Vilnius University
Institute of Mathematics and Informatics
LITHUANIA

INFORMATICS (09 P)

NUMERICAL ANALYSIS AND OPTIMIZATION OF ALGORITHMS FOR PROBLEMS WITH BIG COMPUTATIONAL COSTS.

Rima Kriauzienė

October 2018

Technical Report MII-DS-09P-18-7

VU Institute of Mathematics and Informatics, Akademijos str. 4, Vilnius LT-08663, Lithuania
www.mii.lt

Abstract

This report follows two problems. First, we propose a general methodology for solving optimisation problems when there is a big number of processes available. Second problem is to construct and investigate parallel solvers for three dimensional problems described by elliptic operators of fractional powers. This report contains a formulation of problems, methodology of solving these problems, experimental results, conclusions.

Keywords: multi-level parallelisation, load balancing, heterogeneous computing, parallel optimization, simplex downhill method, Wangs algorithm, finite difference method, fractional diffusion, finite volume method, parallel numeri- cal algorithms, MPI, scalability, multigrid.

Contents

1 Introduction

Objects of research are parallel algorithms and large-scale optimization problems. The main goal is to construct and investigate parallel algorithms for optimization problems in the case with the main constrains are defined by the differential equations and for problems with the fractional powers of elliptic operator. These problems are computationally costly. The numerical costs of data processing are increasing. The technology that allows solving such problems today is parallel computing. The main interest is the creation of technology that can address a variety of tasks that have the following characteristics: large amounts of data and large size of computations, because the tasks one needed to be solved very quickly, especially when solving optimisation problems.

Scalability and efficiency analysis of parallel algorithms are studied.

1.1 The object of research

Parallel algorithms for problems with big computational costs and large-scale optimization problems.

1.2 The goal and objectives of the research

1. The investigation and analysis of non-classical parallel algorithms.
2. The investigation of optimization problems in the case with the main constrains are defined by the differential equations.
3. Construction of the methodology of research:
3.1. The choice and analysis of mathematical models.
3.2. The choice and implementation of computational and parallel algorithms.
4. Theoretical research:
4.1. The approximation of mathematical models by discrete algorithms.
4.2. The formulation of optimization problems and theoretical analysis of their solving algorithm.
4.3. Construction of the parallel algorithms and scalability analysis.
5. Empirical investigation:
5.1. The solution and analysis of non-local problems.
5.2. The solution and analysis of problems with the fractional powers of elliptic operator.
5.3. The solution of problems of finding coefficients for special boundary conditions.
6. The analysis and generalization of obtained data, formulation of the conclusions.

1.3 Presentation and approbation of results

Conferences

1. DAMSS: 9th international workshop on data analysis methods for software systems, November 30–December 2, 2017, Druskininkai.

2. MMA2018: 23nd international conference, May 29-June 1, 2018, Sigulda, Latvia.

3. 3rd NESUS Winter School and 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

Publications

1. Čiegis, R.; Starikovicius, V.; Margenov, S.; Kriauziene, R. A scalability analysis of different parallel solvers for 3D fractional power diffusion problems. Concurrency and computation: practice and experience. (revised)

2. Kriauzienė, R., Bugajev, A., Čiegis, R. A three-level parallelisation scheme and application to the optimisation problems. IEEE Transactions on Parallel and Distributed Systems.

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. eISBN 9783319780245. (Lecture Notes in Computer Science, ISSN 0302-9743, eISSN 1611-3349 ; Vol. 10777). DOI: 10.1007/978-3-319-78024-5_8.

2 Formulation and parallelisation strategies of the first problem

We propose a general methodology for solving optimisation problems when there is a big number of processes available. In this research we investigate a three-level parallelisation algorithm for optimisation problems, different parallelisation levels create different 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 parallelilasation speed-up on the first level is not linear, it can lower the efficiency of the whole parallelisation. In this paper 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 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 sizes must be used for each equation in order to achieve the same level of errors. The third level defines parallel algorithms to solve tasks from the second level. As an example

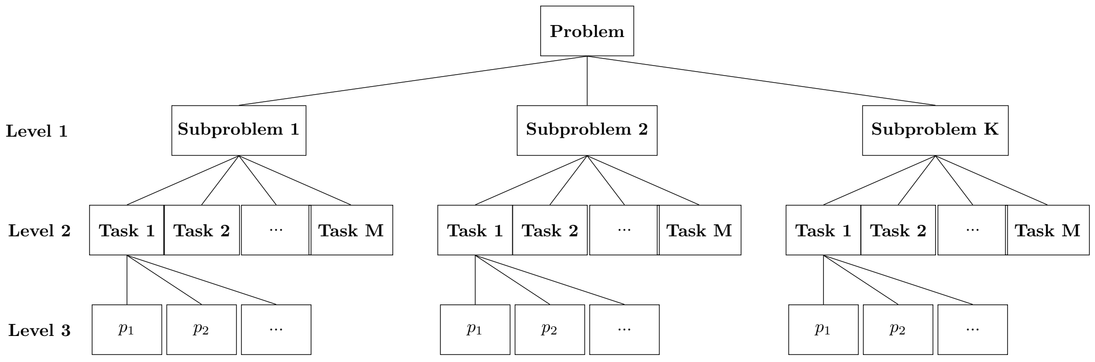

Figure 1: Three level parallelisation scheme.

we take Wang's algorithm to parallelise the solution of systems of linear equations with tridiagonal matrices [Wan81]. . 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 optimisation algorithm with a standard two level parallelisation. This is a new three-level parallelisation scheme 1.

