1}_{1/2}}{\alpha+1}L_h\mathbf{U}_h=d_{\beta}\mathbf{f}_h,\quad X_h\in\bar{\Omega}_h\times\{0\}, \\
&\mathbf{U}_h=0,\quad (X_h,y_j)\in\partial C_{Yh}\setminus\bar{\Omega}_h\times\{0\},
\end{aligned} \tag{2.6}$$

where $y_{j+1/2}=\frac{1}{2}\big(y_j+y_{j+1}\big)$, $H_{j+1/2}=y_{j+1}-y_j$.

2.4.2 Reduction to a Pseudo-Parabolic PDE Problem (M2)

The solution of the non-local problem (2.3) is sought as a mapping [57, 90, 91]:

$$U(x,t)=\big(t(\mathcal{L}-\delta I)+\delta I\big)^{-\beta}f, \tag{2.7}$$

where $\mathcal{L}\geq\delta_0 I$, $\delta=\gamma\delta_0$, $0<\gamma<1$. Thus, it follows from (2.7) that function $U(x,1)=\mathcal{L}^{-\beta}f$ defines the solution of the non-local problem (2.3). The

function U satisfies the evolutionary pseudo-parabolic problem

$$(tG + \delta I)\frac{\partial U}{\partial t} + \beta GU = 0, \quad 0 < t \le 1, \tag{2.8}$$
$$U(x,0) = \delta^{-\beta} f, \quad t = 0,$$

where operator $G = \mathcal{L} - \delta I$. We see a typical property of such transformations, when instead of the non-local problem (2.3) a local pseudo-parabolic problem (2.8) is solved (in higher dimension space $\mathbb{R}^{d+1}$).

We employ the geometrically graded time-stepping scheme [32]. Considering $K = \log(\lambda_M)$, we set a basic non uniform graded mesh: $t^0 = 0$ and $t^n = 2^{n-1-K}$ for $n = 1, \ldots, K+1$. Here, λ_M is the largest eigenvalue of discrete operator L_h. Then, each interval $[t^{n-1}, t^n]$ is partitioned into J subintervals with the points

$$t^{n-1,j} = t^{n-1} + j\tau_n,\ j = 0, \ldots, J,\ \ \tau_n = (t^n - t^{n-1})/J.$$

We denote the mesh points $t^{n,j}$ in the increasing order by t^k for $k = 0, \ldots, M$, where $M = (K+1)J$, and $\tau_n = \tau_{n(k)}$ is the time step for a given k. We approximate the problem (2.8) by the following Crank–Nicolson scheme [18, 19, 27]:

$$(t^{k-1/2}G_h + \delta I_h)\frac{U_h^k - U_h^{k-1}}{\tau_n} + \beta G_h U_h^{k-1/2} = 0, \qquad 1 \le k \le M, \tag{2.9}$$
$$U_h^0 = \delta^{-\beta} f_h,$$

where $G_h = L_h - \delta I_h$, $U_h^{k-1/2} = (U_h^k + U_h^{k-1})/2$ and $t^{k-1/2} = (t^{k-1} + t^k)/2$. It has been proven [32] that the time discretization error of such finite difference scheme is bounded by CJ^{-2}, that is, the discrete solution is converging with the second order in time with respect to J.

If τ_n is constant, then the mesh is uniform.

2.4.3 Integral Representation of the Solution of Initial Problem (2.3) M3

The third numerical method is based on an integral representation of the solution of non-local problem (2.3) using the local elliptic operators (cf. [12, 13]):

$$\mathcal{L}^{-\beta} = \frac{2\sin(\pi\beta)}{\pi}\int_0^\infty y^{2\beta-1}(I + y^2\mathcal{L})^{-1}dy. \tag{2.10}$$

Again, this problem is formally defined in a higher dimension space $\mathbb{R}^{d+1}$, and the integral kernel is singular with respect the extended dimension variable. To calculate integral in (2.10), three different numerical quadrature formulas are proposed in the literature, which we have used to develop three parallel numerical algorithms.

In the first algorithm [12, 13], the integral (2.10) is first transformed to a sum of two integrals

$$\mathcal{L}^{-\beta} = \frac{2\sin(\pi\beta)}{\pi}\Big[\int_0^1 y^{2\beta-1}(I+y^2\mathcal{L})^{-1}dy + \int_0^1 y^{1-2\beta}(y^2 I+\mathcal{L})^{-1}dy\Big]. \tag{2.11}$$

Then a quadrature scheme based on a graded partition of the integration interval $[0,1]$ is applied to resolve the singular behaviour of coefficient $y^{2\beta-1}$:

$$y_{1,j} = \begin{cases} (j/M)^{\frac{1}{2\beta}} & \text{if } 2\beta-1<0, \\ j/M & \text{if } 2\beta-1\geq 0, \end{cases} \qquad j=0,\ldots,M.$$

A similar partition is used to resolve the singularity of $y^{1-2\beta}$. The finite volume discrete operator L_h approximates the elliptic operator $\mathcal{L}$. Then integrals (2.11) to compute $L_h^{-\beta}\mathbf{f}_h$ are approximated as

$$\begin{aligned}\mathbf{U}_{h,1}^{-\beta}\mathbf{f}_h = \frac{2\sin(\pi\beta)}{\pi}\Big[&\sum_{j=1}^{M}\frac{y_{1,j}^{2\beta}-y_{1,j-1}^{2\beta}}{2\beta}\big(I_h+y_{1,j-1/2}^2 L_h\big)^{-1}\mathbf{f}_h \\ &+\sum_{j=1}^{M}\frac{y_{2,j}^{2-2\beta}-y_{2,j-1}^{2-2\beta}}{2-2\beta}\big(y_{2,j-1/2}^2 I_h+L_h\big)^{-1}\mathbf{f}_h\Big].\end{aligned} \tag{2.12}$$

The second quadrature algorithm is defined on the uniform grid points $y_j = jh_y$ and $h_y = 1/\sqrt{M}$ [12, 13]:

$$\mathbf{U}_{h,2}^{-\beta}\mathbf{f}_h = \frac{2h_y\sin(\pi\beta)}{\pi}\sum_{j=-M}^{M} e^{2\beta y_j}\left(I_h+e^{2y_j}L_h\right)^{-1}\mathbf{f}_h. \tag{2.13}$$

It provides an exponential convergence to (2.10). We note that for both quadrature algorithms (2.12) and (2.13) all $2M$ local discrete elliptic problems can be solved independently.

The third quadrature algorithm is defined as in [12, 13]:

$$\mathbf{U}_{h,3}^{-\beta}\mathbf{f}_h = \frac{2k\sin(\pi\beta)}{\pi}\sum_{j=-m_1}^{m_2} e^{2(\beta-1)jk}\left(L_h + e^{-2jk}I_h\right)^{-1}\mathbf{f}_h, \tag{2.14}$$

where $m_1 = \lceil \pi^2/(4\beta k^2)\rceil$ and $m_2 = \lceil \pi^2/(4(1-\beta)k^2)\rceil$. It provides an exponential convergence to (2.10). The parameter $k > 0$ controls the accuracy of the approximation of integral and the number of local elliptic problems that need to be solved.

2.4.4 Approximation of the Solution of Problem (2.3) using Rational Approximations M4

The fourth algorithm is defined by using a different approach. Instead of transforming the non-local problem (2.3) to a locally defined classical PDE in a higher dimension space, the method is based on the best uniform rational approximations of function $t^{1-\beta}$ for BURA [45] and of function t^{β} for R-BURA [44] in the interval $[0, 1]$. The approximate solution U_h of the discrete problem $L_h^{\beta}U_h = f_h$ is defined as

$$U_h = c_0 A_h^{-1}\tilde{f}_h + \sum_{j=1}^{m} c_i (A_h - d_j I)^{-1}\tilde{\mathbf{f}}_h, \tag{2.15}$$

where the matrix A_h and the right-hand side function $\tilde{f}_h$ are scaled as $A_h = h^2/lL_h$ and $\tilde{\mathbf{f}}_h = (h^2/l)^{\beta} f_h$ ($l = 8$ for 2D problem, $l = 12$ for 3D problem), respectively.

The coefficients c_j and d_j for the BURA method are obtained by solving the global optimization problem to find the best uniform rational approximation $r_m^*(t)$ of function $t^{1-\beta}$ [45]:

$$r_m(t) = c_0 + \sum_{j=1}^{m}\frac{c_j t}{t - d_j},$$
$$\min_{r_m}\max_{t\in[0,1]}\left|t^{1-\beta} - r_m(t)\right| = \max_{t\in[0,1]}\left|t^{1-\beta} - r_m^*(t)\right| =: \varepsilon_m(\beta).$$

In recent research [45] the efficiently modified Remez algorithm is proposed to compute the coefficients c_j and d_j. Moreover, the coefficients for $5 \leq m \leq 7$ and $\beta = \{0.25, 0.5, 0.75\}$ are provided. In the another study, [92] the coefficients and errors $\varepsilon_m(\beta)$ are computed with high accuracy for $\beta = \{i/8,\ i = 1, \ldots, 7\}$ and $m \leq 30$. The corresponding coefficients, however, are not provided.

2.5 Parallel Algorithms

We consider the parallelisation of all numerical solution algorithms presented in Section 2.4. In order to implement the developed parallel algorithms, we use the two-level parallel programming templates [26]. The efficient parallel multigrid solvers from HYPRE numerical library [34, 35] are applied as preconditioners in the parallel conjugate gradient method. A similar approach was used also in [23, 24]. On the first level, we define a set of discrete problems, which can be solved independently in parallel and these tasks are statically or dynamically distributed among processors. On the second level, each discrete problem is solved by using the domain decomposition method and a specified parallel linear system solver based on preconditioned CG method. Two types of multigrid solvers from HYPRE numerical library are used as preconditioners in the parallel conjugate gradient method. To study the performance of considered parallel numerical algorithms on structured grids, we use the geometric multigrid solver PFMG. To estimate their performance on general non-structured grids suitable for general domains, we use the algebraic multigrid solver BoomerAMG.

We have compared selected numerical algorithms in terms of accuracy and computational costs using the following test problem:

$$\mathcal{L}^{\beta} u = f(x), \quad x \in \Omega = (0,1)^d, \quad u(x) = 0, \; x \in \partial\Omega, \tag{2.16}$$

with the Laplace operator $\mathcal{L} = -\Delta$, $\beta = 0.25, 0.75$, d – number of dimension, where the right-hand-side f is the well-known checkerboard function (see [13, 45]):

$$f(x) = \begin{cases} 1, & \text{if } \Pi_{i=1}^{d} (x_i - 0.5)^d > 0; \\ -1, & \text{otherwise.} \end{cases} \tag{2.17}$$

Similar test problems (only in 2D domains) with checkerboard right-hand-side functions are used to demonstrate the convergence of the selected numerical methods in many papers: (M2) [32], (M3) [13], (M4) [44,45]. This test demonstrates the important feature of the fractional diffusion problems, how the convergence rate of numerical discretisation methods depends on the smoothness of the solution. The checkerboard right-hand-side function f is leading to the well expressed boundary layers in the solution. The regularity of this solution is decreasing with the decreasing fractional power parameter β, what is reducing the rate of convergence of discrete solutions. A popular approach to resolve such problems is to use adaptive meshes [2].

We restrict to uniform space grids to compare the accuracy of the selected numerical methods.

In order to analyse and compare the accuracy of the numerical solutions, we compute the reference (exact) solutions U_N^F of our test problem (2.16)–(2.17). We use as reference (exact) solutions U_N^F, the numerical solutions obtained via Fourier algorithm on uniform space grid with $N = 2^{12} = 4096$ $N = 2^{15} = 32768$ (for $2D$ case) or $N = 2^{12} = 4096$ (for $3D$ case) discretisation points in each direction. In the $3D$ case, such a fine grid requires over 1024 GB of memory for a double precision. Thus we use the parallel version of our Fourier solver to compute the reference (exact) solutions, where computations are done on 32 nodes with 64 GB of memory per node.

The Fourier method is very fast when the FFT algorithm can be applied (constant coefficients of the elliptic operator and rectangular parallelepiped Ω). In such cases the computational complexity of solving fractional power elliptic problems is the same as for solving classical elliptic problems. However, we are interested in more general methods, which can be applied in general domains for general elliptic operators.

We use the relative errors to report and analyse the accuracy of obtained numerical solutions. The relative error E_N^{M*} of the numerical solution U_N^{M*} obtained by method M* on the uniform space grid with N points in each direction is defined as follows:

$$E_N^{M*} = \frac{\|u - U_N^{M*}\|_\infty}{\|u\|_\infty}, \tag{2.18}$$

where u denotes the reference (exact) solution obtained by the Fourier method. In the case for $2D$ problem and $\beta = 0.25$, the maximal absolute value of the solution is $\|u\|_\infty = 0.3904$; for $3D$ problem and $\beta = 0.25$, the maximal absolute value of the solution is $\|u\|_\infty = 0.4097$, for $\beta = 0.75$, $\|u\|_\infty = 0.0672$. The reference relative error values are shown in Table 2.1–2.2 for solutions with $\beta = 0.25$ or $\beta = 0.75$.

To solve the arising linear systems, we use the preconditioned conjugate gradient (PCG) method with a geometric multigrid preconditioner from the HYPRE library and we set tolerance to 10^{-8} in all tests.

Table 2.1: Relative error E_N^F (2.18) of Fourier solution U_N^F of 2D test problem (2.16)–(2.17)

Mesh size, N^2	128^2	256^2	512^2	1024^2	2048^2
E_N^F for $\beta = 0.25$	0.00946	0.00669	0.00473	0.00334	0.00236

Table 2.2: Relative error E_N^F (2.18) of Fourier solution U_N^F of 3D test problem (2.16)–(2.17)

Mesh size, N^3	16^3	32^3	64^3	128^3	256^3	512^3
E_N^F for $\beta = 0.25$	0.035654	0.025169	0.017792	0.012569	0.008855	0.006171
E_N^F for $\beta = 0.75$	0.012563	0.004399	0.001554	0.000549	0.000193	0.000067

In accordance with the theory (2.4), due to the singularity of the solution the reduced convergence rate $O(h^{2\beta})$ is observed. For $\beta = 0.25$ the error is reduced 2 times when the space step is reduced 4 times, and for $\beta = 0.75$ the error is reduced 8 times.

All tests were performed on the "Avitohol" cluster at the Institute of Information and Communication Technologies (IICT) of the Bulgarian Academy of Sciences The cluster consists of 150 HP Cluster Platform SL250S GEN8 servers. Each computational node has 2 Intel® Xeon® processors E5-2650v2 @ 2.6GHz (8 cores each), 64GB RAM and 2 Intel® Xeon® Phi 7120P coprocessors. The computational nodes are interconnected via fully non-blocking 56Gbps FDR InfiniBand network. Up to 64 nodes (1024 cores) were used in our parallel tests.

The parallel performance analysis considers both strong and weak scalability of the developed parallel algorithms. The strong scaling shows the efficiency of parallel algorithm solving the problem of a fixed size, increasing the number of parallel processes. This is important in situations when the solution time should be reduced as much as possible, for example, for time-critical applications or for solving optimization problems.

Nowadays, it is often more important to solve larger problems in a reasonable time. The weak scaling demonstrates the ability of parallel algorithm to solve the problem of increasing size with a proportionally increased number of parallel processes in a similar time.

Both 2D and 3D are computationally costly. Thus, parallel computation applicable in both cases. However it is easier to perform analysis for 2D problem. Moreover some of methods (M1) have some difficulties in 3D case, so in order to evaluate effectiveness we need to restrict 2D problems.

The aim of this chapter is to construct efficient parallel algorithms for methods (M1)–(M4). First of all, we investigate the weak and strong scalability of the developed parallel algorithms and compare their parallel performance for 2D problems. Based on the results for 2D problems, we made several changes to analyse parallel algorithms for 3D problems:

- The method (M1) was excluded from the list of considered methods as less suitable for 3D case due to high memory requirements. However,

authors [12] have presented a new version of method (M1). This version makes method (M1) competitive comparing to others methods.

- For the pseudo-parabolic method (M2), we use the recently proposed geometrically graded time stepping scheme [32], which should resolve the singular behavior of the solution for pseudo-time t close to 0 and give better convergence results compared to the uniform time stepping scheme used for 2D problem.

- For the quadrature method (M3), we employ a quadrature formula [12] with geometrically graded mesh and show superior convergence results due to this new formula.

- For analysis of the BURA method (M4) [45] we include its recent modification R-BURA [44] which has been recommended for the power coefficient closer to 1.

We compare the developed algorithms in terms of parallel solution time, that parallel solution times of the developed algorithms that are required to achieve the specified accuracy of the solution.

2.5.1 Convergence and Scalability Analysis of the Parallel Algorithms for 2D Problems

Extension to a Mixed Boundary Value Problem (M1)

The finite volume discretisation scheme (2.6) leads to a large system of linear equations. In the case of two-dimensional problem domain Ω, one has to solve a system with 7-point stencil of size $N_{x_1} \times N_{x_2} \times M$.

One standard approach for parallel solution of such problems is the domain decomposition method [77]. The discrete mesh of the problem domain and its associated fields are partitioned into sub-domains, which are allocated to different processes. Note that in our case, the discrete mesh $C_{Y,h}$ of the truncated cylinder $C_Y = \Omega \times \{0, Y\}$ has to be partitioned.

We have used the parallel algebraic multigrid solver BoomerAMG from the well-known HYPRE numerical library [34, 35] as a preconditioner for the parallel conjugate gradient method. The default parameter settings of BoomerAMG are applied, performing only a small tuning to adapt them to our problems. It is clear that the default parameters are optimized for the isotropic problems, thus there remains a potential possibility to improve the scalability of the solvers.

The accuracy and solution times of the parallel elliptic solver are shown in Table 2.3. Second order of convergence is observed with respect to the extended variable y and discrete error values from Table 2.1 are attained. Serial solution times T_1 are shown for the discrete problems that fit into single node's memory. Minimal parallel solution times T_p with optimal decomposition of the discrete mesh and corresponding speed-ups S_p are demonstrated using up to 512 parallel processes.

Table 2.3: The accuracy and solution times of the parallel elliptic solver on uniform space grid $N \times N$

N	M	Error	T_1	p	T_p	S_p
128	128	0.00985	13.03	48	1.02	12.80
128	256	0.00956	25.95	48	2.03	12.78
256	128	0.00703	56.36	64	3.29	17.11
256	256	0.00678	122.93	128	5.10	24.10
512	128	0.00503	236.95	256	6.86	34.56
512	256	0.00481	586.61	256	10.57	55.52
1024	128	0.00360		512	14.85	
1024	256	0.00341		512	29.36	
2048	128	0.00258		512	53.95	
2048	256	0.00241		512	144.19	

We investigate the strong scaling of the developed parallel solver for a fixed problem size $N_{x_1} = N_{x_2} = 400$ and $M = 400$. Parallel performance results of these tests are presented in Table 2.4. Here $p = n_d \times n_c$ is the number of used parallel processes, corresponding to computing with n_d nodes and n_c cores per node. Here, $P_1 \times P_2 \times P_3$ defines the topology of partitioning, while $\mathrm{DOF}/p = N_{x_1} \times N_{x_2} \times M/p$ shows the degrees of freedom per process, i.e. the number of unknowns per core. The total wall time T_p is given in seconds. The AMG preconditioning includes the recursive multilevel factorization, where the coarse grid matrices are computed. The cost of this task can be quite large. Thus we show the BoomerAMG setup time T_{set}, and the solution time T_{sol} of of the parallel conjugate gradient solver with N_{iter} iterations. Finally, we present the obtained values of parallel speed-up $S_p = T_1/T_p$, and parallel efficiency $E_p = S_p/p$.

The parallel algebraic multigrid preconditioner is robust, and the number of iterations is stable at about 20-22, for different numbers of the utilized processes and topologies of grid partitioning. The optimal partitioning topology is 2D and the anisotropic y coordinate should be kept in one process.

The solution time T_{sol} scales quite well for smaller number of processes.

Table 2.4: Total wall time T_p, speed-up S_p, and efficiency E_p, solving the problem (2.5) with $N_{x_1} = N_{x_2} = 400$, $M = 400$, $\beta = 0.25$.

p	$n_d \times n_c$	$P_1 \times P_2 \times P_3$	DOF/p	T_p	T_{set}	T_{sol}	N_{iter}	S_p	E_p
1	1×1	$1 \times 1 \times 1$	$64 \cdot 10^6$	557.0	161.1	360.2	21	1.00	1.00
2	1×2	$2 \times 1 \times 1$	$32 \cdot 10^6$	1455.1	1265.0	172.1	21	0.38	0.19
2	1×2	$1 \times 2 \times 1$	$32 \cdot 10^6$	1463.5	1275.2	170.5	21	0.38	0.19
2	1×2	$1 \times 1 \times 2$	$32 \cdot 10^6$	4561.7	4330.1	213.3	22	0.12	0.06
4	1×4	$4 \times 1 \times 1$	$16 \cdot 10^6$	1189.0	1097.0	82.5	19	0.47	0.12
4	1×4	$2 \times 2 \times 1$	$16 \cdot 10^6$	730.1	635.7	85.2	20	0.76	0.19
4	1×4	$1 \times 2 \times 2$	$16 \cdot 10^6$	1524.4	1404.6	110.2	22	0.37	0.09
8	1×8	$8 \times 1 \times 1$	$8 \cdot 10^6$	617.5	562.1	49.9	21	0.90	0.11
8	1×8	$2 \times 4 \times 1$	$8 \cdot 10^6$	416.1	362.4	48.4	21	1.34	0.17
8	1×8	$2 \times 2 \times 2$	$8 \cdot 10^6$	543.8	479.9	58.5	21	1.02	0.13
16	1×16	$16 \times 1 \times 1$	$4 \cdot 10^6$	391.1	351.8	35.2	21	1.42	0.09
16	1×16	$4 \times 4 \times 1$	$4 \cdot 10^6$	184.5	146.7	33.9	21	3.02	0.19
16	1×16	$4 \times 2 \times 2$	$4 \cdot 10^6$	270.7	223.4	43.5	22	2.06	0.13

The setup costs of the parallel BoomerAMG preconditioner are relatively large. We note that a big increase of the setup complexity and time takes place when a parallel version of multigrid preconditioner is started to be used ($p = 2$). Later, the setup time T_{set} scales quite well. Thus, in order to investigate further the strong scalability of our parallel algorithm, we compute the speed-up and the efficiency coefficients with respect to the solution time obtained on one node with 16 cores.

Strong scaling results for larger numbers of processes are shown in Table 2.5. Here, $S_{n_d} = T_{1\times16}/T_{n_d\times16}$ is the parallel speed-up on n_d nodes with respect to one node, $E_{n_d} = S_{n_d}/n_d$ is the parallel efficiency.

