

Tomáš Kozubek · Martin Čermák
Petr Tichý · Radim Blaheta
Jakub Šístek · Dalibor Lukáš
Jiří Jaroš (Eds.)

LNCS 11087

High Performance Computing in Science and Engineering

Third International Conference, HPCSE 2017
Karolinka, Czech Republic, May 22–25, 2017
Revised Selected Papers

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7407>

Tomáš Kozubek · Martin Čermák
Petr Tichý · Radim Blaheta
Jakub Šístek · Dalibor Lukáš
Jiří Jaroš (Eds.)


High Performance Computing in Science and Engineering

Third International Conference, HPCSE 2017
Karolinka, Czech Republic, May 22–25, 2017
Revised Selected Papers

Editors

Tomáš Kozubek
VSB - Technical University of Ostrava
Ostrava-Poruba
Czech Republic

Martin Čermák 
VSB - Technical University of Ostrava
Ostrava-Poruba
Czech Republic

Petr Tichý 
Charles University in Prague
Prague
Czech Republic

Radim Blaheta
Institute of Geonics of the CAS
Ostrava-Poruba
Czech Republic

Jakub Šístek
University of Manchester
Manchester
UK

Dalibor Lukáš
VSB - Technical University of Ostrava
Ostrava-Poruba
Czech Republic

Jiří Jaroš 
Brno University of Technology
Brno
Czech Republic

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-97135-3 ISBN 978-3-319-97136-0 (eBook)
<https://doi.org/10.1007/978-3-319-97136-0>

Library of Congress Control Number: 2018949143

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume comprises the proceedings of the Third International Conference on High Performance Computing in Science and Engineering – HPCSE 2017, which was held in the hotel Soláň in the heart of the Beskydy Mountains, Czech Republic, during May 22–25, 2017. The biennial conference was organized by IT4Innovations National Supercomputing Center at VSB - Technical University of Ostrava, and its goal was to bring together specialists in applied mathematics, numerical methods, and parallel computing, to exchange experience, and to initiate new research collaborations. We are glad that our invitation was accepted by distinguished experts from world-leading research institutions.

This conference has become an international forum for exchanging ideas among researchers involved in scientific and parallel computing, including theory and applications, as well as applied and computational mathematics. The focus of HPCSE 2017 was on models, algorithms, and software tools that facilitate efficient and convenient utilization of modern parallel and distributed computing architectures, as well as on large-scale applications.

The scientific committee of HPCSE 2017 comprised Radim Blaheta, Zdeněk Dostál, Tomáš Kozubek, Josef Málek, Zdeněk Strakoš, Jakub Šístek, Miroslav Tůma, and Vít Vondrák.

The plenary talks were presented by:

- Erin Claire Carson from New York University (USA)
- Froilan Dopico from Universidad Carlos III de Madrid (Spain)
- Pavel Jiránek from Siemens Industry Software NV, Leuven (Belgium)
- Luca Frediani from the University of Tromsø (Norway)
- Daniel Loghin from the University of Birmingham (UK)
- Dalibor Lukáš from VŠB - Technical University of Ostrava (Czech Republic)
- Kent-Andre Mardal from the University of Oslo (Norway)
- Catherine Powell from the University of Manchester (UK)
- Stefan Ratschan from the Institute of Computer Science, AS CR, Prague (Czech Republic)
- Yousef Saad from the University of Minnesota, Minneapolis (USA)
- Olaf Schenk from Università della Svizzera italiana, Lugano (Switzerland)
- Radek Tezaur from Stanford University (USA)
- Sivan Toledo from Tel Aviv University (Israel)
- Gerardo Toraldo from Università degli Studi di Napoli Federico II (Italy)
- Ludmil Zikatanov from The Pennsylvania State University, State College (USA)

The conference was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science - LQ1602.” We gratefully acknowledge this support.

The HPCSE 2017 conference was a fruitful event, providing interesting lectures, showcasing new ideas, demonstrating the beauty of applied mathematics, presenting numerical linear algebra, optimization methods, and high performance computing, and starting or strengthening collaborations and friendships.

This meeting attracted more than 100 participants from 10 countries. All participants were invited to submit an original paper to this book of proceedings. We give thanks for all contributions as well as for the work of the reviewers, and hope that this volume will be useful for readers. The proceedings were edited by Tomáš Kozubek, Martin Čermák, Petr Tichý, Radim Blaheta, Jakub Šístek, Dalibor Lukáš, and Jiří Jaroš.

Finally, we would like to cordially invite readers to participate in the next HPCSE conference, which is planned to be held at the same place during May 20–23, 2019.