As an example, it can be M different PDEs $L_j u = 0, 1 \leq j \leq M$, which are approximated numerically with $L_j^h U = 0, 1 \leq j \leq M$ with solutions u^h. Solutions depend on parameters $q_1, q_2, \ldots, q_m$. The optimization problem

$$\min_{q_1, q_2, \ldots, q_m} F(u_1^h(q_1, q_2, \ldots, q_m), \ldots, u_M^h(q_1, q_2, \ldots, q_m)).$$

2.1 Balancing of workload distribution

Let a parallel system has P processors. On the first level we can generate up to K blocks $V_1, \ldots, V_K$ of computational tasks. 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. For each task v_j the prediction of computation time $t_j(p)$, $p \leq p_j$, $j = 1, \ldots, M$ is given. Here we assume that the monotonicity condition

$$t_j(p_2) < t_j(p_1), \quad \text{for} \ \tilde{p}_1 < p_2 \leq \bar{P}_j \tag{1}$$

is satisfied, and we assume that $\bar{P}_j$ is a global minimum

$$t_j(p) \geq t_j(\bar{P}_j), \quad \text{for} \ p > \bar{P}_j. \tag{2}$$

We define the maximum number of processes $\tilde{P}_j$ which satisfies the efficiency condition

$$\frac{t_j(1)}{p\, t_j(p)} \geq E_{min}, \quad \text{for} \ p \leq \tilde{P}_j, \tag{3}$$

where $E_{min} \in [0, 1]$ is a given efficiency lower bound. Estimates (2) and (3) give the maximum number of processes P_j that is allowed to be used to solve the j-th task

$$P_j = \min(\bar{P}_j, \tilde{P}_j). \quad (4)$$

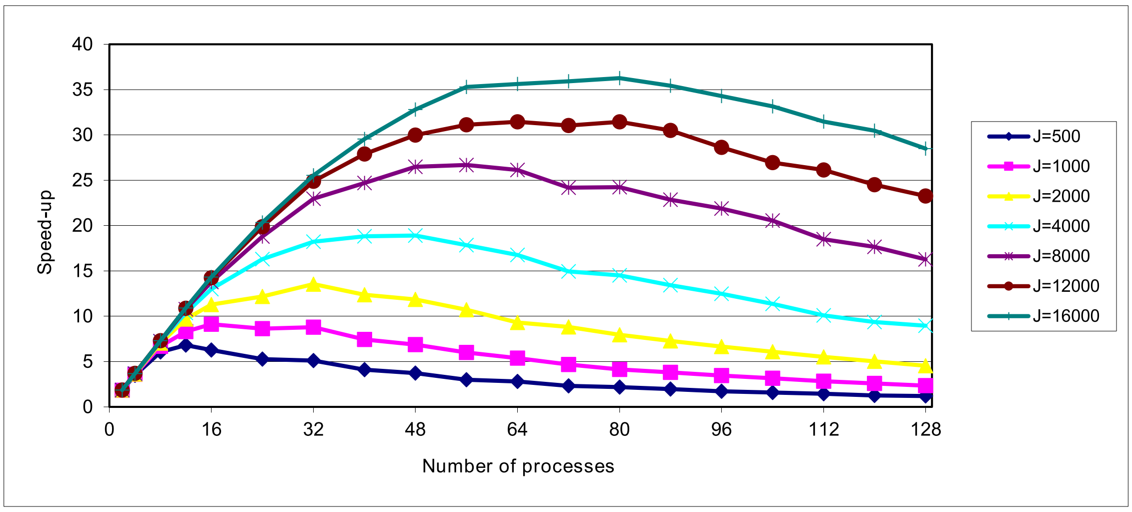

Figure 2: The speed-ups of Wang's parallel algorithm for different number of processes p and sizes J of systems.

In Fig. 2, we present speed-up's of Wang's algorithm for different sizes of linear systems J. As we see from presented curves the speed-up is not linear, if it was linear, the parallelisation on the third level alone would be sufficient. So, in our case the linear model is especially unsuitable. That's why we introduce the time function which interpolates the empirical data.

We propose hybrid multi-level approach, which combines all three strategies together, which improves the scalability.