Table 2.5: Total wall time T_p, speed-up S_{n_d}, and efficiency E_{n_d}, solving the problem (2.5) with $N_{x_1} = N_{x_2} = 400$, $M = 400$, $\beta = 0.25$.

p	$n_d \times n_c$	$P_1 \times P_2 \times P_3$	DOF/p	T_p	T_{set}	T_{sol}	N_{iter}	S_{n_d}	E_{n_d}
16	1×16	$4 \times 4 \times 1$	$4 \cdot 10^6$	184.5	146.7	33.9	21	1.00	1.00
16	2×8	$4 \times 4 \times 1$	$4 \cdot 10^6$	173.4	146.0	24.6	21	1.06	0.53
32	2×16	$4 \times 8 \times 1$	$2 \cdot 10^6$	78.1	58.3	17.8	22	2.36	1.18
32	2×16	$4 \times 4 \times 2$	$2 \cdot 10^6$	109.6	86.0	21.6	22	1.68	0.84
64	4×16	$8 \times 8 \times 1$	$1 \cdot 10^6$	31.9	21.9	8.9	21	5.78	1.45
64	4×16	$4 \times 4 \times 4$	$1 \cdot 10^6$	44.4	30.7	12.6	22	4.15	1.04
128	8×16	$8 \times 16 \times 1$	$5 \cdot 10^5$	17.8	11.6	4.9	22	10.38	1.30
128	8×16	$8 \times 4 \times 4$	$5 \cdot 10^5$	21.5	13.0	7.0	24	8.59	1.07
256	16×16	$16 \times 16 \times 1$	$2.5 \cdot 10^5$	11.9	7.2	3.0	23	15.49	0.97
256	16×16	$8 \times 8 \times 4$	$2.5 \cdot 10^5$	12.3	7.3	3.6	23	14.95	0.93
512	32×16	$16 \times 32 \times 1$	$1.25 \cdot 10^5$	12.4	8.3	2.2	23	14.83	0.46
512	32×16	$8 \times 8 \times 8$	$1.25 \cdot 10^5$	12.1	9.5	2.2	25	15.19	0.47

The parallel solution time with all 16 cores of a single node is similar to the solution time with two nodes using only 8 cores each (see Table 2.5) Our

parallel application is certainly memory bound, as most of the time is spent in operations with stored sparse matrices. However further performance degradation due to the memory saturation increasing the number of utilized cores from 8 to 16 per node appears to be not significant, being comparable with the additional overhead of the inter-node communications. Therefore, we have always used all 16 cores in our multi-node parallel tests.

The parallel preconditioner setup time T_{set} scales even over-linearly. Thus we have obtained efficiencies $E_{n_d} > 1$ in some cases. It follows from the presented results, that up to 256 processes can be used efficiently for this size of the discrete problem.

In these parallel performance tests, the number of grid points per process is kept constant: $\mathrm{DOF}/p = 4 \cdot 10^6$. This also means that the number of grid points per node is constant. The total number of grid points was increased from $6.4 \cdot 10^7$ up to $4.096 \cdot 10^9$ on the largest grid, while the number of processes was increased from 16 to 1024. The results of computational experiments are presented in Table 2.6.

The efficiency of the parallel algorithm is defined as $E_{n_d} = T_{1\times16}/T_{n_d\times16}$. As it follows from the presented results, the number of iterations is slightly increasing when the size of the problem $N = N_{x_1} \times N_{x_2} \times M$ is increased. In order to estimate the performance of parallel algorithm excluding this factor, we also calculate and show in Table 2.6 the scaled parallel efficiency regarding the time per iteration

$$\widehat{E}_{n_d} = \frac{N_{iter}(n_d)}{N_{iter}(1)} E_{n_d}.$$

Table 2.6: Weak scalability of the parallel algorithm solving problem (2.5) with $\beta = 0.25$.

p	$n_d \times n_c$	$P_1 \times P_2 \times P_3$	$N_{x_1} \times N_{x_2} \times M$	T_p	T_{set}	T_{sol}	N_{iter}	E_{n_d}	$\widehat{E}_{n_d}$
16	1×16	$4 \times 4 \times 1$	$400^3 = 64 \cdot 10^6$	184.5	146.7	33.9	21	1.00	1.00
32	2×16	$8 \times 4 \times 1$	$504^3 \approx 128 \cdot 10^6$	241.8	200.4	37.4	23	0.76	0.84
32	2×16	$4 \times 4 \times 2$	$504^3 \approx 128 \cdot 10^6$	320.5	271.4	45.2	23	0.58	0.63
64	4×16	$8 \times 8 \times 1$	$635^3 \approx 256 \cdot 10^6$	276.6	232.9	39.5	24	0.67	0.76
64	4×16	$4 \times 4 \times 4$	$635^3 \approx 256 \cdot 10^6$	394.8	335.6	55.1	24	0.47	0.53
128	8×16	$8 \times 16 \times 1$	$800^3 = 512 \cdot 10^6$	260.8	225.0	32.2	26	0.71	0.88
128	8×16	$8 \times 4 \times 4$	$800^3 = 512 \cdot 10^6$	365.0	300.0	60.1	26	0.51	0.63
256	16×16	$16 \times 16 \times 1$	$1008^3 \approx 1024 \cdot 10^6$	332.8	281.1	46.2	27	0.55	0.71
256	16×16	$8 \times 8 \times 4$	$1008^3 \approx 1024 \cdot 10^6$	404.9	335.5	64.3	28	0.46	0.61
512	32×16	$16 \times 32 \times 1$	$1270^3 \approx 2048 \cdot 10^6$	411.7	353.6	51.8	30	0.45	0.64
512	32×16	$8 \times 8 \times 8$	$1270^3 \approx 2048 \cdot 10^6$	486.2	409.2	72.5	31	0.38	0.56
1024	64×16	$32 \times 32 \times 1$	$1600^3 = 4096 \cdot 10^6$	394.9	315.4	69.3	32	0.47	0.71

The time per iteration scales well for our parallel solver up to 1024

processes. The 2D partitioning of the problem domain is producing better performance results, since the significantly higher setup costs for the partitioning of anisotropic coordinate y are not compensated by the better geometric quality of 3D partitioning. Although, this difference in computation time is decreasing. The test with $16 \times 8 \times 8$ partitioning is missing in Table 2.6 since it does not fit into 64GB of memory available on the nodes. Memory requirements of our elliptic problem solver, mostly of BoomerAMG preconditioner, are growing very fast.

Pseudo-Parabolic Method

The constructed finite volume scheme (2.9) implies that this numerical algorithm will advance in pseudo-time computing $\mathbf{U}_h^n$ from $\mathbf{U}_h^{n-1}$, solving one system of linear equations at each of M time steps. These systems are solved sequentially, one after another. Thus, the pseudo-parabolic numerical algorithm does not allow parallelism in the introduced new pseudo-time dimension. The usage of the proposed general two-level parallelisation template is reduced only to the second level, at which discrete subproblems are solved in parallel. In the case, when the problem domain Ω is two-dimensional, the linear system will have 5-point stencil matrix, three-dimensional – 7-point stencil matrix. We use a standard domain decomposition method for the parallel solution of pseudo-parabolic PDE problem (2.8). The discrete mesh of problem domain Ω and its associated fields are partitioned into sub-domains, which are allocated to different processes.

Table 2.7: The accuracy and solution times of the parallel pseudo-parabolic solver on uniform space grid $N \times N$

N	M	Error	T_1	p	T_p	S_p
128	592	0.00939	7.37	16	4.46	1.65
256	2348	0.00676	102.76	16	26.29	3.91
512	9216	0.00498	1252.90	48	153.36	8.17
1024	36864	0.00353	24712.00	96	950.42	26.00

The errors and solution times of the parallel pseudo-parabolic solver are shown in Table 2.7. We see that the error values of the discrete solution from Table 2.1 are eventually obtained by increasing the number of time steps M. However, the number of time steps M for the uniform time grid should be taken quite large. It makes this method (M2) non-competitive. However this part of the algorithm was improved with the recently proposed geometrically graded time stepping scheme [32], which resolved the singular

behavior of the solution for pseudo-time t close to 0. The results of methods (M2) with geometrically graded time stepping scheme will be presented in Section 2.5.2.

One can easily see the similarities and differences between the (M1) and (M2) approaches. One of the important practical implications is the significantly smaller amount of memory required for the pseudo-parabolic algorithm – $O(N^2)$, compared to the elliptic approach – $O(N^2M)$. On the other hand, the parallel pseudo-parabolic algorithm does not have parallelism in the introduced additional dimension, i.e. in the pseudo-time. This structure of the algorithm poses restrictions on the size of the tasks that can be solved in parallel, thus leading to smaller sub-problems that are assigned to each of the parallel processes.

Parallel performance results on strong scaling of the developed pseudo-parabolic solver are presented in Table 2.8, where $S_p = T_1/T_p$ is the parallel speed-up, $E_p = S_p/p$ is the parallel efficiency. We show the degree of freedom per process, i.e. the number of unknowns in linear solver per core, $\mathrm{DOF}/p = N_{x_1} \times N_{x_2}/p$. We also calculate and present the scaled parallel efficiency for the time per iteration

$$\widehat{E}_p = \frac{N_{iter}(p)}{N_{iter}(1)} E_p,$$

where $N_{iter}(p)$ is the total number of iterations of parallel preconditioned conjugate gradient solver, i.e. summed up for all time steps.

For the pseudo-parabolic problem the additional setup costs of parallel BoomerAMG preconditioner are not so large as for the elliptic problem solver (see in Table 2.8). This is a consequence from the much better structure of the matrix for the discretised pseudo-parabolic problem (2.9). It includes not only a diffusion term but also a positive diagonal matrix.

The parallel multigrid preconditioner is robust for the solved problem, i.e. the number of iterations is quite stable. It increases only slightly with the number of parallel processes.

As it is expected from the standard volume and area analysis, the 2D partitioning strategy is more efficient than the 1D one, because it reduces the data transfer between the parallel processes.

The results of strong scaling for different problem sizes are shown in Figure 2.1.

In accordance with the parallel computing theory, the efficiency of strong scaling is increasing when increasing the problem size. The most interesting performance results are observed for the largest size problems. Due to the

Table 2.8: Total wall time T_p, speed-up S_p, and efficiency E_p solving problem (2.8) with $\beta = 0.25$.

$N_{x_1} \times N_{x_2},\ M$	p	$n_d \times n_c$	$P_1 \times P_2$	DOF/p	T_p	N_{iter}	S_p	E_p	$\widehat{E}_p$
400^2, 400	1	1×1	1×1	$16 \cdot 10^4$	147.5	1738	1.00	1.00	1.00
400^2, 400	2	1×2	2×1	$8 \cdot 10^4$	84.8	1764	1.74	0.87	0.88
400^2, 400	4	1×4	4×1	$4 \cdot 10^4$	51.9	1789	2.84	0.71	0.73
400^2, 400	4	1×4	2×2	$4 \cdot 10^4$	46.7	1899	3.16	0.79	0.86
400^2, 400	16	1×16	16×1	$1 \cdot 10^4$	20.1	1976	7.32	0.46	0.52
400^2, 400	16	1×16	4×4	$1 \cdot 10^4$	16.2	1956	9.13	0.57	0.64
400^2, 400	32	2×16	4×8	$0.5 \cdot 10^4$	10.7	2022	13.80	0.43	0.50
400^2, 400	64	4×16	8×8	$0.25 \cdot 10^4$	12.2	2060	12.10	0.19	0.22
800^2, 800	1	1×1	1×1	$64 \cdot 10^4$	1230.3	3338	1.00	1.00	1.00
800^2, 800	2	1×2	2×1	$32 \cdot 10^4$	729.2	3352	1.69	0.84	0.85
800^2, 800	4	1×4	4×1	$16 \cdot 10^4$	435.8	3370	2.82	0.71	0.71
800^2, 800	4	1×4	2×2	$16 \cdot 10^4$	382.0	3363	3.22	0.81	0.81
800^2, 800	16	1×16	16×1	$4 \cdot 10^4$	139.5	3430	8.82	0.55	0.57
800^2, 800	16	1×16	4×4	$4 \cdot 10^4$	119.7	3436	10.28	0.64	0.66
800^2, 800	32	2×16	32×1	$2 \cdot 10^4$	90.1	3603	13.65	0.43	0.46
800^2, 800	32	2×16	4×8	$2 \cdot 10^4$	63.8	3466	19.28	0.60	0.63
800^2, 800	64	4×16	8×8	$1 \cdot 10^4$	44.4	3589	27.74	0.43	0.47
800^2, 800	128	8×16	8×16	$0.5 \cdot 10^4$	38.0	3758	32.39	0.25	0.29
800^2, 800	256	16×16	16×16	$0.25 \cdot 10^4$	48.2	3892	25.50	0.10	0.12
1600^2, 1600	1	1×1	1×1	$256 \cdot 10^4$	10828.0	6538	1.00	1.00	1.00
1600^2, 1600	2	1×2	2×1	$128 \cdot 10^4$	7007.4	6547	1.55	0.77	0.77
1600^2, 1600	4	1×4	4×1	$64 \cdot 10^4$	4272.2	6558	2.53	0.63	0.64
1600^2, 1600	4	1×4	2×2	$64 \cdot 10^4$	3559.2	6558	3.04	0.76	0.76
1600^2, 1600	16	1×16	16×1	$16 \cdot 10^4$	1276.9	6602	8.48	0.53	0.54
1600^2, 1600	16	1×16	4×4	$16 \cdot 10^4$	1056.7	6595	10.25	0.64	0.65
1600^2, 1600	32	2×16	32×1	$8 \cdot 10^4$	680.9	6635	15.90	0.50	0.50
1600^2, 1600	32	2×16	4×8	$8 \cdot 10^4$	520.9	6624	20.79	0.65	0.66
1600^2, 1600	64	4×16	8×8	$4 \cdot 10^4$	266.8	6670	40.58	0.63	0.65
1600^2, 1600	128	8×16	8×16	$2 \cdot 10^4$	162.0	6728	66.84	0.52	0.54
1600^2, 1600	256	16×16	16×16	$1 \cdot 10^4$	138.4	6805	78.26	0.31	0.32
1600^2, 1600	512	32×16	16×32	$0.5 \cdot 10^4$	149.1	6936	72.64	0.14	0.15
1600^2, 1600	512	64×8	16×32	$0.5 \cdot 10^4$	119.3	6935	90.75	0.18	0.19
1600^2, 1600	1024	64×16	32×32	$0.25 \cdot 10^4$	210.7	7037	51.39	0.05	0.05

over-linear speed-up of the setup costs of BoomerAMG solver, and due to the better utilization of cores for the smaller size problems, the efficiency of the parallel solver is significantly bigger than 100% with respect to the performance of one node with 16 cores.

The size of the discrete problem was defined as $N_{x_1} \times N_{x_2} \times M$. The numbers of grid points and the number of time steps were selected to keep constant the size of each subproblem per process. Thus if the number of processes was increased 8 times, the size of the linear system was increased only 4 times. Therefore, some degradation of the efficiency of the parallel algorithm is expected when the number of processes is increased, in accordance with the previously observed results of the strong scaling. This theoretical

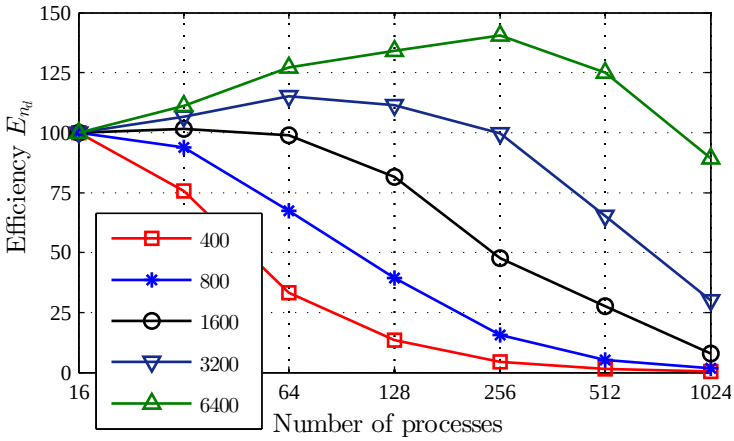


Figure 2.1: Strong scalability of the parallel pseudo-parabolic solver

conclusion agrees well with the experimental results in Figure 2.2, where the parallel efficiency E_{n_d} is shown. However, the weak scaling of the algorithm improves vastly with the increased size of the problem.

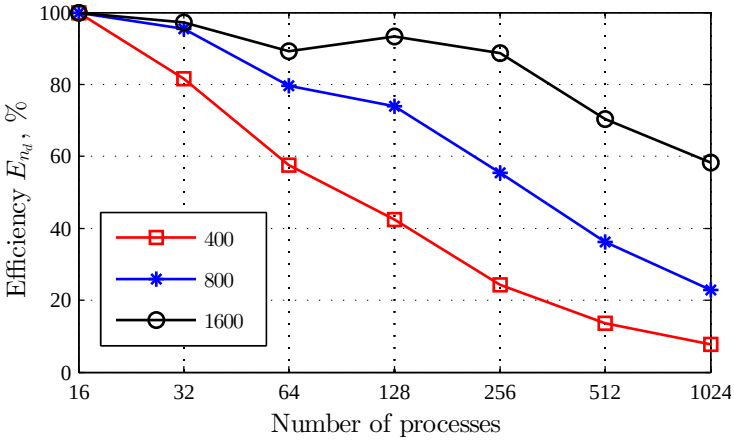


Figure 2.2: Weak scalability of the parallel pseudo-parabolic solver

Integral Representation of the Solution (M3)

Using the third approach described in Section 2.4.3, the non-local fractional diffusion problem (2.16) is transformed into a computation of two integrals (2.11). Each term in both sums of the numerical approximation (2.12) can be computed independently, what is very convenient for the parallelization.

We employ the well-known Master-Slave parallel model [26, 54]. Master process generates and distributes tasks (a block of consecutive y_j values) between the slave processes. For each received y_j value, a slave process

solves the local elliptic problem

$$(I_h + y_j^2 L_h)^{-1} f \quad \text{or} \quad (y_j^2 I_h + L_h)^{-1} f$$

in Ω_h. This is the first level of parallelization. When each independent elliptic problem is solved sequentially, one-level Master-Slave solver is obtained. For large size grids Ω_h, these sub-problems need to be solved in parallel in the two-level parallel solver. The parallel performance results are obtained with one-level Master-Slave solver using the conjugate gradient method with the sequential version of BoomerAMG preconditioner.

Differently from the usual Master-Slave model, in our solver, slave processes do not return to the master results of each task immediately after its solution. The slave processes accumulate the obtained results - compute partial sums of the solution u for each mesh point. These big data vectors of the size $N_{x_1} \times N_{x_2}$ are sent only once, after the solution of the last task. The final solution u is collected from the partial sums at the master process by MPI reduction operation [68].

The main challenge for this parallel algorithm is to guarantee a good load balancing and to minimize the data communication and synchronization costs. Thus we have implemented different task partitioning and distribution approaches, including dynamic and static cyclic algorithms. A single task is defined as a block of K consecutive y_j values. We have used $K = 1$ and $K = 10$ in the presented tests.

Parallel performance results on strong scaling of the developed parallel solver are presented in Table 2.9. The total wall time T_p is given in seconds. Here $p = n_d \times n_c$ is the total number of the used parallel processes using n_d nodes and n_c cores per node, $s = p - 1$ is the number of slave processes, which are solving the computational tasks. In Table 2.9, we also present the obtained parallel speed-up $S_s = T_2/T_p$, the efficiency $E_s = S_s/s$, and the parallel efficiency with respect to one node with 16 cores.

Both task partitioning algorithms (dynamic and static cyclic) are achieving very similar load balancing. This fact is a consequence from the robustness of BoomerAMG preconditioner, since the numbers of iterations and solution times are almost constant with respect to the parameters y_j.

The good strong scaling of the presented parallel algorithm is preserved even for large numbers of processes, if the size of task pool is sufficiently big. Results of strong scaling for increasing problem sizes are shown in Figure 2.3.

As it follows from the results obtained with the block size $K = 10$, a

Table 2.9: Total wall time T_p, speed-up S_p, and efficiency E_p solving problem (2.10) with $\beta = 0.25$.

$N_{x_1} \times N_{x_2}$, M	s	$n_d \times n_c$	Partition	Block	T_p	S_s	E_s	E_{n_d}
400^2, 400	1	1×2	cyclic	1	294.1	1.00	1.00	-
400^2, 400	2	1×3	cyclic	1	150.1	1.96	0.98	-
400^2, 400	15	1×16	cyclic	1	22.5	13.08	0.87	1.00
400^2, 400	31	2×16	cyclic	1	11.8	24.97	0.81	0.95
400^2, 400	63	4×16	cyclic	1	6.1	48.45	0.77	0.93
400^2, 400	127	8×16	cyclic	1	3.4	87.70	0.69	0.84
400^2, 400	255	16×16	cyclic	1	2.3	129.98	0.51	0.62
400^2, 400	511	32×16	cyclic	1	2.5	117.85	0.23	0.28
400^2, 400	15	1×16	dynamic	1	22.8	13.08	0.87	1.00
400^2, 400	31	2×16	dynamic	1	11.6	25.62	0.83	0.98
400^2, 400	63	4×16	dynamic	1	6.1	48.66	0.77	0.93
400^2, 400	127	8×16	dynamic	1	4.0	75.22	0.59	0.72
400^2, 400	255	16×16	dynamic	1	3.9	75.58	0.30	0.36
400^2, 400	511	32×16	dynamic	1	4.1	72.56	0.14	0.17
400^2, 400	15	1×16	dynamic	10	24.6	11.92	0.79	1.00
400^2, 400	31	2×16	dynamic	10	13.1	22.48	0.73	0.94
400^2, 400	63	4×16	dynamic	10	8.8	33.36	0.53	0.70

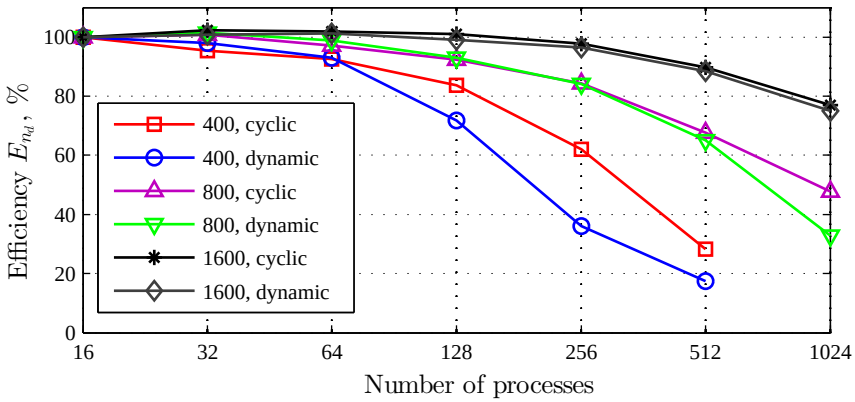


Figure 2.3: Strong scaling of the parallel integral solver

degradation of the performance of the parallel solver is caused by the load imbalance of the slave processes when the number of tasks is not sufficient to guarantee a balanced distribution of tasks. In the case when the complexity of all tasks is the same, this load imbalance can be estimated as [94]

$$g(M,p) = \frac{\max_{0 \le j < p} M_j}{M/p},$$

where M is the number of tasks, p is the number of processes and M_j is the number of tasks solved by jth process. For example, if $M = 800$ then for 400 and 799 processes the parallel computation time is the same $T_{400} = T_{799}$ since in both cases some processes should solve 2 tasks. Even for large size

problems this bottleneck is important if the problem is solved on massively parallel computers.

The weak scalability results of the proposed parallel algorithm are shown in Table 2.10, where again the scaling is done controlling the accuracy of the algorithm. Thus, when p is increased 8 times all three parameters N_{x_1}, N_{x_2} and M are increased 2 times.

Table 2.10: Weak scaling of the parallel integral solver

$N_{x_1} \times N_{x_2},\ M$	p	$n_d \times n_c$	Partition	Block	T_p	E_{n_d}
400^2, 400	16	1×16	cyclic	1	22.48	1.00
800^2, 800	128	8×16	cyclic	1	26.75	0.84
1600^2, 1600	1024	64×16	cyclic	1	36.12	0.62
400^2, 400	16	1×16	dynamic	1	22.79	1.00
800^2, 800	128	8×16	dynamic	1	26.54	0.86
1600^2, 1600	1024	64×16	dynamic	1	35.70	0.64

A degradation of the performance of the parallel solver is caused by the need to solve simultaneously in one node elliptic problems of increasing size.

Next, we will present the convergences analysis of the case with geometric MG preconditioners and thus the setup costs are minimal. The accuracy and solution times of the parallel quadrature solver with graded mesh (2.12) and with exponential mesh (2.14) are shown in Tables 2.11–2.12. The introduction of additional dimension gives the second order of convergence with the graded mesh (2.12) and exponential order of convergence with exponential mesh (2.14), the obtained discrete error values are close to values from Table 2.1.

Minimal parallel solution times T_p with optimal number of groups of processors and decomposition of the discrete mesh and corresponding speed-ups S_p are demonstrated using up to 256 parallel processes. For 256 processors we obtained the values of S_p equal to 114 and 78 with graded mesh (2.12) and with exponential mesh (2.14), respectively.

Table 2.11: The accuracy and solution times of the parallel quadrature solver with graded mesh on uniform space grid $N \times N$

N	M	Error	T_1	p	T_p	S_p
128	128	0.009710	2.67	32	0.15	17.35
256	128	0.007041	9.90	64	0.47	20.87
512	256	0.004854	105.47	208	1.45	72.63
1024	256	0.003517	505.51	256	4.41	114.50
2048	512	0.002417	4764.46			

Table 2.12: The accuracy and solution times of the parallel quadrature solver with exponential mesh on uniform space grid $N \times N$

N	m_1	m_2	k	Error	T_1	p	T_p	S_p
128	40	14	0.50	0.00956	0.43	16	0.062	6.90
128	89	30	0.33	0.00946	0.87	32	0.096	9.07
256	40	14	0.50	0.00678	1.56	32	0.13	11.89
256	89	30	0.33	0.00669	3.20	32	0.22	14.60
512	40	14	0.50	0.00482	7.97	96	0.25	31.81
512	89	30	0.33	0.00473	16.06	96	0.41	38.95
1024	40	14	0.50	0.00343	38.66	256	0.71	54.58
1024	89	30	0.33	0.00334	77.45	256	1.49	51.89
2048	40	14	0.50	0.00245	185.00	256	2.37	78.01
2048	89	30	0.33	0.00236	370.08	256	4.70	78.75

BURA Method (M4)

Solution of the non-local fractional diffusion problem (2.16) is transformed into a computation of sums (2.15). The corresponding $m+1$ discrete elliptic subproblems can be solved independently. The same two-level parallel algorithm as for integral method (M3) can be used. However, the parallelisation degree is small, therefore, two-level parallel algorithm is limited.

Table 2.13: The accuracy and solution times of BURA(m) solver on uniform space grid $N \times N$

	BURA(5)		BURA(6)		BURA(7)	
N	Error	Time	Error	Time	Error	Time
16	0.02697	0.009	0.02704	0.010	0.027025	0.010
32	0.01894	0.012	0.01893	0.013	0.018947	0.015
64	0.01339	0.022	0.01339	0.026	0.013380	0.029
128	0.00947	0.061	0.00946	0.072	0.009464	0.081
256	0.02775	0.209	0.00669	0.245	0.006691	0.280
512	0.07146	0.997	0.02804	1.164	0.006003	1.35
1024	0.17077	4.82	0.05785	5.54	0.02803	6.43
2048	0.06132	21.96	0.17017	26.11	0.05719	30.40

The accuracy and solution times of the BURA(m) solvers are shown in Table 2.13. Analysing the accuracy of BURA(5) solutions, we see that the error values are close to values in Table 2.1 for $N \leq 128$, and BURA(6), BURA(7) for $N \leq 256$. The accuracy of BURA(m) solutions is decreasing with increased N due to the approximation error. Still we note that determination of coefficients c_j and d_j in (2.15) for arbitrary m and β is a non-trivial

and computation demanding task [45, 92]. In the work [92], coefficients and errors $\varepsilon_m(\beta)$ are computed with high accuracy for $\beta = \{i/8,\ i = 1, \ldots, 7\}$ and $m \leq 30$. The corresponding coefficients, however, are not provided.

Comparison of Accuracy

We presented the comparison in terms of the accuracy and parallel computational times for five numerical algorithms: elliptic (2.6), pseudo-parabolic (2.9), quadrature algorithm with graded mesh (2.12), quadrature algorithm with exponential mesh (2.14), BURA(7) (2.15) algorithm. The reference (exact) solution was obtained for test problem (2.16)–(2.17) with $\beta = 0.25$ using the pseudo-spectral Fourier method [4] on uniform space grid with $N = 32768$ points.

The parallel solvers are compared in terms of solution times T_p needed to achieve certain accuracy in Figure 2.4. Minimal parallel solution times are obtained with an optimal number of parallel processes and parameters of parallel algorithms. The corresponding speed-ups are shown in Figure 2.5.

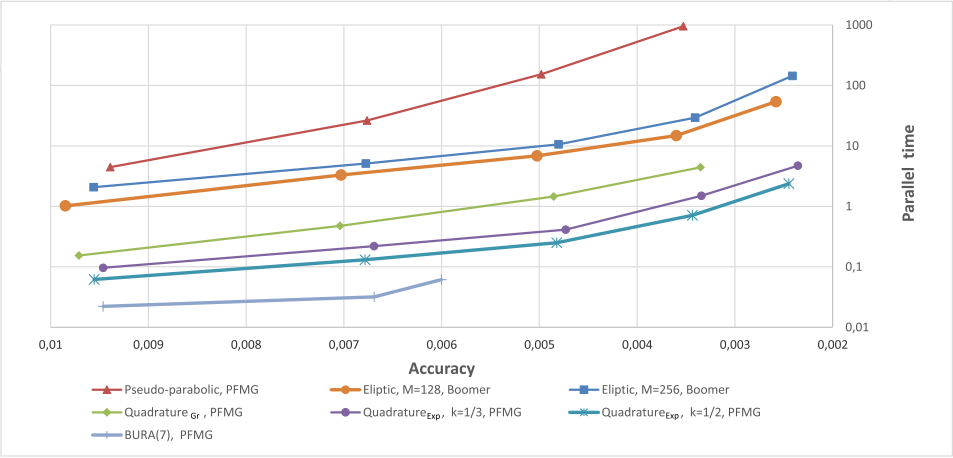


Figure 2.4: Comparison of parallel solvers in terms of accuracy

The performed analysis has shown that selection of the best algorithm is problem- dependent. The BURA method of rational approximation presents an attractive alternative if the high accuracy is not required and coefficients of the method for given fractional power $\beta = 0.25$ are known in advance. For the considered test problem, the relative accuracy of 0.006 can be achieved very cheaply with the BURA(7) algorithm. Seeking more accurate solutions, quadrature algorithm with exponential mesh has the smallest computational cost needed to achieve the required accuracy.

Pseudo-parabolic problem 2.9 was solved using a uniform grid, which

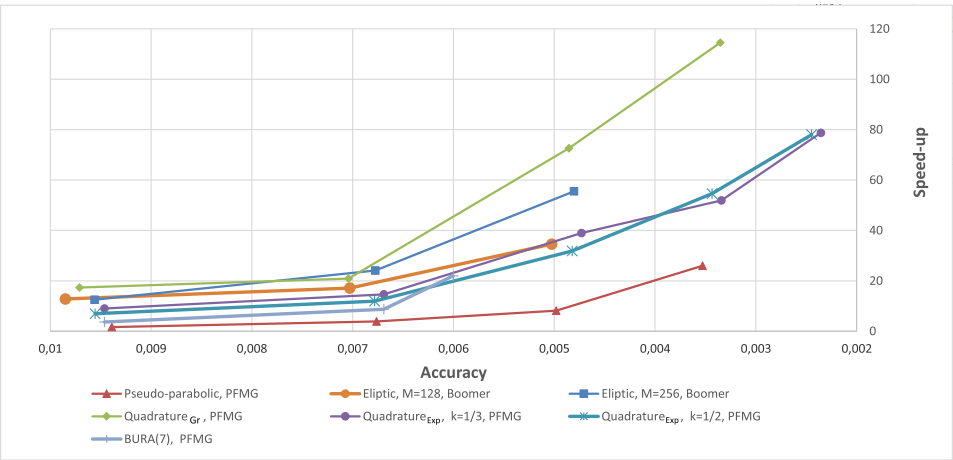


Figure 2.5: Speed-ups of parallel solvers

allowed us to evaluate the properties of its parallelisation, but it becomes competitive in terms of computational time only when geometrically graded time stepping scheme [32] is used. The results of method (M2) with geometrically graded time stepping scheme will be presented in Section 2.5.2.

2.5.2 Convergence and Scalability Analysis of the Parallel Algorithms for 3D Problems

To develop the parallel numerical algorithms, we use the same two-level parallelization approach. The set of M discrete 3D elliptic subproblems defines the first level of parallelization. A significant difference is observed among the three considered methods (M2)–(M4). For the pseudo-parabolic method (M2), the discrete subproblems cannot be solved concurrently. For the quadrature method (M3), the number $M = M(\beta, k)$ depends on the required level of transformation error. For the BURA method (M4), the number $M = m + 1$ is quite small. On the second level of parallel algorithms, each discrete subproblem is solved in parallel using the domain decomposition method. To implement these parallel algorithms, we use our C++ parallel programing templates [26] with MPI library for master–slave, parallelization on the first level. Discrete subproblems are statically or dynamically distributed between the groups of parallel processes. Different number of groups can be used. On the second level, each obtained discrete subproblem is solved in parallel by the corresponding group of processes using the parallel preconditioned conjugate gradient method for solving the arising linear systems.

It is important to note that the method (M1) was excluded from the

list of considered method as less suitable for 3D case due to high memory requirements.

In recent study [12], authors propose the new numerical methods for fractional diffusion. The numerical approximation is based on different definitions of fractional powers of elliptic operators: PDE approach for the spectral definition, integral formulation and a discretization of the Dunford-Taylor formula. In this study [12] authors demonstrated, that the new Dunford-Taylor algorithm is good for parallelisation and gives good weak and strong scalability properties. Authors modified the first method (M1), which made it competitive in 3D case. In the next study [65], authors propose the new approach for first method (M1), it gave reduction of the computational complexity and improved convergence properties of the state-of-the-art discretisation. It was achieved by discretisation with linear finite elements in the original domain and h_p-finite elements in the extended direction. In one more recent study [9], authors demonstrated exponential rates of convergence of h_p-finite elements method for Caffarelli-Silvestre extension in the truncated cylinder $\Omega(0, Y)$ with anisotropic geometric meshes. In study [29], authors recommended a novel discretisation of the spectral fractional on bounded domains based on integral expression of the operator via the heat-semigroup method. The comparison of these new methods requires additional computations, it is planed future research.

Pseudo-Parabolic Method (M2)

In Section 2.5.1 we have used the uniform time-stepping. However, our experiments in solving the current test problem (2.16)–(2.17) have not shown the second order of convergence in time, even for quite small time steps. Such convergence rate is expected asymptotically for the Crank–Nicolson scheme. Consequently a large number of time steps is essential to reduce the time discretization (non-local problem transformation) error to the level of space discretization error. Such a behavior of this version of the numerical scheme makes the respective parallel algorithm not competitive in comparison with the remaining two methods (M3,M4).

In study, [32] it has been shown that the convergence rate of time discretization scheme with the uniform time-stepping depends on the smoothness of the solution. A new geometrically graded time-stepping scheme is proposed to deal with the singular behavior of the solution for time t close to 0. The authors [32] have proved that the convergence rate of the time discretization schemes with such geometrically refined mesh does not depend on the smoothness of the solution.

We performed convergence tests of the constructed scheme (2.9), solving the 3D test problem (2.16)–(2.17). In Table 2.14, we present the relative error E_N^{M2} (2.18) of the discrete solution U_N^{M2} obtained on a uniform space grid with N^3 cells and doing $M = (K+1)J$ time steps. We also try to estimate the transformation (time discretization) error. We use the reference solution U_N^F obtained with the Fourier method on the same space grid instead of the exact solution u in (2.18) to compute and present the estimate $\hat{E}_N^{M2}$ of the transformation error of the method.

In Table 2.14, we present the total number of iterations, N_{iter}, to evaluate the performance of the iterative method, which is employed to solve the discrete 3D elliptic subproblems. We use the preconditioned conjugate gradient method with the geometric multigrid PFMG solver from the HYPRE library [34, 35] as preconditioner. PFMG is a parallel alternating semi-coarsening V-cycle multigrid solver that uses pointwise relaxation for solving scalar diffusion equations on logically rectangular grids. We perform one iteration of the PFMG solver with the symmetric red-black Gauss-Seidel relaxation and default parameters. We present in Table 2.14 the solution times T_1, later used in our parallel scalability studies.

Table 2.14: Convergence of the discrete solution U_N^{M2} (2.9) obtained by the pseudo-parabolic method (M2) for $\beta = 0.25$ and $\beta = 0.75$.

			$\beta = 0.25$				$\beta = 0.75$			
N^3	J	M	E_N^{M2}	$\hat{E}_N^{M2}$	N_{iter}	Time	E_N^{M2}	$\hat{E}_N^{M2}$	N_{iter}	Time
16^3	4	52	0.034854	0.000800	178	$1.65 \cdot 10^{-1}$	0.012284	0.000412	189	$1.69 \cdot 10^{-1}$
16^3	8	104	0.035453	0.000201	324	$3.13 \cdot 10^{-1}$	0.012492	0.000104	362	$3.29 \cdot 10^{-1}$
16^3	16	208	0.035604	0.000050	636	$6.18 \cdot 10^{-1}$	0.012545	0.000026	654	$6.28 \cdot 10^{-1}$
32^3	4	60	0.024296	0.000873	224	$1.16 \cdot 10^{0}$	0.004238	0.000425	235	$1.19 \cdot 10^{0}$
32^3	8	120	0.024949	0.000220	415	$2.22 \cdot 10^{0}$	0.004359	0.000107	454	$2.33 \cdot 10^{0}$
32^3	16	240	0.025114	0.000055	796	$4.38 \cdot 10^{0}$	0.004389	0.000027	831	$4.48 \cdot 10^{0}$
64^3	4	68	0.016950	0.000890	270	$1.13 \cdot 10^{1}$	0.001473	0.000427	282	$1.15 \cdot 10^{1}$
64^3	8	136	0.017580	0.000224	519	$2.20 \cdot 10^{1}$	0.001533	0.000107	546	$2.21 \cdot 10^{1}$
64^3	16	272	0.017739	0.000056	948	$4.25 \cdot 10^{1}$	0.001548	0,000027	1021	$4.38 \cdot 10^{1}$
128^3	4	76	0.011814	0.000895	317	$1.30 \cdot 10^{2}$	0.000512	0.000428	327	$1.32 \cdot 10^{2}$
128^3	8	152	0.012379	0.000225	612	$2.56 \cdot 10^{2}$	0.000539	0.000107	628	$2.59 \cdot 10^{2}$
128^3	16	304	0.012521	0.000057	1117	$4.90 \cdot 10^{2}$	0.000546	0,000027	1204	$5.08 \cdot 10^{2}$
256^3	4	84	0.008207	0.000898	361	$1.22 \cdot 10^{3}$	0.000383	0.000428	378	$1.26 \cdot 10^{3}$
256^3	8	168	0.008692	0.000226	696	$2.41 \cdot 10^{3}$	0.000189	0.000107	710	$2.43 \cdot 10^{3}$
256^3	16	336	0.008815	0.000057	1314	$4.67 \cdot 10^{3}$	0.000192	0.000027	1377	$4.73 \cdot 10^{3}$
512^3	4	92	0.005632	0.000898	416	$1.34 \cdot 10^{4}$	0.000417	0.000428	425	$1.37 \cdot 10^{4}$
512^3	8	184	0.006035	0.000226	778	$2.60 \cdot 10^{4}$	0.000096	0.000107	817	$2.67 \cdot 10^{4}$
512^3	16	368	0.006137	0.000057	1506	$5.10 \cdot 10^{4}$	0.000067	0.000027	1544	$5.16 \cdot 10^{4}$

Analysing the values of the relative error E_N^{M2} in Table 2.14, the reference values from Table 2.2 can be achieved by increasing the number of time steps J and thus reducing the time discretization error. In almost all cases the value of the error E_N^{M2} is smaller for $J = 2$ than for $J = 4$ and

$J = 8$. This is caused by effect of compensation of errors, which is discussed earlier.

Analysing the error $\hat{E}_N^{M2}$ in Table 2.14, the second-order convergence in time can be observed: the error is reduced around four times by doubling J. These results confirm the theoretical estimates [32] for the selected 3D test problem as well. For comparison, we present the results obtained using the uniform time-stepping mesh as well. Then, the error $E_{512}^{M2} = 0.008378$ is obtained for $\beta = 0.25$ and 4096 time steps. The computation time of the parallel algorithm is $T_{256} = 9.04 \cdot 10^3$, that is here 256 processes are used. Method (M2) with the geometrically graded time-stepping scheme can be considerably more competitive.

Analysing the performance of the iterative solver, a quite robust performance of the geometric multigrid preconditioner can be observed. The average number of iterations per time step - N_{iter}/M is quite stable. It is slightly reducing for the doubling M and fixed N. Moreover, it is slightly increasing for the fixed J and doubling N: 3.12, 3.46, 3.82, 4.03, 4.14, 4.23 for $J = 8$. In addition, note here that the total number of iterations N_{iter} and the solution times T_1 are almost identical for $\beta = 0.25$ and $\beta = 0.75$, that is, the computational complexity of the solvers does not depend on the fractional power parameter β.

Quadrature Method (M3)

In Table 2.15, we present the relative error E_N^{M3} (2.18) of the discrete solution U_N^{M3} (2.14) obtained on the uniform space grid with N^3 cells. Two values of the quadrature's parameter k are used: $1/2$ and $1/3$. We estimate and present the method's transformation (integral approximation) error $\hat{E}_N^{M3}$ using the reference Fourier solution U_N^F instead of the exact solution u in (2.18).

In addition we present in Table 2.15 the total number of iterations, N_{iter}, that are preformed by the PCG method to solve M discrete 3D elliptic subproblems. Finally, we present the solution times T_1, later used in our parallel scalability studies.

Analysing the relative error E_N^{M3} values in Table 2.15, the reference values from Table 2.2 are achieved by decreasing the parameter k and thus reducing the error of transformation (integral approximation).

The exponential convergence of employed quadrature formula can be obtained from the transformation error $\hat{E}_N^{M3}$ values given in Table 2.15. The error is reduced 138 times for $\beta = 0.25$ and 98 times for $\beta = 0.75$ ($N = 512$) by increasing the number M of quadrature points y_j twice. For

Table 2.15: Convergence of the discrete solution U_N^{M3} (2.14) obtained by the quadrature method (M3) for $\beta = 0.25$ and $\beta = 0.75$.

			$\beta = 0.25$				$\beta = 0.75$			
N^3	k	M	E_N^{M3}	$\hat{E}_N^{M3}$	N_{iter}	Time	E_N^{M3}	$\hat{E}_N^{M3}$	N_{iter}	Time
16^3	1/2	55	0.035742	$8.82 \cdot 10^{-5}$	187	$1.62 \cdot 10^{-1}$	0.012647	$9.77 \cdot 10^{-5}$	343	$2.18 \cdot 10^{-1}$
16^3	1/3	120	0.035654	$3.71 \cdot 10^{-7}$	370	$3.34 \cdot 10^{-1}$	0.012564	$9.16 \cdot 10^{-7}$	724	$4.62 \cdot 10^{-1}$
32^3	1/2	55	0.025257	$8.82 \cdot 10^{-5}$	194	$9.63 \cdot 10^{-1}$	0.004482	$9.78 \cdot 10^{-5}$	350	$1.33 \cdot 10^{0}$
32^3	1/3	120	0.025169	$4.50 \cdot 10^{-7}$	383	$1.99 \cdot 10^{0}$	0.004400	$1.00 \cdot 10^{-6}$	737	$2.81 \cdot 10^{0}$
64^3	1/2	55	0.017880	$8.82 \cdot 10^{-5}$	204	$8.11 \cdot 10^{0}$	0.001636	$9.78 \cdot 10^{-5}$	360	$1.12 \cdot 10^{1}$
64^3	1/3	120	0.017792	$5.03 \cdot 10^{-7}$	395	$1.66 \cdot 10^{1}$	0.001555	$1.04 \cdot 10^{-6}$	749	$2.36 \cdot 10^{1}$
128^3	1/2	55	0.012657	$8.82 \cdot 10^{-5}$	211	$8.30 \cdot 10^{1}$	0.000631	$9.78 \cdot 10^{-5}$	367	$1.15 \cdot 10^{2}$
128^3	1/3	120	0.012570	$5.53 \cdot 10^{-7}$	407	$1.70 \cdot 10^{2}$	0.000550	$1.05 \cdot 10^{-6}$	761	$2.42 \cdot 10^{2}$
256^3	1/2	55	0.008943	$8.82 \cdot 10^{-5}$	238	$7.35 \cdot 10^{2}$	0.000273	$9.78 \cdot 10^{-5}$	419	$1.04 \cdot 10^{3}$
256^3	1/3	120	0.008856	$6.04 \cdot 10^{-7}$	457	$1.49 \cdot 10^{3}$	0.000194	$1.05 \cdot 10^{-6}$	870	$2.25 \cdot 10^{3}$
512^3	1/2	55	0.006259	$8.82 \cdot 10^{-5}$	249	$7.10 \cdot 10^{3}$	0.000140	$9.78 \cdot 10^{-5}$	427	$1.07 \cdot 10^{4}$
512^3	1/3	120	0.006171	$6.41 \cdot 10^{-7}$	470	$1.52 \cdot 10^{4}$	0.000068	$1.05 \cdot 10^{-6}$	883	$2.25 \cdot 10^{4}$

comparison, we note that the relative error $E_{512}^{M3} = 0.0062042$ for $\beta = 0.25$ is obtained with $M = 512$ quadrature points if the quadrature formula with β-dependent graded mesh is applied.

Analysing the computational complexity of numerical algorithm (2.14) for the same N, M, and different β values, we observe that 1.7–1.9 times more PCG iterations are performed for $\beta = 0.75$ in comparison with $\beta = 0.25$. This result is explained by the adaptive nature of the summation range $[-m_1, m_2]$:

$$\beta = 0.25: \quad [-40, 14] \text{ for } k = 1/2 \quad \text{and} \quad [-89, 30] \text{ for } k = 1/3;$$
$$\beta = 0.75: \quad [-14, 40] \text{ for } k = 1/2 \quad \text{and} \quad [-30, 89] \text{ for } k = 1/3.$$

For negative quadrature points $y_j = kj$, coefficient e^{-2y_j} increases rapidly and then the respective matrices $e^{-2y_j} I_h + L_h$ become with a strongly dominating diagonal. The linear systems with such matrices are solved in just one iteration. For example, in case of $N = 16$ and $k = 1/3$, 76 linear systems are solved in one iteration for $\beta = 0.25$ and only 15 systems for $\beta = 0.75$. Most of the other linear systems in (2.14) are solved in 7–8 iterations for all N sizes, as is expected from the multigrid preconditioner properties. Such a difference in computational complexity for different β values is not observed for the quadrature formula with β-dependent graded mesh.