We consider the following main minimisation problem: find the optimal value k_0 of task blocks

$$T_0(k_0) = \min_{1 \le k \le K} T_B(P/k)/\Gamma(k), \quad (5)$$

$\Gamma(k) = k\gamma_k$, $0 < \gamma_k \le 1$, where $T_B(p)$ defines the optimal time for solving one block of M tasks using p processes:

$$T_B(P) = \min_{(p_1,\ldots,p_M) \in S} \max_{1 \le m \le M} t_m(p_m), \quad (6)$$

where a set S of feasible processors distributions is defined as

$$S = \{(p_1, \ldots, p_M) : \ p_m \le P_m, \ m = 1, \ldots, M, \quad p_1 + \ldots + p_M \le P\}.$$

We propose the algorithm of workload balancing 3. This algorithm takes parameters K, M, problem sizes, number of processes and the array to store the processes distribu-

```
 1: Set $p[m] = 1$, for $m = 1, \dots, M$
 2: $P = P - M$
 3: Compute $t_m(p[m])$, for $m = 1, \dots, M$
 4: stop = 0
 5: **while** $P > 0$ & stop $==$ 0 **do**
 6:     find $j$ such that $t_j(p[j]) = \max_{1 \le m \le M} t_m(p[m])$
 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 3: The algorithm for distribution of P processes between M tasks

tion between problems. We assume that P is bigger than M and is dividable by K. First, we give one process per problem and then the rest of processes are iteratively distributed by selecting the longest of tasks. Here we see a special check if additional processes will not give better (bigger) speed-up, in this case we just leave same of processes unused. Also the number of processes is limited by efficiency requirement, which means it is not allowed increase the number of processes per task v_j if it makes the calculations efficiency smaller than the selected constant E_{min}.

2.2 Example

The linear one-dimensional Schrödinger equation with some initial-boundary conditions in a finite space domain

$$\begin{aligned} & i\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} = 0, \;\; x \in (a, b), t \in [0, T] \\ & u(x, 0) = u_0(x) \\ & L_l u(a) = 0, L_r u(b) = 0. \end{aligned} \tag{7}$$

The boundary conditions approximation

$$\partial_n u = -e^{-i\frac{\pi}{4}} \left(\left(\sum_{k=1}^{\frac{m+1}{2}} q_k \right) u - \sum_{k=1}^{\frac{m-1}{2}} q_{k+1} q_{k+(m+3)/2} \varphi_k \right), \tag{8}$$

where $\partial_n u$ is the normal derivative, the number of parameters $m \in \mathbb{N}$ is odd and φ_k is obtained from

$$\frac{d\varphi_k(x,t)}{dt} + q_{k+(m+3)/2}\varphi_k(x,t) = u\left(x,t\right), \; x = a, b, \; k = 1, \dots, [m/2].$$

Then we formulate an optimization problem

$$\min_{q_1,q_2,\dots,q_m} \max_{1\le j\le M} \|u_j - u_j^h(q_1,q_2,\dots,q_m)\|_\infty$$

Assuming that u_j is known, each value of functional that is being minimized requires to solve M different equations. All M equations can be solved independently.

2.3 Parallelisation scheme

First level of parallel algorithm.
As a local optimizer Nelder-Mead algorithm [NM65] is used. During each iteration we can have these different scenarios

- Reflection (one point: f_R)
- Expansion (two points: f_R, f_e)
- Contraction (two points: f_R, f_c)

We propose to compute all three points simultaneously, in our parallelization scheme that means parameter K is equal 3. We change the order of computations, which let us paralyse simplex method.

Second level of parallel algorithm.
We compute functional

$$\max_{1\le j\le M} \|u_j - u_j^h(q_1,q_2,\dots,q_m)\|_\infty = F(q_1,q_2,\dots,q_m), \tag{9}$$

where u^h approximation error should be small comparing to F.

U_j with different j can be computed independently, computation of maximum is small comparing to computations of finding solutions and computational errors. This leads to efficient parallel calculation of solutions.

Different PDEs can require different discretization sizes in order to achieve the same level of errors. This leads to unequal computational costs for different problems leading to loss of parallel efficiency.

Third level of parallel algorithm.
The system of linear equations

$$\begin{cases} b_0x_0 + c_0x_1 = d_0, \\ a_ix_{i-1} + b_ix_i + c_ix_{i+1} = d_i, \quad i = 1,\dots,N-2 \\ a_{N-1}x_{N-2} + b_{N-1}x_{N-1} = d_{N-1}, \end{cases} \tag{10}$$

where a_i, b_i, c_i, d_i are complex numbers.

Linear equations with tridiagonal matrix are solved using Wang's algorithm.

This level let us to perform the workload balancing at the second level of our algorithm. We assign different numbers of processes for different problems with different computational costs.