BURA Method (M4)

Analysing the values of the relative error E_N^{M4} in Table 2.16, many results, which significantly differ from the reference values in Table 2.2 are

observed. The obtained relative errors, which are close to the reference values, are marked with bold font. For the test problem with $\beta = 0.25$, the approximate solutions U_N^{M4} are sufficiently accurate on the coarse grids with $N = 16, 32, 64$. However, the accuracy starts to deteriorate with the further increasing N (decreasing h) due to the increase in the approximation error. To some extent, this effect can be mitigated by using higher order approximations with $m > 5$. However, the relative error of 0.006171 is not accessible with the available approximations (coefficients c_j and d_j). For the test problem with $\beta = 0.75$, only the R-BURA method with $m = 7$ produces satisfactory results. The relative error of 0.001554 is obtained on the grid with $N = 64$.

Table 2.16: Accuracy of the discrete solution U_N^{M4} (2.15) obtained by the BURA method (M4) for $\beta = 0.25$ and $\beta = 0.75$.

			$\beta = 0.25$				$\beta = 0.75$			
N^3	Method(m)	M	E_N^{M4}	$\dot{E}_N^{M4}$	N_{iter}	Time	E_N^{M4}	$\dot{E}_N^{M4}$	N_{iter}	Time
16^3	BURA(5)	6	**0.035645**	$3.12 \cdot 10^{-4}$	33	$2.58 \cdot 10^{-2}$	0.022243	$1.17 \cdot 10^{-2}$	38	$2.81 \cdot 10^{-2}$
16^3	BURA(6)	7	**0.035657**	$4.06 \cdot 10^{-4}$	39	$2.97 \cdot 10^{-2}$	**0.012599**	$1.42 \cdot 10^{-3}$	45	$3.20 \cdot 10^{-2}$
16^3	BURA(7)	8	**0.035656**	$7.10 \cdot 10^{-5}$	45	$3.33 \cdot 10^{-2}$	**0.012584**	$3.04 \cdot 10^{-3}$	51	$3.51 \cdot 10^{-2}$
16^3	R-BURA(7)	8					**0.012563**	$8.04 \cdot 10^{-5}$	47	$3.40 \cdot 10^{-2}$
32^3	BURA(5)	6	**0.025167**	$3.84 \cdot 10^{-4}$	33	$1.39 \cdot 10^{-1}$	0.015659	$1.50 \cdot 10^{-2}$	38	$1.50 \cdot 10^{-1}$
32^3	BURA(6)	7	**0.025168**	$6.35 \cdot 10^{-4}$	40	$1.65 \cdot 10^{-1}$	0.005484	$4.55 \cdot 10^{-3}$	45	$1.77 \cdot 10^{-1}$
32^3	BURA(7)	8	**0.025169**	$4.07 \cdot 10^{-4}$	45	$1.84 \cdot 10^{-1}$	0.007342	$3.99 \cdot 10^{-3}$	52	$2.01 \cdot 10^{-1}$
32^3	R-BURA(7)	8					**0.004399**	$5.71 \cdot 10^{-5}$	48	$1.91 \cdot 10^{-1}$
64^3	BURA(5)	6	**0.017791**	$2.30 \cdot 10^{-3}$	33	$1.09 \cdot 10^{0}$	0.008597	$7.57 \cdot 10^{-3}$	38	$1.19 \cdot 10^{0}$
64^3	BURA(6)	7	**0.017792**	$8.44 \cdot 10^{-4}$	40	$1.31 \cdot 10^{0}$	0.011952	$1.09 \cdot 10^{-2}$	45	$1.40 \cdot 10^{0}$
64^3	BURA(7)	8	**0.017792**	$6.46 \cdot 10^{-4}$	45	$1.47 \cdot 10^{0}$	0.004588	$4.15 \cdot 10^{-3}$	52	$1.61 \cdot 10^{0}$
64^3	R-BURA(7)	8					**0.001554**	$4.63 \cdot 10^{-4}$	47	$1.51 \cdot 10^{0}$
128^3	BURA(5)	6	0.025491	$1.92 \cdot 10^{-2}$	33	$1.10 \cdot 10^{1}$	0.018907	$1.86 \cdot 10^{-2}$	38	$1.21 \cdot 10^{1}$
128^3	BURA(6)	7	**0.012569**	$3.60 \cdot 10^{-3}$	39	$1.29 \cdot 10^{1}$	0.017447	$1.73 \cdot 10^{-2}$	45	$1.43 \cdot 10^{1}$
128^3	BURA(7)	8	**0.012569**	$8.64 \cdot 10^{-4}$	45	$1.48 \cdot 10^{1}$	0.002476	$2.13 \cdot 10^{-3}$	50	$1.60 \cdot 10^{1}$
128^3	R-BURA(7)	8					0.002224	$1.87 \cdot 10^{-3}$	47	$1.53 \cdot 10^{1}$
256^3	BURA(5)	6	0.061277	$5.69 \cdot 10^{-2}$	35	$9.48 \cdot 10^{1}$	0.042304	$4.22 \cdot 10^{-2}$	41	$1.05 \cdot 10^{2}$
256^3	BURA(6)	7	0.024703	$2.03 \cdot 10^{-2}$	40	$1.09 \cdot 10^{2}$	0.014040	$1.39 \cdot 10^{-2}$	49	$1.24 \cdot 10^{2}$
256^3	BURA(7)	8	**0.008855**	$3.67 \cdot 10^{-3}$	48	$1.28 \cdot 10^{2}$	0.005037	$5.04 \cdot 10^{-3}$	55	$1.40 \cdot 10^{2}$
256^3	R-BURA(7)	8					0.005349	$5.27 \cdot 10^{-3}$	50	$1.32 \cdot 10^{2}$
512^3	BURA(5)	6	0.120413	$1.20 \cdot 10^{-1}$	34	$8.57 \cdot 10^{2}$	0.023142	$2.32 \cdot 10^{-2}$	41	$9.89 \cdot 10^{2}$
512^3	BURA(5)	6	0.120413	$1.20 \cdot 10^{-1}$	34	$8.57 \cdot 10^{2}$	0.023142	$2.32 \cdot 10^{-2}$	41	$9.89 \cdot 10^{2}$
512^3	BUR A(6)	7	0.050764	$4.77 \cdot 10^{-2}$	40	$1.00 \cdot 10^{3}$	0.014994	$1.50 \cdot 10^{-2}$	48	$1.17 \cdot 10^{3}$
512^3	BURA(7)	8	0.023393	$2.03 \cdot 10^{-2}$	47	$1.20 \cdot 10^{3}$	0.016757	$1.67 \cdot 10^{-2}$	55	$1.34 \cdot 10^{3}$
512^3	R-BURA(7)	8					0.010876	$1.08 \cdot 10^{-2}$	49	$1.26 \cdot 10^{3}$

Analysing the computational complexity of numerical algorithm (2.15) for the same N, M, and different β values, we observe that the difference is not significant. On average, one more PCG iteration is performed for each of the solved linear systems for $\beta = 0.75$, that is, 6 and 7 iterations, respectively. The number of iterations N_{iter} is stable with the increasing N, as it should be expected for the multigrid preconditioner.

Comparison of Accuracy

We compare in Fig. 2.6(a) and (b) the serial solution times T_1 from Tables 2.14 to 2.16 that are essential to achieve a similar accuracy with different algorithms. Note that logarithmic scales are used for the relative error E_N^{M*} and serial times T_1 in seconds. We compare the results for the pseudo-parabolic method M2 with $J = 8$, the quadrature method M3 with $k = 1/2$ and $k = 1/3$, and the BURA method M4 with BURA(7) for $\beta = 0.25$ and R-BURA(7) for $\beta = 0.75$.

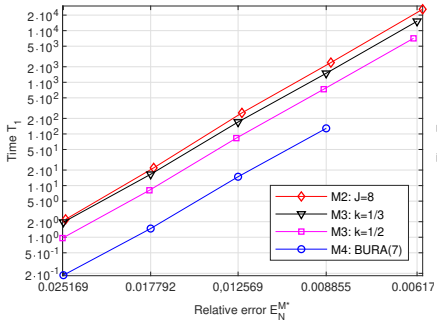
We see that for $\beta = 0.25$ the accuracy of up to two digits is very cheaply obtained by the BURA(7) method (M4). The acceptable error level depends on the application. In some cases, the percent error values of 2.5% - 0.9% may well be acceptable, for example, due to the high level of measurement errors. In these cases, the BURA(m) method (M4) is the best choice. However, if this level of accuracy is not acceptable and coefficients of the BURA method are not available for higher order approximations, then the quadrature method (M3) with $k = 1/2$ is recommended to obtain more accurate solutions.

Due to the faster convergence rate of discrete solution U_h for $\beta = 0.75$, more accurate solutions can be obtained on the same space grids in comparison to $\beta = 0.25$. However, in order to achieve such accuracy, the transformation error also should be decreased to lower levels. R-BURA(7) is showing the best results for the method (M4) and allows to achieve the accuracy of 0.001554 (i.e. 0.1554%). Again, for the higher accuracy, the quadrature method (M3) is recommended.

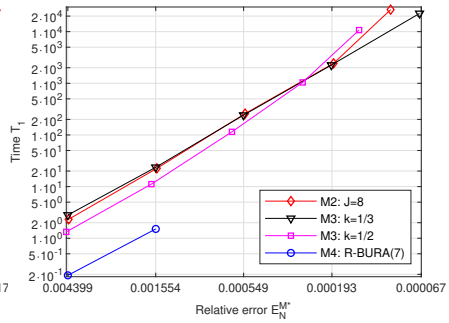
It should be noted that the results of pseudo-parabolic method (M2) with $J = 8$ are very close to the results of quadrature method (M3) with $k = 1/3$ in terms of accuracy and related computational costs for $N = 128$ and 256. This result confirms the good performance of new geometrically graded time-stepping scheme for the pseudo-parabolic method (M2).

The detailed results of comparison of developed parallel solver are presented in Figure 2.6. The parallel algorithm for the pseudo-parabolic method (M2) shows the significantly larger solution times T_p and lower speedups S_p. For the smaller problems $N^3 = 32^3, 64^3, 128^3$, all 512 processes cannot be efficiently employed to achieve additional speedup. Owing to the relatively small number $M = m+1$ for the BURA method (M4), the parallel speedups S_p of the according parallel algorithm are closer to the speedups of pseudo-parabolic method (M2) than to the quadrature method (M3). Third, the highest speedups are obtained for the quadrature method M3: $S_{512} = 259.96$

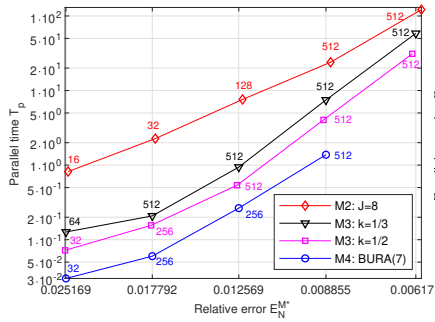
for $\beta = 0.25$ and $S_{512} = 262.27$ for $\beta = 0.75$ when the largest problem with $k = 1/3$ is solved. The reduction of transformation error with decreasing parameter k leads to the higher degree of concurrency.



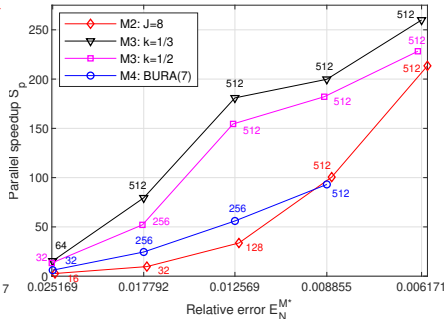
(a) Serial solution times T_1 for $\beta = 0.25$



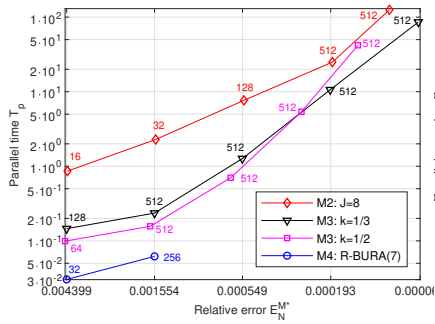
(b) Serial solution times T_1 for $\beta = 0.25$



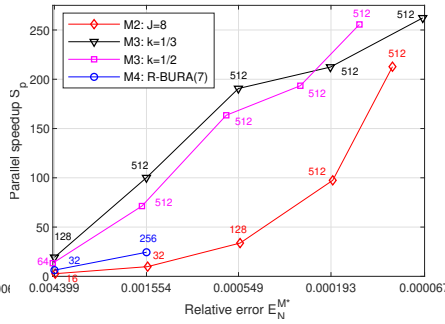
(c) Parallel solution times T_p for $\beta = 0.25$



(d) Parallel speedups S_p for $\beta = 0.25$



(e) Parallel solution times T_p for $\beta = 0.75$



(f) Parallel speedups S_p $\beta = 0.75$

Figure 2.6: Comparison of parallel numerical algorithms for selected methods (M2)–(M4) solving the 3D test problem (2.16)–(2.17)

2.6 Conclusions of the Second Chapter

The Fourier method is the best method when the FFT algorithm can be applied. In this case the eigenvectors and eigenvalues of the elliptic operator should be known in advance. For problems in non-rectangular domains and/or with general elliptic operator, more general methods should be considered.

The BURA method (M4) of the best uniform rational approximation presents an attractive alternative if the high accuracy of numerical solution is not required.

The R-BURA method shows better results for the fractional power β values closer to 1. However, coefficients of the method should be known in advance for the provided fractional power β and they are difficult to compute.

The pseudo-parabolic method (M2) with geometrically graded time-stepping scheme has shown in some cases competitive results in terms of accuracy and related computational costs. However, this method is outperformed in terms of parallel scalability by the quadrature method (M3).

The quadrature method (M3) with exponentially convergent quadrature formula is recommended if the higher accuracy solutions are required.

The first method (M1) appears to be very sensible to the partitioning of anisotropic y coordinate, which was introduced during the transformation. The setup of AMG preconditioner is taking the most of the total solution time (around 80% and more). The 2D partitioning of the problem domain is producing better performance results, since the significantly higher setup costs for the partitioning along the anisotropic coordinate y are not compensated by the better geometric quality of the 3D partitioning. The memory requirements of our elliptic problem solver, mostly of BoomerAMG preconditioner, are growing very fast.

Chapter 3

The construction of absorbing boundary conditions for the one-dimensional Schrödinger equation*

In the second part of the dissertation the non-local problem with fractional powers of elliptic operators was investigated. The aim was to construct and compare parallel algorithms for different state-of-the-art numerical methods used for solving the fractional powers of elliptic problems. The fourth method (M4) is based on the best uniform rational approximation (BURA) of a scalar function of the form t^γ. In this part of dissertation the one dimensional linear Schrödinger equation is analysed. We investigate how to construct the absorbing boundary conditions for this equation. These conditions are obtained by using rational functions to approximate exact transparent boundary conditions. The aim is to find optimal coefficients of rational functions. Different strategies are investigated for the optimal selection of coefficients of these rational functions, including the Padé approximation, the L_2 norm approximation of the Fourier symbol, L_2 minimization of the reflection coefficient, and two adaptive minimization techniques that minimize the error of solution for a selected benchmark problems with known

*Bugajev, A.; Čiegis, R.; Kriauzienė, R.; Leonavičienė, T.; Žilinskas, J. On the accuracy of some absorbing boundary conditions for the Schrödinger equation Mathematical modelling and analysis. Vilnius: Taylor & Francis, VGTU. 2017, Vol. 22, iss. 3, p. 408–423. DOI:10.3846/13926292.2017.1306725.

exact solutions and coupled adaptive strategy. The formulated optimization problems are computationally costly; therefore, parallel computing was used. The results of computational experiments are given and a detailed comparison of the efficiency of these techniques is presented.

Parts of this chapter are published in [15].

3.1 Introduction to this Chapter

The initial-value problem for the linear one-dimensional Schrödinger equation is

$$\begin{cases} i\dfrac{\partial \tilde{u}}{\partial t} + \dfrac{\partial^2 \tilde{u}}{\partial x^2} = 0, \quad -\infty < x < \infty, \quad t > 0, \\ \tilde{u}(x,0) = u_0(x). \end{cases} \tag{3.1}$$

The Schrödinger equation is widely used for modelling quantum mechanics and non-linear optics problems. Usually the equation is supplemented with an initial condition and only asymptotical behaviour of the solution at infinite boundaries is defined (the last requirement is equivalent to the boundedness of the solution in some specified norm). In order to solve this problem a discretisation must be performed and an infinite space domain must be reduced to a finite domain. Thus it is necessary to formulate correct artificial boundary conditions at the new boundaries of the selected domain.

In most cases we are interested to find the solution only on a finite domain so a proper restriction of the infinite domain is well suited for most real world modelling applications. Thus it is a common practice to approximate the initial value problem (3.1) by some initial-boundary value problem.

However, these new boundary conditions must be carefully constructed, because they can perturb essentially a solution of the initial value problem. It is well known that if simple standard boundary conditions are formulated on boundaries of the restricted domain (e.g. homogeneous Dirichlet boundary conditions), the solution after reaching a boundary will be reflected back into the domain and will pollute results of subsequent simulations. A simple and straightforward way to avoid such perturbations is to enlarge the domain of simulations and thus to delay the reflection of the wave from artificial boundaries. However, for a long time modelling this strategy is very inefficient – it forces calculations to be performed on a domain that is much bigger than the domain of interest.

Special artificial and transparent boundary conditions were developed and investigated in many papers, see [5, 6, 10, 14, 21, 66]. A comprehensive

review of construction of transparent boundary conditions not only for differential problems but also for discrete finite difference schemes, the stability analysis and computational experiments is given in [6,66]. The exact transparent boundary conditions are non-local in time and the full history of the solution at the boundary should be preserved during computations. We also mention papers where exact transparent boundary conditions are constructed and the stability of initial-boundary value problem is analysed for the differential Schrödinger equation [5,6,10]. A similar analysis of discrete unconditionally stable schemes for one-dimensional Schrödinger equation is done in [6, 7]. The transparent boundary conditions for high-order finite difference schemes are constructed in [67,78].

Non-local boundary conditions increase the computational costs and memory requirements of the numerical algorithms and they are inefficient for long time modelling problems. Thus it is a challenge to construct appropriate local boundary conditions and to avoid the negative long memory effects included into the definition of exact non-local transparent boundary conditions. A few approaches are proposed to construct such boundary conditions. Various absorbing boundary conditions (ABCs) are constructed for this purpose (see, e.g. [66] and references therein). These boundary conditions absorb the energy of waves as they reach the boundary area and attempt to minimize the amount of energy reflected by the artificial boundaries.

In this part of the dissertation four methods based on approximations by rational functions [14,66] are investigated.

- The first method is the classical Padé method, then coefficients of the approximation can be computed in explicit form [8,74,97].
- The second technique is based on the approximation of the Fourier symbol in the L_2 norm. In this method coefficients of the rational function are defined by solving the minimization problem.
- The third method is minimisation of the reflection coefficient [88], since the main aim of artificial boundary conditions is to avoid a reflection from the new boundaries. We note that the Nelder-Mead method [70] was used to find local minimum points in [88].
- The fourth approach is based on the adaptive strategy [39]. Two representative benchmarks are selected and exact solutions of these problems are known. Then coefficients of rational functions are determined by minimizing the error of the approximate solution in the

L_2 and L_∞ norms. This part of computations enabled us to test the robustness of the developed global minimization algorithms and to estimate the possibilities/limits of rational functions technique when the order of polynomials is small.

The formulated minimization problems are used as black box problems of global optimization [89, 102]. We use the multi-start strategy [63, 89] for estimating the globally optimal solution. Starting points are generated randomly and local optimization by the Nelder-Mead downhill simplex method [70] is applied. This strategy is also favorable for parallelization since different runs of local optimization are independent and may be performed in parallel.

The main aim was to estimate the accuracy of artificial boundary conditions by using results of selected computational experiments. The most important question was to investigate if it is possible to find a universal set of coefficients of rational functions which can be used for important different cases of one dimensional Schrödinger problems.

3.2 Formulation of the Problem

To solve the initial value problem (3.1) we approximate it by an initial-boundary value problem formulated in a finite space domain:

$$\begin{cases} i\dfrac{\partial u}{\partial t} + \dfrac{\partial^2 u}{\partial x^2} = 0, \quad x \in (a,b), \;\; t \in (0,T], \\ u(x,0) = u_0(x), \quad x \in [a,b], \\ L_l u(a) = 0, \quad L_r u(b) = 0, \quad t \in (0,T], \end{cases} \tag{3.2}$$

where operators L_l, L_r define the artificial boundary conditions. Let us assume that we are interested to find a solution only in domain $[A,B]$. Then operators L_l, L_r and the ends of a finite domain a, b must be such that

$$\int_0^T \int_A^B |\tilde{u} - u|^2 \, dxdt \le \varepsilon, \quad a \le A < B \le b, \tag{3.3}$$

where $\tilde{u}$ is the solution of (3.1), ε is a selected accuracy of the approximation. The extended domains $[a,A)$, $(B,b]$ can be used to damp possible reflections and oscillations of the solution from artificial boundaries. But for simplicity we will restrict ourselves to the case $A = a$, $B = b$.

Finite difference scheme. We approximate equation (3.2) by the Crank-Nicolson method. We consider the domain $[a,b]\times[0,T]$ and introduce discretization $\omega_h\times\omega_\tau$, where ω_h and ω_τ are discrete uniform grids, h and τ are discrete steps:

$$\omega_h=\big\{x_j:\ \ x_0=a,\ x_J=b,\ x_k=x_{k-1}+h,\ \ k=1,\dots,J\big\},\tag{3.4}$$

$$\omega_\tau=\{t^n:\ \ t^n=n\tau,\quad n=0,\dots,N,\ N\tau=T\}.\tag{3.5}$$

We consider numerical approximations U_j^n of the exact solution $u_j^n=u(x_j,t^n)$ at the grid points $(x_j,t^n)\in\omega_h\times\omega_\tau$. For the functions defined on the grid we introduce the forward and backward difference quotients with respect to x

$$\partial_x U_j^n=(U_{j+1}^n-U_j^n)/h,\quad \partial_{\bar x}U_j^n=(U_j^n-U_{j-1}^n)/h$$

and similarly the backward difference quotient and the averaging operator with respect to t

$$\partial_{\bar t}U_j^n=(U_j^n-U_j^{n-1})/\tau,\quad U_j^{n-0.5+\theta}=0.5\left(U_j^n+U_j^{n-1}\right).$$

We approximate the differential equation (3.2) by the Crank-Nicolson finite difference scheme [78]

$$i\partial_{\bar t}U_j^n+\partial_x\partial_{\bar x}U_j^{n-0.5}=0\quad x_j\in\omega_h,\ \ n>0.\tag{3.6}$$

Exact transparent boundary conditions. Following [5], for differential equation (3.2) we can write the exact transparent boundary conditions. To do this we factorize the Schrödinger equation (3.2)

$$\left(u_x-\sqrt{(-i)\partial_t}u\right)\left(u_x+\sqrt{(-i)\partial_t}u\right)=0$$

and get the following boundary conditions

$$\partial_n u+e^{-i\frac{\pi}{4}}D_t^{1/2}u=0,\quad x=a,b,\tag{3.7}$$

where ∂_n is the normal derivative and

$$D_t^{1/2}u(x,t)=\frac{1}{\sqrt{\pi}}\frac{\partial}{\partial t}\int_0^t\frac{u(x,s)}{\sqrt{t-s}}\,ds$$

is a nonlocal operator. There are many quadrature algorithms to approximate the integral in this boundary condition.

It is important to note, that a similar factorization can be done also for the discrete finite difference scheme (3.6) (see [5] for details), or for the high-order accuracy Numerov type finite difference scheme (see, e.g. [78] for details). Then specific discrete approximation algorithms are obtained and such finite difference schemes have the same stability properties as the original schemes formulated in the infinite domain.

A very simple and convenient approximation algorithm can be obtained if a semi-discrete finite difference scheme is considered, when the Crank-Nicolson approximation is applied only in time. Then after factorization, we get the following transparent boundary conditions [5]

$$\partial_n U^{n+1} = -e^{i\pi/4}\sqrt{\frac{2}{\tau}}\sum_{k=0}^{n+1}\beta_k U^{n+1-k},$$

where $\beta_k = (-1)^k\alpha_k$, $\alpha_0 = 1$, $\alpha_{2k} = \prod\limits_{j=1}^{k}(2j-1)/2j = \frac{2k-1}{2k}\alpha_{2k-2}$, $\alpha_{2k+1} = \alpha_{2k}$, $k \geq 0$:

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \ldots) = \left(1, 1, \frac{1}{2}, \frac{1}{2}, \frac{3}{8}, \frac{3}{8}, \ldots\right).$$

All these boundary conditions are non-local and for the implementation of them we must store the full history of the solution at the boundary points. Since the coefficients of discrete transparent boundary conditions (see, e.g. β_k) decay very slowly it is impossible to reduce the complexity of the algorithm by summing only a small number of terms.

3.3 Methods for Finding Coefficients for Absorbing Boundary Conditions

Our approach for constructing local artificial boundary conditions is based on approximation of the transparent boundary condition (3.7) by rational functions. First, the Fourier transform is applied to get a spectral representation of the boundary condition:

$$\partial_n \hat{u} + e^{-i\frac{\pi}{4}}\sqrt[+]{i\omega}\hat{u} = 0, \tag{3.8}$$

where $\hat{u}(x,\omega) = \frac{1}{\sqrt{2pi}}\int_{-\infty}^{+\infty} u(x,t)e^{-i\omega t}dt$. Then the Fourier symbol $\sqrt[+]{i\omega}$ is approximated by rational functions:

$$\sqrt[+]{i\omega} \approx \frac{P_m(i\omega)}{Q_m(i\omega)} = c_0 + \sum_{k=1}^{m}\frac{c_k i\omega}{i\omega + d_k},$$

where the stability of finite difference schemes leads to the restrictions on coefficients $c_k > 0$, $d_k > 0$, $k = 0, \ldots, m$ (see [88]).

The important advantage of this algorithm is that there exists an efficient implementation of the inverse Laplace transform, proposed by Lindmann in [59], which leads to local boundary conditions. New functions $\hat{\varphi_k}(x, \omega) = \dfrac{1}{i\omega + d_k}\hat{u}(x, \omega)$ are introduced for $x = a, b$ and linear equations are obtained after simple computations

$$\hat{u} = i\omega\hat{\varphi_k} + d_k\hat{\varphi_k}, \quad k = 1, \ldots, m.$$

Applying the inverse Laplace transform we get the initial value ODEs for $\varphi_k(x, t)$:

$$\frac{d\varphi_k(x,t)}{dt} + d_k\varphi_k(x,t) = u\,(x,t)\,, \quad x = a, b, \quad k = 1, \ldots, m. \tag{3.9}$$

Then the approximate boundary conditions can be written as:

$$\partial_n u = -e^{-i\frac{\pi}{4}}\left(\left(\sum_{k=0}^{m} a_k\right)u - \sum_{k=1}^{m} c_k d_k \varphi_k\right), \quad x = a, b. \tag{3.10}$$

We formulate four methods to compute the coefficients c_k, d_k. Let us denote the set of coefficients $S_m(c, d) = (c_0, c_1, \ldots, c_m, d_1, \ldots, d_m)$, satisfying the stability requirements $c_j \geq 0$, $d_j \geq 0$.

3.3.1 Padé Coefficients

First a well known algorithm is applied to compute the Padé approximation [74](Padé approximation is one of the approximations by rational functions). Then the coefficients are defined by [88]

$$c_0^m = 0, \quad c_k^m = \frac{1}{m\cos^2\left(\dfrac{(2k+1)\pi}{4m}\right)}, \quad d_k = \tan^2\left(\frac{(2k+1)\pi}{4m}\right). \tag{3.11}$$

Since the Padé approximation is based on a local information of the function it is approximating, the convergence can be slow and a sufficiently large m should be used. As a consequence the memory requirements still can be quite large.

3.3.2 Approximation of the Fourier Symbol in the L_2 Norm

In this algorithm we directly minimize the Fourier symbol (FS) approximation error in the L_2 norm

$$R_F = \min_{S_m(c,d)} \frac{1}{\bar{\omega}} \int_0^{\bar{\omega}} \left| \sqrt[+]{i\omega} - c_0 - \sum_{k=1}^{m} \frac{c_k i\omega}{i\omega + d_k} \right|^2 d\omega. \tag{3.12}$$

We integrate only with positive ω, since for negative values the function becomes conjugate of itself. The selection of $\bar{\omega}$ requires a special analysis. In all computations we take $\bar{\omega} = 100$ without any proof of the optimality of such a choice.

3.3.3 Approximation of Reflection Coefficient

An interesting approach is proposed in [88]. There a reflection coefficient

$$RC(\omega) = \frac{-\sqrt{\omega} - ic_0 - i \sum\limits_{k=1}^{m} \dfrac{c_k(-\omega)}{-\omega + d_k}}{\sqrt{\omega} - ic_0 - i \sum\limits_{k=1}^{m} \dfrac{c_k(-\omega)}{-\omega + d_k}},$$
$$\omega \in \mathbb{R}, \quad c_i > 0, \quad d_j > 0, \quad i = 0, 1, \ldots, m, \quad j = 1, 2, \ldots, m \tag{3.13}$$

is minimized. The coefficients of the optimal rational function are obtained for $r = -\frac{1}{\omega}$ by solving the following minimization problem with the weight

$$R = \min_{S_m(c,d)} \left(\int_{\delta t/2\pi}^{T/2\pi} \left| \frac{\sqrt{r} - c_0 r - \sum\limits_{k=1}^{m} c_k r/(1 + d_k r)}{\sqrt{r} + c_0 r + \sum\limits_{k=1}^{m} c_k r/(1 + d_k r)} \right|^2 \frac{dr}{1 + r^2} \right), \tag{3.14}$$

where, $1/(1 + r^2)$ is the weight function. In the case of $m = 3$ we get seven coefficients: $c_0, c_1, c_2, c_3, d_1, d_2, d_3$. It is noted in [88], that because δt is small, the reflection coefficient was optimized in the interval $[0, T/2\pi]$ and the following values of coefficients were obtained

$$a_0 = 0.7269284, \quad a_1 = 2.142767, \quad a_2 = 5.742223, \quad a_3 = 46.58032,$$
$$d_1 = 6.906263, \quad d_2 = 65.82243, \quad d_3 = 1124.376. \tag{3.15}$$

3.3.4 The Adaptive Minimization of Errors in the L_2 and L_∞ Norms

In order to test the potential/limitations of the rational functions to approximate the non-local transparent boundary conditions and to investigate the robustness of the selected global optimization algorithms, we have included one more strategy for the selection of objective functions. Two representative benchmark problems with the known exact solutions are selected and the coefficients of rational functions are obtained by minimizing the errors of the discrete solutions in the L_2 and L_∞ norms:

$$E_2 = \min_{S_m(c,d)} \left(\int_0^T \int_A^B |u - \tilde{u}|^2 dx dt \right)^{1/2} = \min_{S_m(c,d)} \|E\|_2, \tag{3.16}$$
$$E_\infty = \min_{S_m(c,d)} \left(\max_{x \in [A,B],\, t \in [0,T]} |u - \tilde{u}| \right) = \min_{S_m(c,d)} \|E\|_\infty,$$

here $E = u - \tilde{u}$.

We note that this technique adapts the coefficients to the selected benchmark problem, when the exact solution is known. Thus the results for these objective functions may lack the universality, but analysis of them can show the limits of the approach based on rational functions.

In the case of M different problems with different solutions $u_j, j = 1, 2, \ldots, M$ we introduce a coupled adaptive strategy

$$E_\infty^c = \min_{S_m(c,d)} \left(\max_{1 \le j \le M} \left(\max_{x \in [A_j,B_j],\, t \in [0,T_j]} |u_j - \tilde{u}_j| \right) \right). \tag{3.17}$$

3.4 Global optimization

The formulated minimization problems (3.14), (3.16) and (3.17) are used as black box problems of global optimization [89]. The evaluation of the values of these objective functions requires numerical integration or even numerical solution of the modelling problem. Thus, the evaluation of the objective functions is computationally expensive. Moreover, the properties of objective functions of these problems are not known, the unimodality or convexity of the objective functions may not be assumed. In such cases local optimization may result in different solutions. Both from the same point when stochastic optimization is used and when different starting points are used. Moreover, intervals of possible values for variables (coefficients) are not known and optimal values of different coefficients may differ by several orders of magnitude. Therefore the search space may be very large and

standard deterministic covering methods for global optimization are not applicable.

However, the initial investigation has shown that local optimization from different starting points sometimes finishes with the same optimal solution found. This substantiates the use of the multi-start strategy for estimating the globally optimal solution [63,89]. Random starting points are generated and a local optimization algorithm is applied from a number of random starting points. Since the objective functions are black boxes analytical expressions for gradients are not available. Moreover, since the objective functions are computationally expensive, the numerical estimation of the gradient is also too expensive. Therefore derivative-free methods for local optimization should be used. We applied one algorithm in our experimental investigations: the Nelder-Mead downhill simplex method [70]. This method was also used in [88].

Since the objective functions of global optimization problems are computationally expensive, the experiments take a considerable amount of time. Parallel computing should be applied to make experiments shorter. The multi-start strategy for global optimization is favourable for parallelisation. Separate runs of local optimization algorithm are independent and therefore may be performed in parallel. Another level of parallelisation may be evaluation of values of the objective function at different sample points.

3.5 Parallel Algorithm

Parallel calculation was used to solve optimization problems for which the computation of objective function requires solution of M different subproblems, with a priori estimated sizes of subproblems. Such optimization problems are often very computationally intensive. Thus, the parallel computations are necessary.

We use the two-level parallel programming template [20,26]. On the first level, we define a set of discrete problems, which can be solved independently in parallel. On the second level, each discrete problem is solved in parallel using the Wang algorithm [94].

On the first level, we calculate solutions for independent M tasks, each of them consists of numerical solving of the Schrödinger equation. However, for different sizes of discretisation of grids J and N may differ – they are tuned to achieve the required precision levels. The amount of calculations is proportional to $J \cdot N$ and we will refer to it as a size of tasks $Z_i, i = 1, 2, \ldots, M$. Thus, the computational sizes of these tasks can be very different. In order

to perform load balancing, we introduce the second level of parallelisation when subproblems are solved using some parallel algorithm In our case subproblems are non-stationary 1D differential equations. At each time step they are approximated by systems of linear equations with a tridiagonal matrix. We use the Wang [94] algorithm to solve the obtained systems. This second level lets us perform the workload balancing at the first level of our algorithm. We assign different numbers of processes for different problems on the second level with different computational costs. Assuming that we use p processes to compute the functional (3.17), we distribute these processes among M tasks by taking the number of processor for each task proportional to $Z_i = J_i \cdot N_i$, where J_i, N_i are discretization sizes in x and t directions, respectively.

The second level can be used alone, however, the efficiency of the Wang algorithm is limited due to the well known speed-up saturation effect, i.e. computations are slowed down due to large communication costs when the number of processes is increased. It can be see from the curves of speed-up's of the Wang algorithm for different sizes of linear systems in Figure 3.1. As we see from the presented curves the speed-up is not linear, if it was linear, the parallelisation of the Wang algorithm alone would be sufficient. Thus, a well-balanced distribution of tasks in two levels should be defined.

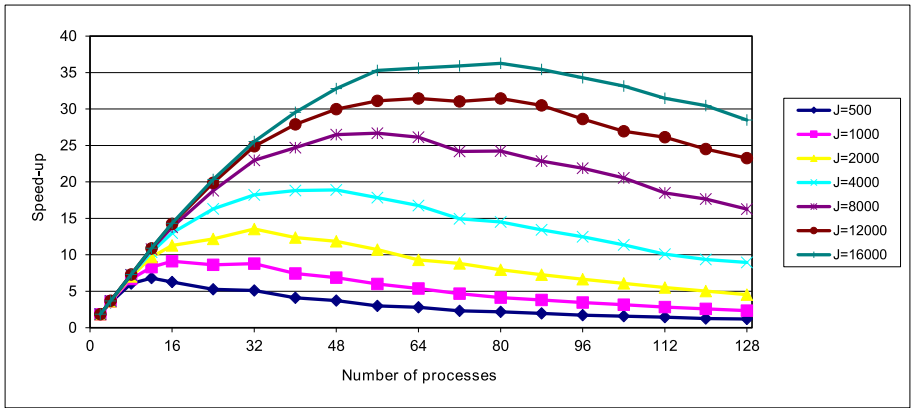


Figure 3.1: The speed-ups of the Wang parallel algorithm for different number of processes p and sizes J of systems.

Load balancing techniques for two level parallelisation are widely used [20, 26]. To test the balancing algorithm we use four differently sized problems: $Z_1 = 3.2 \cdot 10^7$, $Z_2 = 4.8 \cdot 10^7$, $Z_3 = 1.152 \cdot 10^7$, $Z_4 = 9.6 \cdot 10^7$. A simple rule of proportionality to the sizes of problems is provided in the example (see Table 3.1). The workload distribution is not uniform when the number of processes is small. Distribution becomes closer to uniform when we

increase the number of processes. Also, from Table 3.1 it can be seen that increasing the number of processors, leads to the increase of the efficiency of the parallel algorithm. The goal of workload balancing is to ensure that all processors have a similar computations time because the computation of the functional (3.17) is determined by the time of the longest calculated process. When we increase the number of processors, the workload balancing is getting more and more balanced between the groups. As we can see, the last row in Table 3.1 shows that increasing the number of processors increases the efficiency of workload balancing. Hence, the efficiency improvement is the result of the good workload balancing of the algorithm.

Table 3.1: Distribution of processes using simple rule of proportionality to the sizes of problems, efficiency= $\frac{\text{sum of complexities of all tasks}}{\max(\text{complexities of all tasks})}$

p \ j	1	2	3	4	Efficiency
4	1	1	1	1	0,49
8	1	2	1	4	0,73
16	3	4	1	8	0,98
32	6	8	2	16	0,98
64	11	17	4	32	0,98
128	22	33	8	65	0,99

Algorithm 1 *The partitioning function*

1: $getTaskNumAndSize(p, Z, M)$
2: $T = \sum_{i=1}^{M} Z_i$
3: $S_i = floor(\frac{Z_i \cdot p}{T})$, $i = 1, 2, \ldots, M$
4: $rem = size - \sum_{i=1}^{M} S_i$
5: **for** $i = 1 \rightarrow rem$ **do**
6: $\quad w = \arg \max_{i=1,\ldots,M} \frac{Z_i}{S_i}$
7: $\quad S[w] = S[w] + 1$
8: **end for**
9: End

In Algorithm 1 we implement the partitioning function. This algorithm takes these parameters: Z – problem sizes, M – number of tasks, p – number of processors. The complexity of tasks is proportional to size of task $Z_i = J_i \cdot N_i$. Each of process calculates this complexity of all tasks

$$T = \sum_{i=1}^{M} Z_i.$$

The number of processes in the group is chosen so that we get a the uniform distribution of the work:

$$\mu_i = \lfloor Z_i/T \rfloor,$$

$$S_i = max(1, \mu_i \cdot p).$$

The rest of processes is distributed using a greedy algorithm:

$$rem = p - \sum_{M=1}^{M} p_M.$$

The two-level parallel algorithm allows to use significant by more processes comparing to one level parallelisation. However, for a big number of processes p and a small number of problems M two-level parallelisation algorithm may become inefficient. That's why we propose three level approach in Chapter 4, which lets to eliminate the mentioned drawback.

3.6 Numerical Experiments

Two examples, presented in papers [88, 103], are chosen as representative test problems.

Example 1. We use the exact solution of (3.1) (see [88]):

$$u(t,x) = \frac{\exp(-i\pi/4)}{\sqrt{4t-i}} \exp\left(\frac{ix^2 - 6x - 36t}{4t-i}\right). \tag{3.18}$$

The problem is solved in domain $[-5,5]$ for $t \in [0, 0.8]$. It can be noted that the solution has almost compact support in $(-5,5)$ at $t = 0$ and crosses the boundary $x = -5$ for some $t < 1$. In order to avoid the influence of discretization errors we take the uniform grid $J \times N = 8000 \times 4000$.

Example 2. We use the exact solution of (3.1) (see [103]):

$$u(t,x) = \frac{1}{\sqrt[+]{1+it/\alpha}} \exp\left(ik(x - x^{(0)} - kt) - \frac{(x - x^{(0)} - 2kt)^2}{4(\alpha + it)}\right), \tag{3.19}$$

where $k = 100, \alpha = 1/120, x^{(0)} = 0.8$ presented in [103]. In this case we compute the numerical solutions in domain $[0, 1.5]$ for $t \in [0, 0.04]$ and the

uniform grid $J \times N = 12000 \times 4000$ is used.

In all computations the finite difference scheme (3.6) is used and boundary conditions are approximated by (3.10). All error values provided are at least 10 times bigger than the errors of approximation of the Schrödinger equation (3.6). The discretization (3.6) size is enough to keep the errors of approximation of Schrödinger equation, at least 10 times smaller than the values of errors that will be provided in all tables.

Comparison of different approaches. To solve the formulated global minimization problems random starting points are used. For problems (3.12), (3.14) we have taken up to 10000 points. For each parameter the exponential distribution is applied with averages equal to values defined by (3.15).

In the cases of adaptive minimization the evaluation of the objective functions requires solution of the 1D non-stationary discrete problem for each sample point. This step is computationally quite costly. So we have restricted to 100 runs of local optimization even in the case of parallel computations.

The integration in (3.12) and (3.14) is performed by using the Gauss-Lobatto quadrature with adaptive refinement, the accuracy tolerance is equal to 10^{-12}. According to [70], the following parameters are specified for the downhill simplex algorithm: reflection coefficient $\alpha = 1$, expansion coefficient $\gamma = 2$, contraction coefficient $\beta = 0.7$ (used for both – single point and full contractions), the tolerance parameter $\varepsilon = 10^{-10}$ (for (3.14) we have used $\varepsilon = 10^{-15}$).

In Table 3.2 we present coefficients of the optimal rational functions obtained applying different objective functions. It follows from the presented results that the coefficients vary a lot from one approach to another. We only note that adaptive techniques in different norms (for the same test problem) give similar coefficients. So it is hard to formulate any constructive conclusion about the distribution of these coefficients. The Padé coefficients presented in Table 3.2 are calculated by explicit formulas (3.11). It is well known that Padé approximation can be not accurate for small m but the accuracy is improved as m increases. Thus in order to test the potential of Padé approximation we take bigger m to analyze the convergence rate. The results are presented in Table 3.3. It follows from Table 3.3 that the convergence rate for Example 2 is significantly smaller than for Example 1. So the required value of m may be quite large depending on the problem and this can be computationally expensive. Therefore, alternative techniques to

Table 3.2: Coefficients obtained with different approaches

Approach	a_0	a_1	a_2	a_3	d_1	d_2	d_3
Refl. coef. [88](3.14)	0.727	2.14	5.74	46.6	6.91	65.8	1124
Padé [74](3.11)	0	0.357	0.667	4.98	0.0717	1	13.9
FS(3.12)	0.602	1.78	3.53	23.1	4.94	38.0	364
Adapt.(E_∞) Ex. 1	1.01	2.11	3.30	24.5	10.4	51.3	413
Adapt.(E_2) Ex. 1	1.07	2.11	3.48	26.9	10.9	53.5	470
Adapt.(E_∞) Ex. 2	9.30	29.3	7.20	229	0.437	1.01	28847
Adapt.(E_2) Ex. 2	11.3	25.1	4.64	210	0.434	13.7	25567
Adapt.(E_∞^c)	2.91	0	10.6	138	5.26	106	10329

Table 3.3: Padé test for two benchmark problems

Example 1			Example 2		
m	$\|E\|_\infty$	$\|E\|_2$	m	$\|E\|_\infty$	$\|E\|_2$
3	0.179	0.174	3	0.882	1.02e-2
9	1.407e-2	1.395e-2	15	0.627	7.17e-3
15	1.463e-3	1.356e-3	90	7.58e-2	8.30e-4

compute coefficients for small m, e.g. $m = 3$, are needed.

The problems (3.14) and (3.12) are global optimization problems. In our experiments we select a fixed number of starting points and use a local optimization algorithm to descent to the minimum. Then the best local minimum point is used as an approximation of the global minimum point. This strategy is based on the assumption that a smaller value of the objective function means also a smaller error of the discrete solution of the Schrödinger problem. In order to analyse this assumption we have selected 7 random local minima for each functional R and R_F that were found during optimization. By using these coefficients we have solved Examples 1 and 2 and computed the errors of discrete solutions. Thus, it follows from Tables 3.4 and 3.5 that for both examples the errors of discrete solutions monotonically decrease as the values of functionals become smaller. But errors for Example 2 are quite large, so it is hard to make any conclusion on the convergence in this case. It is importat to note, that for Example 2 the L_2 norm is scaled $\|E\|_2^* = \|E\|_2 / \sqrt{T}$.

The comparison of obtained results (the optimization results for universal functionals R and R_F and solution errors in L_∞ and L_2 norms) by using all different techniques is presented in Table 3.6. R and R_F values for adaptive techniques are presented for Example 2, since it gives the biggest values.

Table 3.4: Monotonicity analysis of local minima using the reflection coefficient R

	Example 1		Example 2	
R	$\|E\|_\infty$	$\|E\|_2$	$\|E\|_\infty$	$\|E\|_2^*$
7.29e-06	3.39e-2	3.35e-2	0.712	4.08e-2
7.14e-06	3.68e-2	3.61e-2	0.685	3.92e-2
2.66e-09	1.01e-2	9.74e-3	0.507	2.89e-2
2.41e-09	9.77e-3	9.42e-3	0.505	2.88e-2
2.25e-09	9.57e-3	9.23e-3	0.505	2.87e-2
2.61e-10	5.71e-3	5.36e-3	0.422	2.40e-2
1.40e-10	5.16e-3	4.89e-3	0.452	2.57e-2

Table 3.5: Monotonicity analysis of local minima using the Fourier symbol R_F

	Example 1		Example 2	
R_F	$\|E\|_\infty$	$\|E\|_2$	$\|E\|_\infty$	$\|E\|_2^*$
0.446	5.08e-2	4.62e-2	0.738	4.24e-2
0.161	2.53e-2	2.26e-2	0.746	4.28e-2
8.20e-2	1.46e-2	1.22e-2	0.726	4.16e-2
5.81e-2	1.25e-2	1.08e-2	0.712	4.08e-2
4.27e-2	6.89e-3	6.95e-3	0.702	4.02e-2
3.54e-2	5.87e-3	6.06e-3	0.694	3.98e-2
3.18e-3	2.17e-3	1.56e-3	0.636	3.64e-2

Several conclusions follow from the presented results of computational experiments. First, the heuristic for solving global optimization problems is quite robust. We have no guarantees that the global optimization problem is solved exactly, however, the smallest values for all objective functions are obtained when the global optimization algorithm has minimized the specified objective functional (the best results of functional are marked with bold font). So we have concluded that the obtained computational results are sufficient to formulate our main qualitative conclusions.