2.4 Experimental results

We have three different sized benchmarks 1.

Table 1: Benchmarks with different sizes

Benchmark 1		Benchmark 2		Benchmark 3	
Eq.	$J \times N$	Eq.	$J \times N$	Eq.	$J \times N$
1	8000×40000	1	8000×20000	1	8000×10000
2	4000×20000	2	4000×20000	2	2000×20000
3	2000×20000	3	4000×10000	3	2000×10000
4	2000×10000	4	2000×10000	4	1000×20000

The results of first benchmark are in Table 2. We can see how processes are distributed in these cases. As for parameter k, we choose it just by calculating with equal 1, equal 3 and selecting the best of them. The parameter k is selected automatically. The exception is the last column that was made for demonstration purposes. It shows what efficiency of the parallel algorithm is obtained only for two level template.

As we see, if P = 128; k = 1 then only 70 processes are used. However, the result is very similar to P = 64, additional 6 (almost 10%) processes decreased the computation time only by 0.3%, which means that these additional resources were used very inefficiently. Also, there are experimental times, actual times and speed-ups. There are some differences between real and our model values. The experimental time is smaller than model time.

The Gantt chart 4 shows theoretical model (which is based on experiments) time, that is needed to obtain the solutions of different equations. The workload distribution becomes closer to uniform as the number of processes is increased.

Table 2: The results of first benchmark.

p	16	32	64	96	128	128
	$k = 1$			$k = 2$	$k = 3$	$K = 1$
Eq. 1	10	22	**50**	34	29	**56**
Eq. 2	3	5	8	8	7	8
Eq. 3	2	3	4	4	4	4
Eq. 4	1	2	2	2	2	2
Model time	11.145	5.784	3.614	2.742	2.272	3.605
Exp. time	11.003	5.394	**3.608**	2.719	2.308	**3.600**
Speed-up	12.68	25.86	38.66	51.31	60.44	38.75

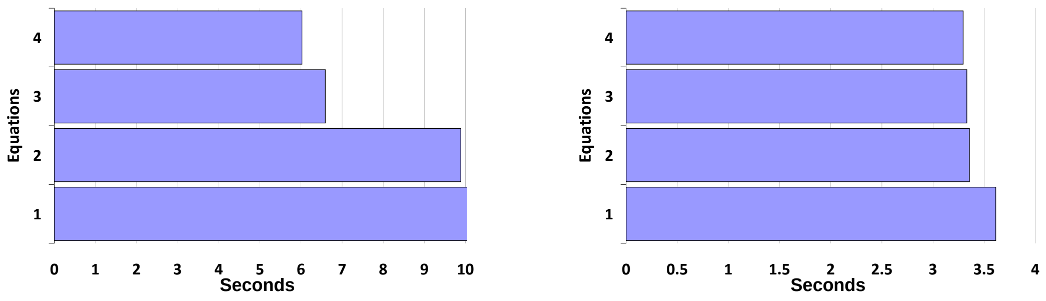

Figure 4: Benchmark 1 T values with $p = 16$(left) and $p = 64$(right)

There are results with efficiency requirement 3. The results in this table indicate that even with efficiency limitation $E_{min} = 0.75$ the proposed three-level approach lets to maintain a big number of parallel processes active. The last column in this table with $K = 1$ represents the results for the two-level approach (without the first level). This approach would have the limited parallelisation possibilities.

Table 3: The results of benchmark 1 with $E_{min} > 0$

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
Model time	2.45	3.17	3.84
Exp. time	2.49	3.08	3.76
Speed-up	56.03	45.37	37.11

2.5 Conclusions

Comparing to one level parallelization the proposed algorithm with three levels greatly expands the number of processes that can be used.

Workload balancing was analysed, model-based approach performs balancing well enough for practical purposes. The model prediction times are close to times of the real computational experiments.

We propose the heuristic with parameter E_{min} which guarantees that the efficiency of calculations on the third level is not lower than the value of E_{min}. This heuristic is not optimal, however, for considered cases we show that it is sufficient.

3 Formulation of the second problem

Let Ω be a bounded domain in $\mathbb{R}^3$, with boundary $\partial\Omega$. Given a function f, we seek u such that

$$L^\beta u = f, \quad X \in \Omega \tag{11}$$

with some boundary conditions on $\partial\Omega$, $0 < \beta < 1$ and the elliptic operator:

$$Lu = -\sum_{j=1}^{3} \frac{\partial}{\partial x_j}\left(k(X)\frac{\partial u}{\partial x_j}\right).$$

In our paper we use the following definition of fractional power operators. Let us denote by $\{\phi_k\}$, $k = 1, 2, \ldots, N$ the orthonormal basis

$$L\phi_k = \lambda_k \phi_k.$$

For convenience, here we restrict to the case of finite number of modes typical for discrete approximations. Then the fractional powers of the elliptic operator are defined by (cf. [NOS15b, NOS15a])

$$L^\beta u = \sum_{k=1}^{N} \lambda_k^\beta w_k \phi_k, \tag{12}$$

where $w_k = (u, \phi_k)$.

This definition can be used for direct solution of problem (11) by the Fourier method. However, in general, implementation of this approach is very expensive. It requires the computation of all eigenvectors and eigenvalues of large matrices. However, when the problem with fractional power of Laplace operator is solved in a rectangular domain, then the basis functions are known in advance and FFT techniques can be applied. In this paper, we formulate such 3D test problem and use the Fourier algorithm to obtain benchmark solutions. These benchmark solutions are used to study the convergence and accuracy of numerical solution algorithms, which are presented in the next section.