Second, considering both universal objective functions (reflection coefficient R and Fourier symbol R_F), the accuracy of both ABCs is similar: the errors of discrete solutions are small for Example 1 and these errors are much larger for Example 2. The two presented techniques are not universal, there is no guarantee that they are suitable for all initial conditions, at least with small $m = 3$. Moreover, from Table 3.6 we see that both functionals R

Table 3.6: Values of all objective functions obtained with different approaches

			Example 1		Example 2	
Approach	R	R_F	$\|E\|_\infty$	$\|E\|_2$	$\|E\|_\infty$	$\|E\|_2^*$
Refl. coef.(3.14)	**1.40e-10**	2.15e-2	5.16e-3	4.89e-3	0.452	2.57e-2
FS(3.12)	4.62e-06	**3.18e-3**	2.18e-3	1.56e-3	0.636	3.64e-2
Adapt.(E_∞)	0.106	1689	**3.21e-4**	2.75e-4	**9.43e-3**	6.13e-4
Adapt.(E_2)	0.100	1255	7.19e-4	**2.58e-4**	1.53e-2	**4.09e-4**
Padé	2.06e-3	16.2	0.179	0.173	0.882	5.07e-2

and R_F obtain big values when we adaptively optimize these coefficients for Example 2. Although $m = 3$ is small so this result was expected – adaptive error minimization cannot guarantee small values of general functionals.

Third, the adaptive techniques for Example 2 gave sufficiently small errors of the discrete solution. So even for this example an accurate approximation by small order rational functions is still possible. However, this adaptive technique fits the coefficients to the selected example and the obtained optimal coefficients can give considerably bigger errors for other initial conditions.

We investigate the suitability of ABCs approximation by rational functions technique in general case. In Table 3.7, a cross-check of adaptive techniques for different examples is done, i.e. errors of discrete solutions are estimated for both examples using coefficients that were obtained applying adaptive techniques for both benchmarks.

Table 3.7: Cross-check of the accuracy for different examples

		Example 1		Example 2	
	Approach	$\|E\|_\infty$	$\|E\|_2$	$\|E\|_\infty$	$\|E\|_2^*$
Example 1	Adapt.(E_∞)	3.21e-4	2.75e-4	0.600	3.43e-2
	Adapt.(E_2)	7.19e-4	2.58e-4	0.596	3.40e-2
Example 2	Adapt.(E_∞)	0.594	0.560	9.43e-3	6.13e-4
	Adapt.(E_2)	0.576	0.542	1.53e-2	4.09e-4
coupled	Adapt.(E_∞^c)	4.44e-2	3.73e-2	4.44e-2	2.64e-3

It follows from the results presented in Table 3.7, that errors of the discrete solutions are small when the specific norm for the same solution is used as an objective function. The type of the norm is not very important in computation of optimal coefficients for the given benchmark problem. At the same time errors of the discrete solution are big for the remaining test problem. In order to test the approximation accuracy of the family

of rational functions with $m = 3$ we have used one more objective function (3.17), when a coupling of errors of both benchmark problems is considered. The results presented in Table 3.7 show that this objective function enables to find coefficients giving small enough errors for both benchmarks. Thus a coupled adaptive strategy still allows us to find an efficient compromise among two discrete solutions.

3.7 Conclusions of the Third Chapter

Two universal techniques for obtaining coefficients, approximation of the Fourier symbol and minimization of the reflection coefficient, resulted in small errors of the same order in the case of the first test problem. However, these techniques gave big errors for the second test problem. This means that these techniques do not suit all possible problems.

Minimization of the actual error gave a significant increase in accuracy of calculations. However, this technique is tuned only to selected test problem.

The coupled adaptive technique for obtaining values of coefficients showed that it is possible to find values that suit both test problems. However, it is unclear, if such values exist in a general case with any initial conditions. In Chapter 4 it is demonstrated that it is also possible to find values for four test problems.

The two-level parallel algorithm allows us to use significantly more processes comparing to one level parallelisation. However, for a big number of processes p and small number of problems M two-level parallelisation algorithm may become inefficient. That's why we propose a three-level approach in Chapter 4, which lets to eliminate the mentioned drawback.

Chapter 4

Three-level parallelisation scheme for optimizations problems*

We propose a general methodology for solving problems when there is a big number of processes available. We investigate a three-level parallelisation algorithm for optimization problems, different parallelisation levels create different possibilities but also challenges. At the first level of parallelisation we assume that there exist parallel alternatives to the original sequential modelling algorithm. The first level of parallelisation becomes a part of a new parallel algorithm and the degree of parallelism can be selected dynamically during the computations. The parallelisation speed-up on the first level is not linear, it can reduce the efficiency of the whole parallelisation. But this level enables to use many much more processes and to solve the given problem faster than two-level parallelisation. In this chapter as an example we consider the parallelised simplex downhill method. On the second level, a set of computational tasks with different computational sizes is defined. The work amount distribution between tasks is non-uniform – this makes the parallelisation challenging. This leads to the necessity of third level, because a proper load balancing must be performed. As an example we investigate the case when M partial differential equations are solved. The computational sizes of these tasks are non-equal because different discretisation grids must be used for each equation in order to achieve the same level of accuracy. The third level defines parallel algorithms to

*Kriauzienė, R.; Bugajev, A.; Čiegis, R. A three-Level parallelisation scheme and application to the Nelder-Mead Algorithm. 2019, http://arxiv.org/abs/1904.05208

solve tasks from the second level. As an example we take Wang's algorithm to parallelise the solution of systems of linear equations with tridiagonal matrices [94]. This level can be used alone, however, it is limited due to Amdahl's law. We presented a general methodology, which combines the parallelisation of a local optimization algorithm with a standard two-level parallelisation.

Parts of this chapter are published in [53].

4.1 Introduction into this Chapter

Current trends in supercomputing show that in order to accumulate high computing power, computers with more, but not faster, processors are used. This trend induces changes in the development of parallel algorithms. The important challenge is to develop parallelization techniques which enable exploitation of substantially more computational resources than the standard existing methods.

This part of the dissertation deals with problems that can be split into a collection of independent subproblems and this splitting step is repeated iteratively. The solutions of subproblems define the solution of an initial problem. Thus, an additional splitting step increases the potential parallelisation degree of a parallel algorithm.

Any multi-level parallelisation can be considered as a way to generate a pool of tasks. After the pool of tasks is obtained, it is not important how many parallelisation levels were used. However, often such final simplification of the template leads to a loss of important information and as a consequence to degraded efficiency of the parallel algorithm. Especially this is true if different levels of the scheme are characterised by different properties of an algorithm that should be properly addressed.

We consider a special case of a three-level parallelisation. The template of this approach is given in Fig. 4.1:

- At the first level of parallelisation we assume that there exist a few parallel alternatives A_j (see Figure 4.1) to the original modelling algorithm. The first level of parallelisation becomes a part of a new parallel algorithm and the degree of the first level parallelism can be selected dynamically during the computations – a selection of the best algorithm is performed. As an example we consider two new parallel modifications of the Nelder-Mead method [69].

- On the second level, a set of computational tasks $V^j = \{v_1^j, v_2^j, \ldots, v_{M_j}^j\}$ (see Figure 4.1) with different computational complexities is defined. These tasks are solved in parallel. As an example we investigate the case when computation of one value of the objective function requires to solve numerically M partial differential equations. The computational complexities of tasks are non-equal because different discretisation steps must be used for different equations in order to achieve the same accuracy for each equation.

- The third level defines parallel algorithms to solve tasks from the second level. As an example we use the Wang algorithm to parallelise the solution of systems of linear equations with tridiagonal matrices [94].

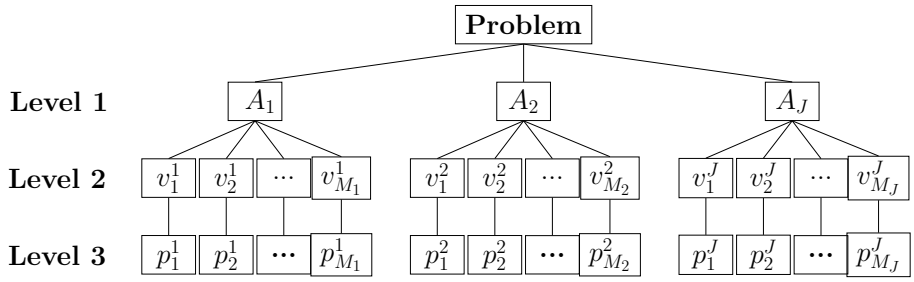


Figure 4.1: Three level parallelisation scheme.

The second and the third levels define a well-investigated two-level parallelisation template. We note that load balancing techniques for two-level parallelisation are widely used in applications, see, e.g., [20], [47].

The scheduling problem can be formulated representing a parallel algorithm by a directed acyclic graph (DAG). The vertices define computational tasks, the edges define connections/order among tasks. Then a set of partially ordered computational tasks is scheduled onto a multiprocessors system to minimise the computational time (or to optimise some other performance criteria). It is well known that the scheduling problem is NP complete. Many interesting heuristics are proposed to solve it, we mention greedy algorithms [22], genetic algorithms [84], [85], simulated annealing and tabu search algorithms [51], [41], [42]. Such algorithms include a possibility of dynamic scheduling and allow for tasks to arrive continuously and they can consider variable in time computational resources.

A scheduling task can be very challenging due to the specificity of a given application problem and the necessity to parallelise it on modern parallel

architectures. As an example we mention the particle simulation which is solved by appropriate domain decomposition techniques [40]. Another example is the dynamic load balancing on heterogeneous clusters for parallel ant colony optimization [60]. In the recent work [30] it is concentrated on the problem of high-dimensionality of the data while solving subspace clustering problem.

In this article we focus on the scheduling problem, when all tasks in the set are independent and can be solved in parallel. It is well known that the given optimization problem can be redefined as a problem to equalise the computational times of all processes. The simplest load balancing algorithm is based on the assumption that the computation time is proportional to the sizes of sub-tasks. Then the domain decomposition algorithm is applied to guarantee that the sizes of subtasks scheduled for each group of processors are equal [20].

The quasi-optimal distributions of tasks can be obtained using the greedy strategy to distribute the work on demand, i.e. to apply dynamic load balancing techniques such as work-stealing [49], self-organising process rescheduling [81].

However, the efficiency of two-level approach is limited due to a typical saturation of the speed up of parallel algorithms for increased numbers of processors and fixed sizes of tasks. Exactly this situation has motivated us to introduce an additional level of parallelisation template. In most cases its usage leads to a less efficient algorithms than the initial state-of-the-art algorithm. But the additional degree of parallelism on the second level gives a large overall speed-up, if the number of available resources is large.

Recent developments of new architectures of parallel processors make even more challenging the task to build accurate theoretical performance models. The empirical data shows that for some advanced algorithms the efficiency of parallel computations can depend non-monotonically on the size of a task. Thus the model-based load balancing method starts to become the main tool in developing efficient and accurate task scheduling algorithms. In our work we build the model for prediction of computation time empirically by solving the specialised benchmarks for a wide range of problem sizes and numbers of processors. In fact this analysis resembles the classical experimental strong scalability analysis of a given parallel algorithm. We note, that these measurements are always done for all processes working simultaneously in order to reflect their actual performance during the execution of real applications (see, also [55, 56]).

Here we mention two interesting papers, where the model-based task

scheduling algorithms are considered. In [56], it is concentrated on multi-core co-processors Xeon Phi, where the empirical computation time curves are used to find optimal parameters for a workload distribution. The obtained model predicts non-monotonic dependence of computation speed on the sizes of problems. The authors call their approach "load imbalancing", however, it can be considered as an advanced balancing which adapts the scheduling algorithm to the specificity of Xeon Phi processors. Obviously in this case the assumption that computation time is proportional to the task size is not valid. In a similar research [55], computations were performed on non-uniform memory access (NUMA) parallel platform with various shared on-chip resources such as Last Level Cache. Again the model-based approach enables to take into account the specific properties of the algorithm and processors. The matrix multiplication and Fast Fourier Transform are used as benchmark problems. It is interesting to note that, according to the presented results, the globally optimal solutions may not load-balance the sizes of sub-tasks. The authors pay a special attention to the energy efficiency of calculations. We note, that there are some papers that are specifically dedicated to load balancing of energy efficiency [75]. In our work we formulate some restrictions that are connected to energy efficiency as well – we do not use additional available computational resources if the parallelisation efficiency drops below some specific level. The other work [80] is dedicated to model-based optimization on hybrid heterogeneous systems composed of CPUs and accelerators. In that research the authors investigate the problem of communications costs due to uneven workload distribution between accelerators and CPUs. They propose to generalise the τ-Lop [79] model for heterogeneous computations.

In this part of the dissertation we propose a general methodology for parallelisation of algorithms. As an example we use it to solve some applied optimizations problems. is shown The superiority of the three level parallelisation scheme is shown, comparing it with two-level paralleisation scheme. On the second level a set of different-size tasks is defined, which is a typical situation for computation of one value of a black box objective function. In most cases these tasks (or groups of tasks) are independent but computationally costly. Thus each task also should be solved in parallel. This fact leads to a necessity of the third level. The second and third levels of the template define a set of tasks solved in parallel and some load balancing algorithm should be used to take into account the different sizes of subtasks. The necessity of the additional first level comes from the assumption of having more computational resources than can be utilised by the

two-level parallelisation approach. It is a consequence of the efficiency saturation for parallel algorithms when the size of the problem is fixed and the number of processes is increased. We select a different optimization method (or a modification of the basic solver) which gives additional degrees of parallelisation thus enabling the possibility to use more processors. At the first level of the template the optimal algorithm is selected. This part requires finding a compromise between the increased parallelisation degree and the decreased convergence rate of the modified parallel optimization algorithm.

In this work we are also interested to address some green computing (GC) challenges. In a broader sense GC is the practices and procedures of designing, manufacturing, using of computing resources in an environment friendly way while maintaining overall computing performance and finally disposing in a way that reduces their environmental impact [82]. The research in green computing is done in many areas [71]: Energy Consumption; E-Waste Recycling; Data Center Consolidation and Optimization; Virtualization; I. T Products and Eco-labeling. One of approaches for optimization of energy consumption on the software level is the autotuning software, which is able to optimise its own execution parameters with respect to a specific objective function (usually, it is execution time) [16]. Well known examples of autotuning software are: FFTW [38] (fast Fourier transformations); ATLAS [96], PHiPAC [11] (dense matrix computations); OSKI [93], SPARSITY [48] (sparse matrix computations).

Usually, the goal for any autotuning software is to achieve the same result with the same resources, however, reducing the computation time – in terms of parallelisation it means to increase the parallelisation efficiency. Another way to decrease the power consumption is to increase the efficiency by avoiding inefficient calculations; this may slightly increase the execution time, however will give a reasonable increase of parallel efficiency, which leads to the energy savings. We propose to control the efficiency of the parallel algorithm on the load balancing stage of the parallelisation template. In many cases this strategy reduces the amount of computational resources used in computations. This analysis is done a priori, meaning that the user knows how many cores should be used for solving a specific parallel task even before starting real computations.

4.2 Workload Balancing Problem

In this section we formulate the workload balancing problem for the two-level parallelisation. Also we present a greedy scheduling algorithm to dis-

tribute the processes among tasks. Next, we introduce the additional level – the first and second levels of the two-level parallelisation technique becomes the second and the third levels, accordingly and the first level is a new parallelisation level. On the first level the selection of the optimal algorithm is performed.

First, we will present two-level parallelisation template. Let's assume that we solve a given problem by using the basic method A. The solution process consists of K blocks of tasks (a simple DAG)

$$A = \{V_1 \prec V_2 \prec \ldots \prec V_K\}, \tag{4.1}$$

and all blocks must be solved sequentially one after another. Each block consists of M tasks

$$V_k = \{v_1(X_k), v_2(X_k), \ldots, v_M(X_k)\}, \quad k = 1, \ldots, K,$$

where X_k defines a set of parameters for the V_k block. V_k defines the first level of the two-level parallelisation scheme. Each task v_m can be solved by a parallel algorithm – this is the second level of the scheme.

The complexities of tasks v_m are different, however, they are known in advance and do not depend on k. For each task v_m the prediction of computation time $t_m(p)$, $p \leq P$, $m = 1, \ldots, M$ is given – it is based on the modelling results, P is the number of processors in a parallel system. We assume that up to P_m processes the computation time monotonically decreases:

$$t_m(p_2) < t_m(p_1), \quad \text{for } p_1 < p_2 \leq P_m. \tag{4.2}$$

For P_m the predicted computation time function $t_m(p)$ reaches the minimum value:

$$t_m(p) \geq t_m(P_m), \qquad p > P_m. \tag{4.3}$$

Such a model of computation time $t_m(p)$ is important for algorithms with limited scalability such as Wang's algorithm. In Fig. 3.1 we present speed-ups of this algorithm for different sizes of linear systems. It is important to mention that the provided results include some additional costs for computation of the objective function along with Wang's algorithm computational costs. These additional calculations slightly increase the overall parallelisation scalability, thus the provided figure represents the optimistic scenario for general Wang's algorithm and the realistic scenario for actual computations, that were done.

In our specific case this data was derived from a simple benchmark

implementing Wang's algorithm. This benchmark performs computations using different numbers of processes and different problem complexity parameters J. It is important to note, that nodes were artificially loaded with calculations to imitate the real situation. For example, with the number of processes $p = 4$ there were 32 tasks that were solved by 128 processes at the same time. Thus this benchmark must be run once, using all processes available.

From Figure 3.1 it follows that the computation time monotonically decreases till some critical number of processes and therefore the efficient usage of processes is limited to this number of processes. Even for large size systems, when the number of equations is $J = 16000$, the maximum number of processes P_m does not exceed 80. This analysis justifies our motivation to use the multi-level approach in order to solve the given applied problem.

In the two-level parallelisation scheme for each block of tasks V_k we select the number of processes such that the overall solution time is minimised:

$$\arg \min_{(p_1,\ldots,p_M)\in Q} \max_{1\le m\le M} t_m(p_m),$$

where a set of feasible processors distributions Q is defined as

$$Q = \{(p_1, \ldots, p_M) : p_1 + \ldots + p_M \le P\}.$$

In the case when we solve only few large size tasks and the remaining tasks are much smaller and the number of processes P is not very big, the optimal scheduling is obtained when a few smaller tasks are combined into one group $\widetilde{v}_m$. Then sub-task $\widetilde{v}_m$ consists of tasks $v_{l_1}, \ldots, v_{l_n}$. The computation time for this combined task is predicted by the model:

$$\widetilde{t}_m(p_m) = \sum_{i=1}^{n} t_{l_i}(\widetilde{p}_{l_i}), \quad \widetilde{p}_{l_i} = \min\left(p_m, P_{l_i}\right).$$

In this work we are interested to solve the scheduling problem, when the number of processes is large, so the aggregation step is not used.

Next, we propose a simple greedy partitioning algorithm, which is described in Algorithm 4.2. It aims to find a near-optimal distribution of M tasks of different sizes between homogeneous P processes by using the model-based complexity model $t_m(p)$ (similar ideas are also used in [55]). We assume that $P \ge M$. The interesting feature of the presented algorithm is that for a given number of processes P the number of active processes can be taken less than P to minimise the overall execution time of the parallel

1: Set $p[m] = 1$, for $m = 1, \ldots, M$
2: $P = P - M$
3: Compute $t_m(p[m])$, for $m = 1, \ldots, M$
4: stop = 0
5: **while** $P > 0$ & stop == 0 **do**
6: find j such that $t_j(p[j]) = \max\limits_{1 \le m \le M} t_m(p[m])$
7: **if** $p[j] == P_j$ **then**
8: stop = 1
9: **else**
10: $p[j] = p[j] + 1$
11: $P = P - 1$
12: **end if**
13: **end while**

Figure 4.2: The algorithm for distribution of P processes between M tasks

algorithm.

The algorithm starts from the initial distribution when one process is assigned for each task and the predictions of parallel execution times are calculated using the selected performance model. Then, the greedy iterative procedure is applied to distribute the remaining processes. At each iteration, one additional process is assigned to the task which has the largest predicted computation time. Then its parallel execution time is updated. Iterations are repeated until all processes are distributed or the number of processes for some task reaches the limit P_m.

Note, that before $t_m(p)$ has reached the minimum, value starts to decrease slowly, thus the parallelisation efficiency drops. Therefore, it may be wise to restrict the number of processes by taking into account the efficiency value.

We define the maximum number of processes $\widetilde{P}_k$ for which the efficiency condition is still satisfied

$$E_p(V_k) \ge E_{min}, \quad \text{for } p \le \widetilde{P}_m, \tag{4.4}$$

where $E_p(V_k) = S_p(V_k)/p$ ir $S_p(V_k) = t_k(1)/t_k(p)$, $E_{min} \in [0, 1]$ is a given efficiency lower bound. Estimate (4.4) is used to modify the limit of the maximum number of processes (4.3) that can be used to solve the j-th task

$$P_m = \min(P_m, \widetilde{P}_m). \tag{4.5}$$

Therefore, in the presented technique P_m includes two restrictions:

- The number of processes cannot exceed the number after which the speed-up drops down (see Fig. 3.1).

- The number of processes is limited by efficiency requirement (4.4), which states: the number of processes per block of tasks V_k is not allowed to be increased if the efficiency of the parallel algorithm on the third level reaches the critical value E_{min}.

In fact the second level of the two-level scheme can be used alone, however, it is limited due to Amdahl's law [3], i.e. the efficiency begins to drop as the number of processes increases for a fixed size of problem. The two-level approach let us solve this issue up to some point.

Exactly this situation has motivated us to introduce an additional level of parallelisation template.

In the new three-level parallelisation scheme, the second and third levels represent the two-level scheme part described before. Additionally, we add a new first level of the template. We assume that there exist parallel alternative algorithms A_j:

$$A_j = \{V_1^j \prec V_2^j \prec \ldots \prec V_{K_j}^j\}, \quad j = 1, \ldots, J.$$

Each block V_k^j consists of M_j independent tasks

$$V_k^j = \{v_1^j(X_k), v_2^j(X_k), \ldots, v_{M_j}^j(X_k)\}, \quad k = 1, \ldots, K.$$

The numbers of blocks of tasks K_j, the numbers of tasks per block M_j, the sizes of tasks $|v_m^j|$ may be different for different j.

Next, we select the optimal algorithm according to the number of resources available. We denote

$$T_P(A_j) = T_P(V^j)K_j$$

the total solution time for algorithm A_j. The block of tasks V^j is solved by using the heuristic proposed above. Then the optimal algorithm is defined as

$$\arg \min_{1 \le j \le J} T_P(A_j).$$

The usage of $j > 1$ may lead to a less efficient algorithm than the initial basic algorithm. But the additional degree of parallelism gives a large overall speed-up.

4.3 Application of the Three-Level Parallelisation Scheme

The three level parallel partitioning algorithm proposed in Section 4.2 to solve local optimization problems using the well-known Nelder-Mead algorithm [69] was applied.

The aim is to find optimal values of parameters $\{a_0, a_1, \dots a_l, d_1, a_2, \dots d_l\}$, when the following minimisation problem is solved

$$\begin{aligned} E^c_\infty &= \min_{\{a_k, d_k\}} V_k, \quad V_k = \max_{1 \le m \le \tilde{M}} v_m(X_k), \\ v_m &= \max_{j \in [0, J_m],\, n \in [0, N_m]} \left| u(x_{j,m}, t^n_m) - U^n_{j,m} \right|, \end{aligned} \tag{4.6}$$

and $\widetilde{M}$ specially selected benchmark PDEs are solved.