3.1 PDE Equivalent Models and Approximations of the Fractional Problem

3.1.1 Reduction to a pseudo-parabolic PDE problem

The solution of the non-local problem (11) is sought as a mapping [Vab16]:

$$V(X,t) = \big(t(L - \delta I) + \delta I\big)^{-\beta} f,$$

where $L \geq \delta_0 I$, $\delta = \gamma\delta_0$, $0 < \gamma < 1$. Thus, it follows that $V(X,1) = L^{-\beta} f$. The function V satisfies the evolutionary pseudo-parabolic problem

$$\begin{aligned}(tG + \delta I)\frac{\partial V}{\partial t} + \beta G V = 0, &\quad 0 < t \leq 1, \\ V(0) = \delta^{-\beta} f, &\quad t = 0,\end{aligned} \tag{13}$$

where $G = L - \delta I$. Thus, instead of the non-local problem (11), we solve a local pseudo-parabolic problem (13) (formally in $\mathbb{R}^4$). We use the following Crank-Nicolson scheme for the discretization in time [ČT14]:

$$(t^{n-1/2} G_h + \delta I_h) \frac{V_h^n - V_h^{n-1}}{\tau} + \beta G_h V_h^{n-1/2} = 0, \qquad 0 < n \le M, \tag{14}$$
$$V_h^0 = \delta^{-\beta} f_h,$$

where $G_h = L_h - \delta I_h$, $V_h^{n-1/2} = (V_h^n + V_h^{n-1})/2$ and $t^{n-1/2} = (t^{n-1} + t^n)/2$. In order to solve (14), M discrete 3D subproblems need to be solved.

3.2 Integral representation of the solution of problem (11)

The second algorithm we consider here is based on an integral representation of the non-local operator (12) using the local elliptic operators of the following form (cf. [BJ15]):

$$L^{-\beta} = \frac{2\sin(\pi\beta)}{\pi} \Big[\int_0^1 y^{2\beta-1} (I + y^2 L)^{-1} dy + \int_0^1 y^{1-2\beta} (y^2 I + L)^{-1} dy \Big]. \tag{15}$$

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

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

A similar partition is used to resolve the singularity of $y^{1-2\beta}$. The integrals (15) are approximated as

$$L_h^{-\beta} f_h = \frac{2\sin(\pi\beta)}{\pi} \Big[\sum_{j=1}^{M} \frac{y_{1,j}^{2\beta} - y_{1,j-1}^{2\beta}}{2\beta} \big(I_h + y_{1,j-1/2}^2 L_h\big)^{-1} f_h \tag{16}$$
$$+ \sum_{j=1}^{M} \frac{y_{2,j}^{2-2\beta} - y_{2,j-1}^{2-2\beta}}{2 - 2\beta} \big(y_{2,j-1/2}^2 I_h + L_h\big)^{-1} f_h \Big].$$

Note that local 3D elliptic subproblems $(I_h + y_j^2 L_h)^{-1} f$ and $(y_j^2 I_h + L_h)^{-1} f$ can be solved independently, $2M$ in total.

3.2.1 Approximation of the solution of problem (11) using rational approximations

In this case, the approximate solution is defined as in [HLM+16], namely,

$$\widetilde{U}_h = c_0 A_h^{-1} f_h + \sum_{j=1}^{m} c_i (A_h - d_j I)^{-1} f_h, \tag{17}$$

where $A_h = ch^2 L_h$, the coefficients c_j and d_j are defined by solving the global optimization problem to find the best uniform rational approximation $r_m^*(t)$,

$$r_m(t) = c_0 + \sum_{j=1}^{m} \frac{c_j t}{t - d_j},$$

$$\min_{r_m} \max_{t \in [0,1]} \left| t^{1-\beta} - r_m(t) \right| = \max_{t \in [0,1]} \left| t^{1-\beta} - r_m^*(t) \right|.$$

Let us define the error of the best uniform rational approximation as

$$\varepsilon_m(\beta) = \max_{t \in [0,1]} \left| t^{1-\beta} - r_m^*(t) \right|.$$

Then we have the following error bound

$$\|\widetilde{U}_h - U_h\|_{A_h} \le \varepsilon_m(\beta) \|f_h\|_{A_h^{-1}}.$$

Some examples of obtained approximations and results of numerical experiments are provided in [HLM+16].

3.3 Comparison of accuracy

In this work, we have compared selected numerical algorithms in terms of accuracy using the following 3D test problem:

$$L^{\beta} u = f(\vec{x}), \quad \vec{x} \in \Omega = [0,1] \times [0,1] \times [0,1]$$

with $\beta = 0.25$, Laplace operator $L = \Delta$ and

$$f(\vec{x}) = e^{-\left(\frac{x_1 - 0.5}{0.25}\right)^6} e^{-\left(\frac{x_2 - 0.5}{0.25}\right)^6} e^{-\left(\frac{x_3 - 0.5}{0.25}\right)^6}.$$

On sufficiently fine uniform grid ($N = 512$) we have computed reference solution with the Fourier method.

Solving the test problem with the integral method (16), we have used $M = N$ integration points. In order to have the same number of local elliptic problem to be solved, we have used $M = 2N$ time steps in computations with the pseudo-parabolic method (14). Method of rational approximation (17) was used with $m = 5$. Obtained errors in the maximum norm are presented in Table 4.

Some conclusions follow from Table 4. In accordance with the theory, the Fourier method converges exponentially and it is very fast when it can be applied. However, in this study, we are interested in methods, which can be used in non-rectangular domains for general elliptic operators. Results of the tests with pseudo-parabolic and integral algorithms show the second order convergence in accordance with the approximation properties of employed numerical schemes for smooth solutions (such as one selected for our tests). However, one needs to remember that in general for solutions of nonlocal

Table 4: Errors in maximum norm

N	Fourier	Pseudo-parabolic	Integral	Rational approximation
16	$1.66 \cdot 10^{-4}$	$4.05 \cdot 10^{-3}$	$1.70 \cdot 10^{-3}$	$2.67 \cdot 10^{-3}$
32	$2.94 \cdot 10^{-7}$	$1.21 \cdot 10^{-3}$	$5.00 \cdot 10^{-4}$	$6.68 \cdot 10^{-4}$
64	$8.88 \cdot 10^{-13}$	$3.34 \cdot 10^{-4}$	$1.23 \cdot 10^{-4}$	$7.08 \cdot 10^{-4}$
128	0	$8.62 \cdot 10^{-5}$	$3.05 \cdot 10^{-5}$	$4.32 \cdot 10^{-3}$
256	0	$2.18 \cdot 10^{-5}$	$7.66 \cdot 10^{-6}$	$7.51 \cdot 10^{-3}$

problems we have the lack of boundary regularity. Then the convergence rate of discrete solutions is reduced and depends on β.

Accuracy of the solutions obtained by the method of rational approximation is limited by the approximation error. If needed, more accurate approximations with $m = 6, 7$ should be used. However, determination of coefficients c_j and d_j for arbitrary β is a non-trivial task [HLM+16]. This makes the application of this method less practical.

3.4 Parallel algorithms and their efficiency

In this section we consider the parallelization of pseudo-parabolic and integral algorithms and compare their parallel performance. All tests were performed on the "Avitohol" cluster at Institute of Information and Communication Technologies (IICT) of the Bulgarian Academy of Sciences (http://www.iict.bas.bg/avitohol/). 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) and 64GB RAM. Computational nodes are interconnected via the fully non-blocking 56Gbps FDR InfiniBand network. We have used up to 32 nodes (512 cores) in our parallel tests.

3.4.1 Parallel pseudo-parabolic solver

The constructed finite volume scheme (14) implies that this numerical algorithm advances in pseudo-time solving one discrete 3D problem at each time step. M such problems need to be solved in total. However, they need to be solved sequentially, one after another. Hence the pseudo-parabolic algorithm does not have parallelism in the introduced dimension, i.e. in pseudo-time.

Thus our parallelization template is reduced only to the second level. We solve in parallel 3D elliptic subproblems. A standard domain decomposition method is applied. The discrete mesh Ω_h of size $N_{x_1} \times N_{x_2} \times N_{x_3}$ is partitioned into sub-domains, which are allocated to different processes. In this work, we use the parallel algebraic multigrid solver BoomerAMG from the well-known HYPRE numerical library [FY02, FJY06] as a preconditioner for the parallel conjugate gradient method. We use default BoomerAMG parameter settings for 3D elliptic problems.

Parallel performance results on strong scaling of the developed pseudo - parabolic solver are presented in Table 5. 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 DOF$/p = N_{x_1} \times N_{x_2} \times N_{x_3}/p$ 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. We show the total BoomerAMG setup time T_{set}, i.e. summed up for all time steps, and the total solution time T_{sol} of the parallel conjugate gradient solver with N_{iter} iterations in total. Finally, we present the obtained values of parallel speed-up $S_p = T_1/T_p$ and parallel efficiency $E_p = S_p/p$. We also calculate and present the scaled parallel efficiency

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

to remove the effect of slight increasing number of CG iterations.

Table 5: Total wall time T_p, speed-up S_p, and efficiency E_p solving the test problem with parallel pseudo-parabolic solver and $N_{x_1} = N_{x_2} = N_{x_3} = 128$, $M = 256$.

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	$\widehat{E}_p$
1	1×1	$1 \times 1 \times 1$	$2.1 \cdot 10^6$	2986.4	1723.5	967.7	1626	1.00	1.00	1.00
2	1×2	$1 \times 2 \times 1$	$1.0 \cdot 10^6$	1954.3	1277.2	531.9	1757	1.53	0.76	0.83
4	1×4	$1 \times 2 \times 2$	$5.2 \cdot 10^5$	1122.9	763.0	284.2	1811	2.66	0.66	0.74