In all examples we use $l = 3$, i.e., the dimensionality of the optimization problem (4.6) is equal to 7. Here discrete approximations of PDEs represent the tasks v_m in (4.1). To solve v_m we must find solutions of N systems of linear equation with tridiagonal matrix [15]. According to our three-level parallelisation scheme, the calculations of a single point in minimisation problem (4.6) define the block of tasks V_k.

The systems of linear equations with tridiagonal matrices are solved using Wang's algorithm. It is well known that if the size of a system is J and p processes are used then the computation time can be estimated as

$$T_{Wp} = 17\frac{J}{p} + 8p + T_{c1}(p), \tag{4.7}$$

where $T_{c1}(p)$ defines communication costs. The time to compute a value of the objective function f for the specified equation can be estimated as

$$T_{Op} = c_1\frac{J}{p} + T_{c2}(p). \tag{4.8}$$

In this work instead of theoretical complexity models (4.7) and (4.8) we use $t_m(p)$, $m = 1, \dots, M$, based on empirical computations for a selected set of benchmark problems. Such an approach takes into account all specific details of the parallel algorithm and the computer system.

It is interesting to note that the complexity of computational task v_m depends on both parameters: the number of linear equations J_m of the system and the number of integration in time steps N_m. The computation time T_{mp} is equal to $N_m t_m(p)$, but the scalability of the parallel algorithm

depends on J_m only, since the integration in time is done sequentially step by step.

Next, we have solved an example with $M = 4$, where four different benchmark PDE problems (3.2) with explicit solutions [88, 103] are defined in Chapter 3:

1.
$$u(t,x)=\frac{\exp\left(-i\pi/4\right)}{\sqrt{4t-i}}\exp\left(\frac{ix^2-6x-36t}{4t-i}\right), \tag{4.9}$$
$x \in [-5,5]$, $t \in [0,0.8]$. The problem is approximated on the uniform grid $J \times N = 8000 \times 4000$.

2.
$$u(t,x) = \frac{1}{\sqrt[+]{1+it/\alpha}}\exp\Big(ik(x-x^{(0)}-kt) - \frac{(x-x^{(0)}-2kt)^2}{4(\alpha+it)}\Big), \tag{4.10}$$
where $k = 100, \alpha = 1/120, x^{(0)} = 0.8$. $x \in [0,1.5]$, $t \in [0,0.04]$. We use the uniform discretisation grid $J \times N = 12000 \times 4000$.

3. The solution is defined by (4.9), $x \in [-10,10]$, $t \in [0,2]$. We use the uniform discretisation grid $J \times N = 16000 \times 10000$.

4. The solution if defined by (4.10), where $k = 100, \alpha = 1/120, x^{(0)} = 0.8$. $x \in [0,2]$, $t \in [0,0.08]$. We use the uniform discretisation grid $J \times N = 16000 \times 8000$.

Next, we consider the problem (4.6) as a local optimization problem, which can be solved using an iterative algorithm with a given initial starting point. As a local optimiser Nelder-Mead algorithm is used [69].

We propose a family of modifications of the original Nelder-Mead (see Figure 4.3) algorithm in order to increase the parallelisation degree of it. At each iteration the following four different scenarios can be obtained:

- Reflection – compute the value f_R of the objective function at the point X_R. Depending on the value f_R this can be the end of the iteration.

- Expansion – depending on the f_R, an additional computation of the objective function at the point X_E is done, meaning the total computation of two objective function values: f_R, f_E.

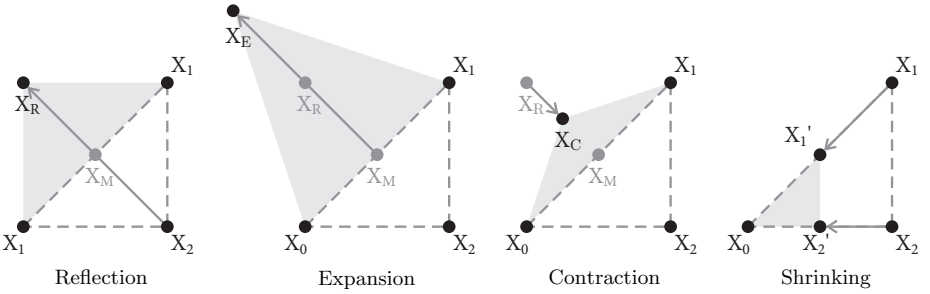


Figure 4.3: One possible step of the Nelder-Mead algorithm applied to a problem in $\mathbb{R}^2$ [95], here X_i is vertices, X_M – centroid of two worst vertices.

- Contraction – depending on the f_R, an additional computation of the objective function at point X_C is done, meaning the total computation of two objective function values: f_R, f_C.
- Compression – compute m objective function values, as well as f_R and f_C. Here m is the number of simplex dimensions.

The first three scenarios require to compute one or two values of the objective function from the set: f_R, f_E, f_C. We can neglect the last scenario, because it occurs very rarely. For the first three scenarios we propose to compute two or three points simultaneously. Algorithmically this means that we change the order of computations, which let us to parallelise the Nelder-Mead method. In most cases only two of three points will be used. Therefore, some redundant calculations will be performed, however, this modification gives an additional parallelisation of computations.

Thus, two modifications of the sequential (A_1) Nelder-Mead method are defined. For A_2 we compute in parallel values f_R, f_E and for $A_3 = 3$ we compute in parallel all three values f_R, f_E, f_C. As a test case we assume that the first scenario is relatively rare, the extension step is done with probability 2/3 and contraction steps occurs with probability 1/3. Then we get that the algorithmic efficiency of the proposed parallel modifications are equal to $\gamma_2 = 0.75$ and $\gamma_3 = 2/3$, respectively. We note, that these values can be estimated more precisely for specific applications, and one example is given for the computational experiments with the Rosenbrock objective function in Section 4.5.

In some cases the Nelder-Mead method is not the best method for local optimization problem. In paper [64], it was shown that the method converge to a non-stationary point. However, for the most practical problems it gives good results with the reasonable amount of computations. Thus it is widely

used, that's why we used it as an example for applications of our methodology. In our studied case this gives sufficiently good solutions. It is important to note, that our methodology can be applied to other method such as the Spendley Hext and Himsworth simplex [86], Hooke-Jeeves algorithm [50], etc.

On the first level different parallel algorithms can be used, however, the proposed approach is oriented to the cases when the increased degree of parallelisation gives the speed-up at the cost of efficiency which is a typical situation in parallel algorithms theory (Amdahl's law). As one more example we mention new algorithms developed to solve the global optimization problems. The modification of the well-known DIRECT method [37] was presented in [87], it is called DIRECT-GL. The new modification is based on the idea at each iteration to analyse more potential optimal rectangles. This approach increases the global sensitivity of the method but in many cases this property is achieved at the cost of additional computations. The potential parallelisation degree of the DIRECT-GL algorithm can increase up to 2-3 times. But the results of computational experiments in [87] show that for many benchmark problems (in [87] these cases are numbered 1,2,5,6,20,21,22,24,35,37,38,47,48,49,52) the DIRECT-GL algorithm increases the computational costs to achieve the same accuracy of approximations as DIRECT algorithm. Thus, the classical DIRECT algorithm and its modification DIRECT-GL fit well into the proposed three-level parallelisation template. Then the degree of parallelisation should be increased only if this increasement compensates the reduced efficiency of the modified algorithm. Thus we state, that in order to apply the proposed three level parallelisation scheme, first the computations of one point should be parallelised by a two-level parallelisation approach. Then alternative cases of parallel algorithms with additional degrees of parallelisation should be identified and the optimal algorithm should be selected.

4.4 Experimental Results

In this section we present the results of the parallel scalability tests. All parallel numerical tests in this work were performed on the computer cluster "HPC Sauletekis" at the High Performance Computing Center of Vilnius University, Faculty of Physics. We used up to 8 nodes with Intel® Xeon® processors E5-2670 with 16 cores (2.60 GHz) and 128 GB of RAM per node. Computational nodes are interconnected via the InfiniBand network.

Our main goal is to investigate the efficiency of the proposed three level

template of workload distribution between processes. First, we selected three specific benchmarks with different discretizations (3.4), when $M = 4$ discrete approximations of PDEs (3.6) are solved numerically to compute one value of the objective function. The sizes $(J_m \times N_m)$, $m = 1, \ldots, 4$ of discrete problems are given in Table 4.1.

Table 4.1: Benchmarks with different sizes $J_m \times N_m$ of the discrete problem (3.6)

	Benchmark 1	Benchmark 2	Benchmark 3
Eq.	Sizes	Sizes	Sizes
1	8000×40000	8000×20000	8000×10000
2	4000×20000	4000×20000	2000×20000
3	2000×20000	4000×10000	2000×10000
4	2000×10000	2000×10000	1000×20000

In the first benchmark the size of one task v_1 is much bigger than the sizes of the remaining three tasks. In the second benchmark two changes are made. They make this set of tasks more suited for parallelisation on a large number of processes: the size of task v_1 is reduced twice by taking a smaller number of time steps N_1; the size of task v_3 remains the same, but the number of points J_3 is increased twice, therefore the scalability of Wang's algorithm is improved for this task. In the third benchmark the relative sizes of tasks v_m are more homogeneous than in the first benchmark, but this result is achieved by reducing the number of space grid points J_2, J_4, therefore the scalability of Wang's algorithm is decreased for these two tasks, especially for v_4.

First, we exclude the efficiency condition from the load balancing algorithm by taking $E_{min} = 0$ in (4.4). The distribution of processors between tasks is presented in Tables 4.2–4.4. We also provide the actual computation time T_p along with T_{Mp} that were predicted by the theoretical complexity model. As we can see from Table 4.2 the model and experimental times are close to each other. The experimental time is smaller in cases when there is no interpolation error. Also it is smaller than the model time – it is an expected result, model times (see Figure 3.1) are based on benchmark, that imitate pessimistic scenario – as it was mentioned before, all nodes were artificially loaded at the same time. The prediction accuracy depends on many parameters such as cluster architecture, network loads during computations.

For comparison purposes we provide the results obtained by using the two-level parallelisation template. $K = 1$, then the first level of the three-

level template is not used.

It is important to note, that in Tables 4.2-4.5 we present the CPU time needed to compute one useful point (4.6), i.e., the actual time is divided by $\gamma_k\, k$, which represents the usefulness of computations. Optimal algorithm A_k is selected automatically using the approach that was described above.

As it follows from Table 4.2, the usage of the first level with $k = 3$ and $P = 128$ processes increases the potential speed-up from 38.75 to 60.44. If $P = 128$ and $k = 1$ then only 70 processes are used. However the result is very similar to the case when $P = 64$ processes are used, which means that these additional resources are used very inefficiently.

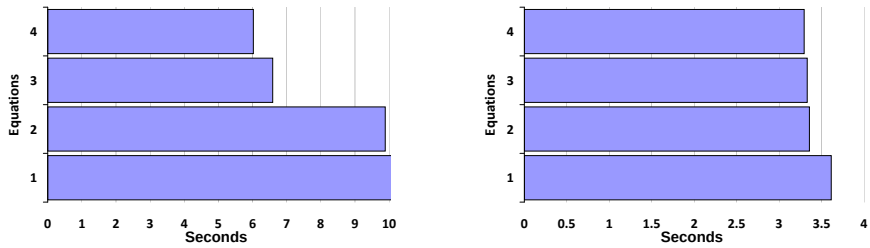


Figure 4.4: Experimental model times for Benchmark 1 with $p = 16$(left) and $p = 64$(right)

In the Figure 4.4 the Gantt charts show theoretical model time $t_m(p)$, that is needed to obtain the solutions of different equations. The workload distribution becomes closer to uniform as the number of processes is increased.

4.4.1 The Control of Efficiency

The reduction of the energy consumption is an important goal, especially when increasement of computation speed-up are small for additional processes. The presented results indicate that in some cases there is a highly inefficient usage of computational resources.

For the purposes of controlling the efficiency of calculations the condition (4.4) was introduced in Algorithm 4.2. This condition guarantees that the efficiency of the numerical solution of each block of tasks will be at least E_{min}. It is important to note, that we are not attempting to generate optimal mappings of processors – we have developed an heuristic that provides the quality of distribution of tasks, which is sufficient for the most practical purposes. The quality of the algorithm is improved when more processors

Table 4.2: The results for Benchmark 1. T_p is the CPU time in seconds required to compute one useful point (4.6)

p	16	32	64
		$k = 1$	
Eq. 1	10	22	50
Eq. 2	3	5	8
Eq. 3	2	3	4
Eq. 4	1	2	2
Total number of p	16	32	64
Model T_{Mp}	11.145	5.784	3.614
T_p	11.003	5.394	3.608
Speed-up	12.679	25.862	38.664

p	96	128	128
	$k = 2$	$k = 3$	$K = 1$
Eq. 1	34	29	56
Eq. 2	8	7	8
Eq. 3	4	4	4
Eq. 4	2	2	2
Total number of p	96	126	70
Model T_{Mp}	2.742	2.272	3.605
T_p	2.719	2.308	3.600
Speed-up	51.307	60.444	38.75

are available.

Next, a more detailed analysis of the Benchmark 1 is provided. In Table 4.5 the results for $E_{min} > 0$ are presented. Comparing the results in Table 4.5 with the results in Table 4.2 we see that for $K = 1$ and $E_{min} = 0.6$ the number of processes for the first equation is decreased by 14, however, the computation times are almost the same as it was in the case of $E_{min} = 0$. Also, for $K = 3$ the efficiency requirement begins to limit the number of processes for $E_{min} = 0.75$ and it decreases further with $E_{min} = 0.8$.
However, even then a three level approach with $K = 3$ is superior to the standard two-level approach in terms of the final speed-up. The results in Table 4.5 indicate that even for the efficiency limitation $E_{min} = 0.75$ the proposed three-level approach lets us maintain a big number of parallel processes active, this number is equal to $(26+7+4+2)\times 3 = 117$. The speed-up is 56 and the efficiency of the parallel algorithm is $56/117 \approx 0.48$. The last column in Table 4.5 with $K = 1$ presents the results for the two-level approach (without the first level). A straightforward two-level parallelisation

Table 4.3: The results for Benchmark 2. T_p is the CPU time in seconds required to compute one useful point (4.6).

p	16	32	64	96	128	128
		$k = 1$		$k = 2$		$K = 1$
Eq. 1	9	18	37	26	37	56
Eq. 2	4	8	15	12	15	18
Eq. 3	2	4	8	7	8	8
Eq. 4	1	2	4	3	4	4
Total number of p	16	32	64	96	128	86
Model time	6.59	3.36	2.01	1.65	1.34	1.8
T_p	6.69	3.37	1.98	1.62	1.33	1.86
Speed-up	13.6	27.03	46.03	56.24	68.25	49.03

Table 4.4: The results for Benchmark 3. T_p is the CPU time in seconds required to compute one useful point (4.6).

p	16	32	64	96	128	128
		$k = 1$		$k = 2$		$K = 1$
Eq. 1	8	16	32	24	32	56
Eq. 2	4	8	16	12	16	31
Eq. 3	2	4	8	6	8	8
Eq. 4	2	4	8	6	8	9
Total number of p	16	32	64	96	128	104
Model time	3.33	1.76	1.05	0.87	0.7	0.9
T_p	3.38	1.76	1.06	0.86	0.7	0.95
Speed-up	14.33	27.55	45.96	56.72	69.08	51.23

approach would have a limited parallelisation possibility especially for problems of the size $J = 2000$. For such small subproblems it would be possible to utilise only up to 32 processes (Figure 3.1), the speed-up would be quite limited as well.

Note, that all previous results represent the analysis based on a single Nelder-Mead iteration. Next, we solve the actual real-world optimization problem (4.6). The maximum number of processes $P = 128$ the load balancing algorithm has selected $k = 1$. The number of Nelder-Mead method iterations was fixed to 1000. The parallel and sequentional versions gave the same results the minimum value of the error $E_\infty^C = 0.0806$. The sequentional version of computations took 180286 seconds, the parallel version computations took 2232 seconds. Thus, a speed-up factor of 81.8 was achieved. The selection of $k = 1$ indicates that the number of processes can be greatly in-

Table 4.5: The results for Benchmark 1 with $E_{min} > 0$. T_p is the CPU time in seconds required to compute one useful point (4.6).

p	128		
E_{min}	0.75 $k = 3$	0.8 $k = 3$	0.6 $K = 1$
Eq. 1	26	19	42
Eq. 2	7	5	8
Eq. 3	4	3	4
Eq. 4	2	1	2
Total number of p	117	84	56
Model T_{Mp}	2.45	3.17	3.84
T_p	2.49	3.08	3.76
Speed-up	56.03	45.37	37.11

creased – the algorithm has selected $k = 1$ automatically for a given number of processes.

4.5 The Comparison of Different Nelder-Mead Parallelisation Methods

Here we present the analysis of the convergence properties of different modifications of the Nelder-Mead method. As it was mentioned before, the convergence rate of the selected algorithm directly affects the parallelisation efficiency, which is represented by γ_k, where k is the parallelisation degree. In this section we measure γ_k by measuring the experimental parallel efficiency of algorithms.

The detailed analysis of convergence behaviour for different objective functions is out of the scope of this research. However, the objective function from the previous sections is suitable for a narrow class of applications. Thus, to perform a comparison of different parallel versions of Nelder-Mead method we minimise the Rosenbrock objective function that is widely used by researchers in the field of optimization theory [33], [87].

We show that in the case of the Rosenbrock function the real experimental γ_k values are different than were assumed to be in the experiments of the previous sections. The reason is that the significant number of iterations require to compute only one point F_R.

We compare the results of our parallel modification of the Nelder-Mead method with the state-of-art technique proposed in [58]. As a benchmark

we use the Rosenbrock function

$$f(x_1, \ldots, x_d) = \sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2, \tag{4.11}$$

which makes the optimization problem challenging. It should be noted that the parallel algorithm [58] can achieve the parallelisation degree K that is equal to the optimization problem dimension d. Thus potentially this algorithm is well suited for parallel computers with a big number of processes.

Table 4.6: The γ_k values for direct Nelder-Mead parallelisation

k	$d = 3$	$d = 6$	$d = 7$
2	0.603	0.604	0.606
3	0.584	0.517	0.502

In the Table 4.6 we compare three cases $d = 3, 6, 7$: $d = 3$ – the minimum, that is needed for parallelilsation with both methods, $d = 7$ – the case that was studied in the previous section, $d = 6$ – to show the tendency for smaller d. We provide the results obtained when the Rosenbrock function of different dimensions $d = 3, 6, 7$ was minimized by using our parallel modification of the Nelder-Mead method. The values of the efficiency coefficients γ_k are presented. They show that this parallel algorithm is quite stable and it is well-suited to be used in the three-level template solver for small dimension objective functions.

Table 4.7: The γ_k values for the parallel Nelder-Mead algorithm from (Lee and Wiswall, 2007)

k	$d = 3$	$d = 6$	$d = 7$
2	0.668	0.685	0.714
3	–	0.436	0.454
4	–	0.023	0.104
5	–	0.001	0.002
6	–	–	0.001

Table 4.7 presents results obtained by using the state-of-the-art parallel Nelder-Mead algorithm from [58]. It follows, that in all investigated cases the parallelisation degree is very limited, since the convergence drops significantly when the parallelization degree is increased. This method is mainly targeted to solve problems when the dimension of the objective function is big (e.g. problems in financial mathematics, when $d \approx 100$).

4.6 Conclusions of the Fourth Chapter

We introduced a three-level parallelisation template which utilises a new computational cost model based on experimental data. This model is used to perform the load balancing during task distribution step.

- Comparing the three-level template to the classical two-level scheme, the proposed algorithm greatly expands the number of processes that can be used. in the case when a big number is computational resources is available.
- The model-based approach performs balancing well enough for practical purposes. The model prediction times are close to times of the real computational experiments.
- The possibilities of the three-level parallelization template are demonstrated for solving local optimization problems. The proposed load balancing algorithm chooses the optimal version of the parallel Nelder-Mead algorithm. It dynamically increases the parallelisation degree on the first level when the speed-up of the second and third levels begins to saturate.
- The proposed approach extends the parallelisation degree allowing to achieve an additional speed-up.
- The proposed load balancing algorithm limits the size of computational resources to preserve the efficiency requirement which can be controlled by selecting the parameter E_{min}. The results of computational experiments show, that This heuristic for considered cases is sufficient.

General Conclusions

1. The proposed parallel algorithms for the problems of fractional powers of elliptic operators are efficient enough (in some cases, efficiency is close to 1). The highest accuracy solutions are obtained using the quadrature method (M3) with exponentially convergent quadrature formula.

2. Two techniques to obtain coefficients of rational function, approximation of the Fourier symbol and minimization of the reflection coefficient, are not universal. The appropriate set of coefficients for four qualitatively different testing Schrödinger problems in the selected seven parameters space was found using a coupled adaptive technique, when evaluating errors for different tasks.

3. The proposed three-level parallel scheme was compared to the classical two-level parallel algorithm. The proposed parallel scheme improves the degree of parallelism and the additional degree of parallelism, this lets to use a bigger number of available computational resources (1.5 times more in the considered cases) and lets to achieve an additional speed-up.

List of Publications by the Author on the Topic of the Dissertation

The articles published in peer-reviewed periodical journals

1. Čiegis, R.; Starikovičius, V.; Margenov, S.; Kriauzienė, R. Scalability analysis of different parallel solvers for 3D fractional power diffusion problems. *Concurrency and computation: practice and experience*. Chichester: John Wiley & Sons, Ltd. ISSN 1532-0626. eISSN 1532-0634. 2019, Vol. 3, iss. , p. –. DOI: 10.1002/cpe.5163.

2. Čiegis, R; Starikovičius, V; Margenov, S; Kriauzienė, R. Parallel solvers for fractional power diffusion problems *Concurrency and computation: practice and experience*. Chichester: John Wiley & Sons, Ltd. ISSN 1532-0626. eISSN 1532-0634. 2017, Vol. 29, iss. 24, p. 1-12. DOI: 10.1002/cpe.4216.

3. Bugajev, A.; Čiegis, R.; Kriauzienė, R.; Leonavičienė, T.; Žilinskas, J. On the accuracy of some absorbing boundary conditions for the Schrödinger equation *Mathematical modelling and analysis: the Baltic journal on mathematical applications, numerical analysis and differential equations*. Vilnius: Taylor& Francis, VGTU. ISSN 1392-6292. eISSN 1648-3510. 2017, Vol. 22, iss. 3, p. 408–423. DOI: 10.3846/13926292.2017.1306725.

Proceedings of other conferences:

1. Čiegis, R., Starikovičius, V., Margenov, S., Kriauzienė, R. 2018. A comparison of accuracy and efficiency of parallel solvers for fractional power diffusion problems. Parallel Processing and Applied Mathematics: 12th international conference, PPAM 2017, Lublin, Poland,

September 10–13, 2017. Basel, Springer International Publishing, pp. 79-89, ISBN 9783319780238. DOI: 10.1007/978-3-319-78024-5_8.

Book chapter

1. Margenov, S; Rauber, Th.; Atanassov, E.; Almeida, F.; Blanco, V.; Čiegis, R.; Cabrera, A.; Frasheri, N.; Harizanov, S.; Kriauzienė, R.; Rünger, R.; Segundo, P.S.; Starikovičius, V.; Szabo, S.; Zavalnij, B. Ultrascale Computing Systems. Applications for ultrascale systems. Institution of Engineering and Technology, 2019, p. 189-244. DOI: 10.1049/PBPC024E_ ch6. https://digital-library.theiet.org/content/books/10.1049/pbpc024e_ch6

Abstracts in the conference proceedings:

1. Kriauzienė, R; Bugajev, A; Čiegis, R. The analysis and application of a three-level parallelisation scheme Mathematical Modelling and Analysis [MMA2019]: 24rd international conference, May 28-May 31, 2019, Talinn, : abstracts.