8	1×8	$2 \times 2 \times 2$	$2.6 \cdot 10^5$	618.1	413.0	166.2	1957	4.83	0.60	0.73
16	1×16	$2 \times 4 \times 2$	$1.3 \cdot 10^5$	381.8	242.7	111.0	1969	7.82	0.49	0.59
32	2×16	$4 \times 4 \times 2$	$6.6 \cdot 10^4$	216.0	142.8	58.8	1989	13.83	0.43	0.53
64	4×16	$4 \times 4 \times 4$	$3.3 \cdot 10^4$	142.6	99.1	39.0	2024	20.95	0.33	0.41
128	8×16	$4 \times 8 \times 4$	$1.6 \cdot 10^4$	112.2	78.4	32.2	2082	26.61	0.21	0.27
256	16×16	$4 \times 8 \times 8$	$8.2 \cdot 10^3$	111.2	76.0	34.6	2138	26.86	0.10	0.14
512	32×16	$8 \times 8 \times 8$	$4.1 \cdot 10^3$	152.8	107.2	46.4	2162	19.55	0.04	0.05

The setup costs of parallel BoomerAMG preconditioner are large: 2-3 times bigger than the elapsed times of parallel CG iterations. For structured meshes, geometric multigrid methods should be considered as preconditioners. However, applied 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. Our study have also shown that 3D partitioning is better than 2D partitioning mainly due to slightly smaller setup costs of parallel BoomerAMG preconditioner.

Degradation of the efficiency of the parallel algorithm is clearly seen when the number of processes is increased. This effect is well known for parallel linear solvers with decreasing DOF$/p$ and increasing amount of communications.

3.4.2 Parallel integral solver

Solution of the non-local fractional diffusion problem (11) is transformed into a computation of two sums (16). Corresponding 3D elliptic subproblems can be solved independently, what gives a first level of parallelization. On a second level, each 3D elliptic subproblem can be also solved in parallel.

In the two-level parallel algorithm, parallel processes are split into some number of groups. Values y_j of sums (16) can be distributed between these groups of processes statically or dynamically. For each received y_j value, corresponding group of processes solves the 3D elliptic problem

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

using the domain decomposition method. To implement the two-level parallel algorithm, we use the parallel programing templates [ČSTR16] and the parallel algebraic multigrid solver BoomerAMG from the well-known HYPRE numerical library [FY02, FJY06] as a preconditioner for the parallel conjugate gradient method.

Parallel performance results on strong scaling of the developed integral solver are presented in Table 6. 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, g is the number of used groups of processes, $P_1 \times P_2 \times P_3$ denotes the topology of domain decomposition. "Mem" is the amount of memory used by the solver in GB. The total wall time T_p is given in seconds. s_g is the standard deviation of solution times of g groups, where T_p is the maximum. We also show the maximal total BoomerAMG setup time T_{set}, i.e. summed up for all y_j tasks received by some group, and the maximal total solution time T_{sol} of the parallel conjugate gradient solver with N_{iter} iterations in total. Finally, we present the obtained values of parallel speed-up $S_p = T_1/T_p$ and efficiency $E_p = S_p/p$.

Some conclusions follow from Table 6. First, the parallel integral solver with one group ($g = 1$) is only slightly slower than the parallel pseudo-parallel solver. We remind here that the same number of 3D elliptic subproblems is solved in these tests and solution obtained with integral solver is much more accurate (see Table 4). Interesting to note, that there is a noticeable difference in the number of CG iterations - N_{iter}.

The setup times of parallel BoomerAMG preconditioner are relatively smaller for the integral solver. Although, they are still very significant and exceed the time of CG iterations. As one can see, the setup times are minimal when $g = p$, i.e. 3D elliptic subproblems are solved sequentially. However, in this case the memory requirements of integral solver are growing very fast, because up 16 3D elliptic problems need to be solved on one node simultaneously. Third, our tests have shown that a static cyclic distribution can be used for y_j tasks on the first level of parallelization, since obtained values of standard deviation s_g are quite small.

3.4.3 Conclusions

The performance results of the parallel integral solver are very promising. The computations in the forth extended dimension are formulated in this algorithm as evaluation of two integral sums (16). The calculation of these sums is easily parallelizable, and requires minimal amount of communication.

Table 6: Total wall time T_p, speed-up S_p, and efficiency E_p solving the test problem with parallel integral solver and $N_{x_1} = N_{x_2} = N_{x_3} = 128$, $M = 128$.

p	$n_d \times n_c$	$g \times P_1 \times P_2 \times P_3$	Mem	T_p	s_g	T_{set}	T_{sol}	N_{iter}	S_p	E_p
1	1×1	$1 \times 1 \times 1 \times 1$	1.6	3183.6	0.0	1676.0	1353.6	2467	1.00	1.00
2	1×2	$2 \times 1 \times 1 \times 1$	3.1	1586.1	0.2	836.7	678.6	1238	2.01	1.00
2	1×2	$1 \times 1 \times 2 \times 1$	1.6	2090.4	0.0	1266.2	746.0	2655	1.52	0.76
4	1×4	$4 \times 1 \times 1 \times 1$	5.9	846.5	5.7	457.6	354.4	617	3.76	0.94
4	1×4	$2 \times 1 \times 2 \times 1$	3.3	1076.1	4.9	649.1	387.6	1328	2.96	0.74
4	1×4	$1 \times 1 \times 2 \times 2$	1.9	1189.8	0.0	745.7	404.2	2790	2.68	0.67
8	1×8	$8 \times 1 \times 1 \times 1$	11.2	475.3	5.4	270.3	185.3	310	6.70	0.84
8	1×8	$4 \times 1 \times 2 \times 1$	6.6	578.0	5.0	347.2	210.5	667	5.51	0.69
8	1×8	$2 \times 1 \times 2 \times 2$	3.9	620.5	3.9	381.5	217.0	1390	5.13	0.64
8	1×8	$1 \times 2 \times 2 \times 2$	2.6	653.3	0.0	398.2	233.1	2958	4.87	0.61
16	1×16	$16 \times 1 \times 1 \times 1$	23.1	277.2	8.3	154.3	112.3	158	11.49	0.72
16	1×16	$8 \times 1 \times 2 \times 1$	12.9	331.8	3.4	206.2	114.5	338	9.60	0.60
16	1×16	$4 \times 1 \times 2 \times 2$	8.7	365.7	4.8	214.3	139.7	700	8.71	0.54
16	1×16	$2 \times 2 \times 2 \times 2$	6.3	379.5	1.0	210.6	154.5	1483	8.39	0.52
16	1×16	$1 \times 2 \times 4 \times 2$	5.1	405.8	0.0	234.4	156.6	2997	7.85	0.49
32	2×16	$32 \times 1 \times 1 \times 1$	26.7	146.2	7.9	74.1	65.2	81	21.77	0.68
32	2×16	$16 \times 1 \times 2 \times 1$	15.8	173.3	5.9	102.0	65.8	172	18.37	0.57
32	2×16	$8 \times 1 \times 2 \times 2$	10.4	184.6	2.5	109.3	69.1	350	17.24	0.54
32	2×16	$4 \times 2 \times 2 \times 2$	7.2	191.2	1.2	106.3	77.8	739	16.65	0.52
32	2×16	$2 \times 2 \times 4 \times 2$	6.6	203.5	0.9	117.6	78.5	1505	15.64	0.49
32	2×16	$1 \times 2 \times 4 \times 4$	5.0	231.7	0.0	141.3	83.2	3034	13.74	0.43
64	4×16	$64 \times 1 \times 1 \times 1$	26.8	74.3	5.6	37.0	33.4	41	42.83	0.67
64	4×16	$8 \times 2 \times 2 \times 2$	7.9	96.9	1.0	53.5	39.7	376	32.85	0.51
64	4×16	$1 \times 4 \times 4 \times 4$	6.0	150.7	0.0	94.6	54.0	3084	21.13	0.33
128	8×16	$128 \times 1 \times 1 \times 1$	26.9	38.6	4.1	19.0	17.2	21	82.39	0.64
128	8×16	$16 \times 2 \times 2 \times 2$	8.1	48.8	1.1	27.2	19.8	187	65.24	0.51
128	8×16	$1 \times 4 \times 8 \times 4$	6.3	117.4	0.0	73.8	43.4	3141	27.12	0.21
256	16×16	$256 \times 1 \times 1 \times 1$	26.8	20.4	3.0	9.2	9.3	11	155.80	0.61
256	16×16	$32 \times 2 \times 2 \times 2$	9.1	25.0	1.1	13.9	10.3	96	127.57	0.50
256	16×16	$1 \times 4 \times 8 \times 8$	7.3	117.3	0.0	71.6	46.8	3257	27.15	0.11
512	32×16	$256 \times 1 \times 2 \times 1$	13.7	12.8	1.8	6.4	5.1	12	247.80	0.48
512	32×16	$64 \times 2 \times 2 \times 2$	9.6	13.2	0.9	7.0	5.6	48	241.98	0.47
512	32×16	$1 \times 8 \times 8 \times 8$	9.3	162.2	0.0	99.5	62.4	3310	19.63	0.04

References

[BJ15] A. Bonito and Pasciak J. Numerical approximation of fractional powers of elliptic operator. *Mathematics of Computation*, 84, 2015.

[ČSTR16] 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.

[ČT14] 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.

[FJY06] R. Falgout, J. Jones, and U. Yang. The design and implementation of Hypre, a library of parallel high performance preconditioners. In *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.

[FY02] R. Falgout and U. Yang. Hypre: A library of high performance preconditioners. In *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.

[HLM$^+$16] S. Harizanov, R. Lazarov, P. Marinov, S. Margenov, and Y. Vutov. Optimal solvers for linear systems with fractional powers of sparse SPD matrices. *arXiv:submit/1751899*, 2016.

[NM65] J.A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[NOS15a] R. Nochetto, E. Otárola, and A. Salgado. A PDE approach to numerical fractional diffusion. Proceedings of the 8th ICIAM, Beijing, China, pages 211–236, 2015.

[NOS15b] R.H. Nochetto, E. Otárola, and A.J. Salgado. A PDE approach to fractional diffusion in general domains: a priori error analysis. *Foundations of Computational Mathematics*, 15(3):733–791, 2015.

[Vab16] 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.

[Wan81] H.H. Wang. A parallel method for tridiagonal equations. *ACM Transactions on Mathematical Software (TOMS)*, 7(2):170–183, 1981.