2. Kriauzienė, R; Bugajev, A; Čiegis, R. A three-level parallelisation algorithm for optimization problems Mathematical Modelling and Analysis [MMA2018]: 23rd international conference, May 29-June 1, 2018, Sigulda, Latvia: abstracts. Riga: University of Latvia, 2018. ISBN 9789934195174. p. 41.

3. Kriauzienė, R; Bugajev, A.; Čiegis, R. Numerical analysis and optimization of parallel algorithms for problems with big computational costs 3rd NESUS Winter School and PhD Symposium 2018, 22nd-25th January 2018, Zagreb, Croatia. Zagreb: Ruđer Bošković Institute. 2018, vol. 1, no. 1, p. 10.

4. Kriauzienė, R.; Bugajev, A.; Čiegis, R. Three level parallelisation scheme for optimization problems involving simultaneous calculations of multiple differential equations DAMSS 2018 : 10th international workshop on "Data analysis methods for software systems", Druskininkai, Lithuania, November 29 – December 1, 2018 : [abstract book]. Vilnius : Vilnius University Press, 2018. ISBN 9786090700433. p. 48. DOI: 10.15388/DAMSS.2018.1.

5. Čiegis, R.; Starikovičius, V.; Margenov, S.; Kriauzienė, R. A comparison of accuracy and efficiency of parallel solvers for fractional power diffusion problems PPAM 2017: 12th International Conference on

Parallel Processing and Applied Mathematics, September 10-13, 2017, Lublin, Poland: book of abstracts. Czestochowa: Czestochowa University of Technology. 2017, p. 99.

6. Čiegis, R.; Starikovičius, V.; Margenov, S.; Kriauzienė, R. Lygiagrečiųjų skaitinių algoritmų, skirtų uždaviniams su elipsiniais operatoriais, pakeltais trupmeniniu laipsniu, analizė Fizinių ir technologijos mokslų tarpdalykiniai tyrimai: 7-oji jaunųjų mokslininkų konferencija, 2017 m. vasario 9 d. Vilnius: Lietuvos mokslų akademijos leidykla. 2017, p. 56-57.

7. Starikovičius, V.; Čiegis, R.; Margenov, S.; Kriauzienė, R.. Parallel algorithms for the numerical solution of problems with fractional powers of elliptic operators Mathematical Modelling and Analysis [MMA2017] : 22nd international conference, May 30-June 2, 2017, Druskininkai, Lithuania : abstracts. Vilnius : Technika, 2017. ISBN 9786094760228. eISBN 9786094760211. p. 64-65.

8. Kriauzienė, R.; Bugajev, A.; Čiegis, R. The new parallel multilevel tool for implementation of applied optimization algorithms 9th International workshop on Data Analysis Methods for Software Systems (DAMSS), Druskininkai, Lithuania, November 30 - December 2, 2017. Vilnius : Vilniaus University, 2017. ISBN 9789986680642. p. 28-29. DOI: 10.15388/DAMSS.2017.

9. Bugajev, A.; Kriauzienė, R.; Čiegis, R. On the accuracy of some absorbing boundary conditions for the Schrodinger equation Mathematical Modelling and Analysis (MMA2016): 21st international conference, June 1-4, 2016 in Tartu, Estonia: abstracts Institute of Mathematics and Statistics of the University of Tartu, European Consortium for Mathematics in Industry, Vilnius Gediminas Technical University, Estonian Mathematical Society. Tartu : University of Tartu, 2016. ISBN 9789949918096. p. 11.

10. Kriauzienė, R.; Bugajev, A.; Čiegis, R.; Leonavičienė, T.; Žilinskas, J.; Jankevičiūtė, G. Optimization of efficient absorbing boundary conditions for schrödinger equation Data analysis methods for software systems: 8th international workshop on data analysis methods for software systems, Druskininkai, December 1-3, 2016 : [abstract]. Vilnius: Vilniaus universiteto leidykla, 2016. ISBN 9789986680611. p. 30-31. DOI: 10.15388/DAMSS.2016.

Bibliography

[1] G. Acosta and J. Borthagaray. A fractional laplace equation: Regularity of solutions and finite element approximations. *SIAM Journal on Numerical Analysis*, 55(2):472–495, 2017.

[2] M. Ainsworth and C. Glusa. Aspects of an adaptive finite element method for the fractional Laplacian: A priori and a posteriori error estimates, efficient implementation and multigrid solver. *Computer Methods in Applied Mechanics and Engineering*, 327:4–35, 2017.

[3] G. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings*, 30:483–485, 1967.

[4] S. Amiranashvili, R. Čiegis, and M. Radziunas. Numerical methods for a class of generalized nonlinear schrödinger equations. *Kinetic and Related Models*, 8(2):215–234, 2015.

[5] X. Antoine, A. Arnold, C. Besse, M. Ehrhardt, and A. Schädle. A review of transparent and artificial boundary conditions technique for linear and nonlinear Schrödinger equations. *Communications in Computational Physics*, 4(4):729–796, 2008.

[6] X. Antoine and C. Besse. Unconditionally stable discretization schemes of non-reflecting boundary conditions for the one-dimensional Schrödinger equation. *Journal of Computational Physics*, 188:157–175, 2003. doi:10.1016/S0021-9991(03)00159-1.

[7] A. Arnold. Numerically absorbing boundary conditions for quantum evolution equations. *VLSI Design*, 6(1–4):313–319, 1998.

[8] G. A. Baker and J. L. Gammel. *The Padé approximant in theoretical physics*, volume 71. Academic Press, New York, NY, USA, 1970.

[9] L. Banjai, J. M. Melenk, R. H. Nochetto, E. Otárola, A. J. Salgado, and C. Schwab. Tensor FEM for spectral fractional diffusion. *Foundations of Computational Mathematics*, 19(4):901–962, 2018.

[10] V. Baskakov and A. Popov. Implementation of transparent boundaries for numerical solution of the Schrödinger equation. *Wave Motion*, 14(2):123–128, 1991.

[11] J. Bilmes, K. Asanovic, C.-W. Chin, and J. Demmel. Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology. In *Proceedings of the 11th international conference on Supercomputing*, pages 340–347. ACM, 1997.

[12] A. Bonito, J. P. Borthagaray, R. H. Nochetto, E. Otárola, and A. J. Salgado. Numerical methods for fractional diffusion. *Computing and Visualization in Science*, 19(5):19–46, 2018.

[13] A. Bonito and J. Pasciak. Numerical approximation of fractional powers of elliptic operator. *Mathematics of Computation*, 84, 2015.

[14] C. Bruneau, L. D. Menza, and T. Lehner. Numerical resolution of some nonlinear Schrödinger-like equations in plasmas. *Numerical Methods for Partial Differential Equations*, 15(6):672–696, 1999.

[15] A. Bugajev, R. Čiegis, R. Kriauzienė, T. Leonavičienė, and J. Žilinskas. On the accuracy of some absorbing boundary conditions for the schrodinger equation. *Mathematical Modelling and Analysis*, 22(3):408–423, 2017.

[16] J. Carretero, S. Distefano, D. Petcu, D. Pop, T. Rauber, G. Rünger, and D. E. Singh. Energy-efficient algorithms for ultrascale systems. *Supercomputing frontiers and innovations*, 2(2):77–104, 2015.

[17] A. Caserta, R. Garra, and E. Salusti. Application of the fractional conservation of mass to gas flow diffusivity equation in heterogeneous porous media. *Geophysics*, 2016. arXiv:1611.01695v1.

[18] R. Čiegis. Investigation of difference schemes for a class of models of excitability. *Computational Mathematics and Mathematical Physics*, 32(6):757–767, 1996.

[19] R. Čiegis. *Parallel algorithms and networking technologies.* Technika, Vilnius, 2005. (in Lithuanian).

[20] R. Ciegis and M. Baravykaite. Implementation of a black-box global optimization algorithm with a parallel branch and bound template. *Applied Parallel Computing: State of the Art in Scientific Computing*, 4699:1115–1125, 2007.

[21] R. Čiegis and M. Radziunas. Effective numerical integration of traveling wave model for edge-emitting broad-area semiconductor lasers and amplifiers. *Mathematical Modelling and Analysis*, 15(4):409–430, 2010.

[22] R. Čiegis and G. Šilko. A scheme for partitioning regular graphs. In R. Wyrzykowski, E. Deelman, J. Dongarra, K. Karczewski, J. Kitowski, and K. Wiatr, editors, *Proc. 4th International Conference on Parallel Processing and Applied Mathematics (PPAM2001, Naleczsow, Poland, September9-12, 2001)*, volume 2328 of *Lecture Notes in Computer Science*, pages 404–409, Berlin, Germany, 2002. Springer.

[23] R. Čiegis, V. Starikovičius, S. Margenov, and R. Kriauzienė. Parallel solvers for fractional power diffusion problems. *Concurrency and Computation: Practice and Experience*, 25(24), 2017.

[24] R. Čiegis, V. Starikovičius, S. Margenov, and R. Kriauzienė. A comparison of accuracy and efficiency of parallel solvers for fractional power diffusion problems. In *Parallel Processing and Applied Mathematics, (PPAM2017, Lublin, Poland, September 9–13, 2017) Proceedings, part I*, volume 10777 of *Lecture Notes in Computer Science*, pages 79–89, Berlin, Heidelberg, 2018. Springer.

[25] R. Čiegis, V. Starikovičius, S. Margenov, and R. Kriauzienė. Scalability analysis of different parallel solvers for 3D fractional power diffusion problems. *Concurrency and Computation: Practice and Experience*, 2019.

[26] R. Čiegis, V. Starikovičius, N. Tumanova, and M. Ragulskis. Application of distributed parallel computing for dynamic visual cryptography. *The Journal of Supercomputing*, 72(11):4204–4220, 2016.

[27] R. Čiegis and N. Tumanova. On construction and analysis of finite difference schemes for pseudoparabolic problems with nonlocal boundary conditions. *Mathematical Modelling and Analysis*, 19(2):281–297, 2014.

[28] N. Cusimano, A. Bueno-Orovio, I. Turner, and K. Burrage. On the order of the fractional laplacian in determining the spatio-temporal evolution of a space-ractional model of cardiac electrophysiology. *PLOS ONE*, 10(12):1–16, 2015.

[29] N. Cusimano, F. del Teso, L. Gerardo-Giorda, and G. Pagnini. Discretizations of the spectral fractional laplacian on general domains with Dirichlet, Neumann, and Robin boundary conditions. *SIAM Journal on Numerical Analysis*, 56(3):1243–1272, 2018.

[30] A. Datta, A. Kaur, T. Lauer, and S. Chabbouh. Exploiting multi–core and many–core parallelism for subspace clustering. *International Journal of Applied Mathematics and Computer Science*, 29(1):81–91, 2019.

[31] M. D'Elia and M. Gunzburger. The fractional Laplacian operator on bounded domains as a special case of the nonlocal diffusion operator. *Computers and Mathematics with Applications*, 66(7):1245–1260, 2013.

[32] B. Duan, R. Lazarov, and J. Pasciak. Numerical approximation of fractional powers of elliptic operators, 2019.

[33] I. Fajfar, Á. Bűrmen, and J. Puhan. The nelder–mead simplex algorithm with perturbed centroid for high-dimensional function optimization. *Optimization Letters*, pages 1–15, 2018.

[34] R. Falgout, J. Jones, and U. Yang. The design and implementation of Hypre, a library of parallel high performance preconditioners. In A. M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers, part III*, volume 51 of *Lecture Notes in Computational Science and Engineering*, pages 264–294, Springer, Berlin, Heidelberg, 2006.

[35] R. Falgout and U. Yang. hypre: a library of high performance preconditioners. In P. M. A. Sloot, A. Hoekstra, C. Tan, and J. Dongarra, editors, *Computational Science 2002. International Conference (ICCS, Amsterdam, The Netherlands, April 21–24, 2002) Proceedings, part III*, volume 2331 of *Lecture Notes in Computer Science*, pages 632–641, Berlin, Heidelberg, 2002. Springer.

[36] M. Fan, S. Li, and L. Zhang. Weak solution of the equation for a fractional porous medium with a forcing term. *Computers and Mathematics with Applications*, 67(1):145–150, 2014.

[37] D. Finkel. DIRECT optimization algorithm user guide. *Center for Research in Scientific Computation, North Carolina State University*, 2:1–14, 2003.

[38] M. Frigo and S. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[39] H. Fu, M. Ng, M. Nikolova, and J. Barlow. Efficient minimization methods of mixed l2-l1 and l1-l1 norms for image restoration. *SIAM Journal on Scientific Computing*, 27(6):1881–1902, 2006.

[40] M. Furuichi and D. Nishiura. Iterative load-balancing method with multigrid level relaxation for particle simulation with short-range interactions. *Computer Physics Communications*, 219:135–148, 2017.

[41] F. Glover. Tabu search–part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[42] F. Glover. Tabu search–part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[43] C. Gong, W. Bao, and G. Tang. A parallel algorithm for the Riesz fractional reaction-diffusion equation with explicit finite difference method. *Fractional Calculus and Applied Analysis*, 16(3):654–669, 2016.

[44] S. Harizanov, R. Lazarov, P. Marinov, S. Margenov, and J. Pasciak. Comparison analysis on two numerical methods for fractional diffusion problems based on rational approximations of $t^\gamma,\ 0 \leq t \leq 1$, 2018.

[45] S. Harizanov, R. Lazarov, P. Marinov, S. Margenov, and Y. Vutov. Optimal solvers for linear systems with fractional powers of sparse SPD matrices. *Numerical Linear Algebra with Applications*, 25(5), 2018.

[46] B. I. Henry, T. A. M. Langlands, and P. Straka. *An Introduction to Fractional Diffusion*, pages 37–89. World Scientific, 2012.

[47] I. Huismann, J. Stiller, and J. Frohlich. Two-level parallelization of a fluid mechanics algorithm exploiting hardware heterogeneity. *Computers & Fluids*, 117:114–124, 2015.

[48] E.-J. Im and K. Yelick. Optimizing sparse matrix computations for register reuse in SPARSITY. In *International Conference on Computational Science*, pages 127–136. Springer, 2001.

[49] S. Imam and V. Sarkar. Load balancing prioritized tasks via work-stealing. In *Euro-Par 2015: Parallel Processing*, volume 9233, pages 222–234, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[50] C. Kelley. *Iterative methods for optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[51] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[52] M. Köpf, R. Metzler, O. Haferkamp, and T. F. Nonnenmacher. *NMR Studies of Anomalous Diffusion in Biological Tissues: Experimental Observation of Lévy Stable Processes*, pages 354–364. Birkhäuser Basel, 1998.

[53] R. Kriauzienė, A. Bugajev, and R. Čiegis. A three-level parallelisation scheme and application to the Nelder-Mead algorithm. *http://arxiv.org/abs/1904.05208*, 2019.

[54] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms*. Benjamin-Cummings Pub Co, 1994.

[55] A. Lastovetsky and R. R. Manumachu. New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1119–1133, 2017.

[56] A. Lastovetsky, L. Szustak, and R. Wyrzykowski. Model-based optimization of eulag kernel on intel xeon phi through load imbalancing. *Ieee Transactions on Parallel and Distributed Systems*, 28(3):787–797, 2017.

[57] R. Lazarov and P. Vabishchevich. A numerical study of the homogeneous elliptic equation with fractional order boundary conditions. *Fractional Calculus and Applied Analysis*, 20(2):337–351, 2017.

[58] D. Lee and M. Wiswall. A parallel implementation of the simplex function minimization routine. *Computational Economics*, 30(2):171–187, 2007.

[59] E. Lindman. Free-space boundary conditions for the time dependent wave equation. *Journal of Computational Physics*, 18(1):16–78, 1985.

[60] A. Llanes, J. M. Cecilia, A. Sanchez, J. M. Garcia, M. Amos, and M. Ujaldon. Dynamic load balancing on heterogeneous clusters for parallel ant colony optimization. *Cluster Computing-the Journal of Networks Software Tools and Applications*, 19(1):1–11, 2016.

[61] R. Magin, C. Ingo, L. Colon-Perez, W. Triplett, and T. Mareci. Characterization of anomalous diffusion in porous biological tissues using fractional order derivatives and entropy. *Microporous and Mesoporous Materials*, 178:39–43, 2013.

[62] R. L. Magin. Fractional calculus models of complex dynamics in biological tissues. *Computers and Mathematics with Applications*, 59(5):1586–1593, 2010.

[63] R. Martí, J. Lozano, A. Mendiburu, and L. Hernando. *Multi-start methods*. Springer International Publishing, Vilnius, Lithuania, 2015.

[64] K. McKinnon. Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158, 1998.

[65] D. Meidner, J. Pfefferer, K. Schürholz, and B. Vexler. h_p-finite elements for fractional diffusion. *SIAM Journal on Numerical Analysis*, 56(4):2345–2374, 2018.

[66] L. D. Menza. Transparent and absorbing boundary conditions for the Schrödinger equation in a bounded domain. *Numerical Functional Analysis and Optimization*, 18(7–8):759–775, 1997.

[67] C. Moyer. Numerov extension of transparent boundary conditions for the Schrödinger equation in one dimension. *American Journal of Physics*, 72(3):351–358, 2004.

[68] MPI: A message passing interface standard.

[69] J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[70] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[71] B. S. Nemalikanti, P. Sindhura, P. K. Tumalla, and S. Vemuru. Achieving green computing through algorithmic efficiency. *i-Manager's Journal on Information Technology*, 1(1):39, 2011.

[72] R. Nochetto, E. Otárola, and A. Salgado. A PDE approach to fractional diffusion in general domains: a priori error analysis. *Foundations of Computational Mathematics*, 15(3):733–791, 2015.

[73] R. Nochetto, E. Otárola, and A. Salgado. A PDE approach to numerical fractional diffusion. In *Proceedings of the 8th ICIAM, Beijing, China*, pages 211–236, 2015.

[74] H. Padé. Sur la représentation approchée d'une fonction par des fractions rationnelles. *Annales scientifiques de l'École Normale*, 9(3):1–93, 1892.

[75] B. Perez, E. Stafford, J. Bosque, and R. Beivide. Energy efficiency of load balancing for data-parallel applications in heterogeneous systems. *Journal of Supercomputing*, 73(1):330–342, 2017.

[76] C. Pozrikidis. *The Fractional Laplacian.* CRC Press, 2016.

[77] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations.* Oxford Science Publications, 1999.

[78] M. Radziunas, R. Čiegis, and A. Mirinavičius. On compact high order finite difference schemes for linear Schrödinger problem on non-uniform meshes. *International Journal of Numerical Analysis and Modelling*, 11(2):303–314, 2014.

[79] J. A. Rico-Gallego and J. C. Diaz-Martin. τ-Lop: Modeling performance of shared memory MPI. *Parallel Computing*, 46(C):14–31, 2015.

[80] J. A. Rico-Gallego, A. L. Lastovetsky, and J. C. Diaz-Martin. Model-based estimation of the communication cost of hybrid data-parallel applications on heterogeneous clusters. *IEEE Transactions on Parallel and Distributed Systems*, 28(11):3215–3228, 2017.

[81] R. D. Righi, R. D. Gomes, V. F. Rodrigues, C. A. da Costa, A. M. Alberti, L. L. Pilla, and P. O. A. Navaux. Migpf: Towards on self-organizing process rescheduling of bulk-synchronous parallel applications. *Future Generation Computer Systems-the International Journal of Escience*, 78:272–286, 2018.

[82] B. Saha. Green computing: Current research trends. *International Journal of Computer Sciences and Engineering*, 6(3):467–469, 2018.

[83] E. Scalas, R. Gorenflo, and F. Mainardi. Fractional calculus and continuous-time finance. *Physica A: Statistical Mechanics and its Applications*, 284(1-4):376–384, 2000.

[84] A. Sharma and M. Kaur. An efficient task scheduling of multiprocessor using genetic algorithm based on task height. *Journal of Information Technology & Software Engineering*, 5(2):1000151, 2015.

[85] R. Singh. Task scheduling in parallel systems using genetic algorithm. *International Journal of Computer Applications*, 108(16):34–40, 2014.

[86] W. Spendley, G. Hext, and F. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.

[87] L. Stripinis, R. Paulavičius, and J. Žilinskas. Improved scheme for selection of potentially optimal hyper-rectangles in DIRECT. *Optimization Letters*, 12(7):1699–1712, Oct 2018.

[88] J. Szeftel. Design of absorbing boundary conditions for Schrödinger equations in R^d. *SIAM Journal on Numerical Analysis*, 42(4):1527–1551, 2004.

[89] A. Törn and A. Žilinskas. *Global Optimization*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[90] P. Vabishchevich. Numerically solving an equation for fractional powers of elliptic operators. *Journal of Computational Physics*, 282:289–302, 2015.

[91] P. Vabishchevich. Numerical solving unsteady space-fractional problems with the square root of an elliptic operator. *Mathematical Modelling and Analysis*, 21(2):220–238, 2016.

[92] R. Varga and A. Carpenter. Some numerical results on best uniform rational approximation of x^α on $[0, 1]$. *Numerical Algorithms*, 2(2):171–185, 1992.

[93] R. Vuduc, J. W. Demmel, and K. A. Yelick. OSKI: A library of automatically tuned sparse matrix kernels. In *Journal of Physics: Conference Series*, volume 16, pages 521–530. IOP Publishing, 2005.

[94] H. Wang. A parallel method for tridiagonal equations. *ACM Transactions on Mathematical Software (TOMS)*, 7(2):170–183, 1981.

[95] T. Weise. *Global optimization algorithms. Theory and application.* Self-Published Thomas Weise, 2009.

[96] R. C. Whaley and J. J. Dongarra. Automatically tuned linear algebra software. In *Supercomputing, 1998. SC98. IEEE/ACM Conference on*, pages 38–38. IEEE, 1998.

[97] H. S. Yamada and K. Ikeda. A numerical test of padé approximation for some functions with singularity. *International Journal of Computational Mathematics*, 2014, 2014.

[98] Q. Yang, D. Chen, T. Zhao, and Y. Chen. Fractional calculus in image processing: A review. *Fractional Calculus and Applied Analysis*, 19(5):1222–1249, 2016.

[99] Q. Yang, F. Liu, and I. Turner. Numerical methods for fractional partial differential equations with Riesz space fractional derivatives. *Applied Mathematical Modelling*, 34(1):200–218, 2010.

[100] S. B. Yuste, L. Acedo, and K. Lindenberg. Reaction front in an $A+\vec{B}$ C reaction-subdiffusion process. *Physical Review E*, 69(3):036126, 2004.

[101] Y. Zheng, C. Li, and Z. Zhao. A note on the finite element method for the space-fractional advection diffusion equation. *Computers and Mathematics with Applications*, 59:1718–1726, 2010.

[102] J. Žilinskas. *Black box global optimization: covering methods and their parallelization.* KTU, Kaunas, Lithuania, 2002. Doctoral dissertation.

[103] A. Zlotnik and I. Zlotnik. Remarks on discrete and semi-discrete transparent boundary conditions for solving the time-dependent Schrödinger equation on the half-axis. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 31(1):51–64, 2016.

Rima Kriauzienė

PARALLEL ALGORITHMS FOR NON-CLASSICAL PROBLEMS WITH BIG COMPUTATIONAL COSTS

Doctoral Dissertation
Natural Sciences
Informatics (N 009)
Editor Zuzana Šiušaitė

UŽRAŠAMS

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt,
www.leidykla.vu.lt
Tiražas 20 egz.