June 2018

Tomáš Kozubek

Organization

Program Committee

Radim Blaheta	Institute of Geonics of the CAS, Czech Republic
Zdeněk Dostál	VSB - Technical University of Ostrava, Czech Republic
Tomáš Kozubek	VSB - Technical University of Ostrava, Czech Republic
Josef Málek	Charles University in Prague, Czech Republic
Zdeněk Strakoš	Charles University in Prague, Czech Republic
Jakub Šístek	The University of Manchester, UK
Miroslav Tůma	Charles University in Prague, Czech Republic
Vít Vondrák	VSB - Technical University of Ostrava, Czech Republic

Contents

A High Arithmetic Intensity Krylov Subspace Method Based on Stencil Compiler Programs	1
<i>Simplice Donfack, Patrick Sanan, Olaf Schenk, Bram Reys, and Wim Vanroose</i>	
Applications of Trace Estimation Techniques	19
<i>Shashanka Ubaru and Yousef Saad</i>	
Fourier Method for Approximating Eigenvalues of Indefinite Stekloff Operator	34
<i>Yangqingxiang Wu and Ludmil Zikatanov</i>	
Proportionality-Based Gradient Methods with Applications in Contact Mechanics	47
<i>Zdeněk Dostál, Gerardo Toraldo, Marco Viola, and Oldřich Vlach</i>	
Schur Complement-Schwarz DD Preconditioners for Non-stationary Darcy Flow Problems	59
<i>Radim Blaheta, Tomáš Luber, and Jakub Kružík</i>	
Relating Computed and Exact Entities in Methods Based on Lanczos Tridiagonalization	73
<i>Tomáš Gergelits, Iveta Hnětynková, and Marie Kubínová</i>	
Software Tool for Cranial Orthosis Design	88
<i>Milan Jaros, Tomas Karasek, Petr Strakos, and Alena Vasatova</i>	
Implementation of BM3D Filter on Intel Xeon Phi for Rendering in Blender Cycles	101
<i>Milan Jaros, Petr Strakos, and Tomas Karasek</i>	
Investigating Convergence of Linear SVM Implemented in PermonSVM Employing MPRGP Algorithm	115
<i>Jakub Kružík, Marek Pecha, Václav Hapla, David Horák, and Martin Čermák</i>	
Using ESPRESO as Linear Solver Library for Third Party FEM Tools for Solving Large Scale Problems	130
<i>Ondřej Meca, Lubomír Říha, Alexandros Markopoulos, Tomáš Brzobohatý, and Tomáš Kozubek</i>	

MERIC and RADAR Generator: Tools for Energy Evaluation and Runtime Tuning of HPC Applications	144
<i>Ondrej Vysocky, Martin Beseda, Lubomír Říha, Jan Zapletal, Michael Lysaght, and Venkatesh Kannan</i>	
Disc vs. Annulus: On the Bleaching Pattern Optimization for FRAP Experiments.	160
<i>Ctirad Matonoha, Štěpán Papáček, and Stefan Kindermann</i>	
Modeling and Simulation of Microalgae Growth in a Couette-Taylor Bioreactor	174
<i>Štěpán Papáček, Ctirad Matonoha, and Karel Petera</i>	
Karhunen-Loève Decomposition of Isotropic Gaussian Random Fields Using a Tensor Approximation of Autocovariance Kernel.	188
<i>Michal Běreš</i>	
A Bayesian Approach to the Identification Problem with Given Material Interfaces in the Darcy Flow	203
<i>Simona Domesová and Michal Běreš</i>	
Author Index	217



A High Arithmetic Intensity Krylov Subspace Method Based on Stencil Compiler Programs

Simplice Donfack¹, Patrick Sanan¹, Olaf Schenk^{1(✉)}, Bram Reys²,
and Wim Vanroose²

¹ Institute of Computational Science, Università della Svizzera italiana (USI),
Lugano, Switzerland

{simplice.donfack,patrick.sanan,olaf.schenk}@usi.ch

² University of Antwerp, Antwerp, Belgium

{bram.reys,wim.vanroose}@uantwerpen.be

Abstract. Stencil calculations and matrix-free Krylov subspace solvers represent important components of many scientific computing applications. In these solvers, stencil applications are often the dominant part of the computation; an efficient parallel implementation of the kernel is therefore crucial to reduce the time to solution. Inspired by polynomial preconditioning, we remove upper bounds on the arithmetic intensity of the Krylov subspace building block by replacing the matrix with a higher-degree matrix polynomial. Using the latest state-of-the-art stencil compiler programs with temporal blocking, reduced memory bandwidth usage and, consequently, better utilization of SIMD vectorization and thus speedup on modern hardware, we are able to obtain performance improvements for higher polynomial degrees than simpler cache-blocking approaches have yielded in the past, demonstrating the new appeal of polynomial techniques on emerging architectures. We present results in a shared-memory environment and an extension to a distributed-memory environment with local shared memory.

Keywords: Stencil compilers · Performance engineering
Krylov methods · Code generation · Autotuning · HPC · CG
Polynomial preconditioning

1 Introduction

Simulation as a discipline relies on increasingly compute-intensive models that require ever more computational resources. Many simulations in science and engineering (e.g., fluid dynamics, meteorology, and geophysics) are based on implicit time-stepping computations using Krylov subspace methods to solve systems based on stencil operators on structured grids. As a result, these applications represent an important high-performance computing software pattern on the current generation of processor architectures [1]. Advancing both stencil

grid applications and matrix-free Krylov subspace linear solvers is of utmost importance for implicit time integration applications on structured grids.

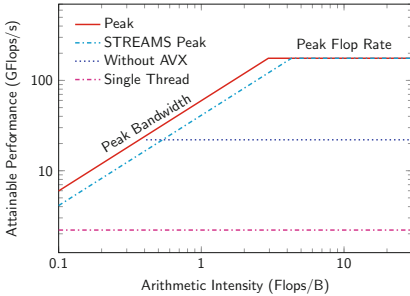


Fig. 1. Roofline model based on the Intel Xeon 2660 v2 Ivy Bridge microarchitecture used in the performance experiments in this paper.

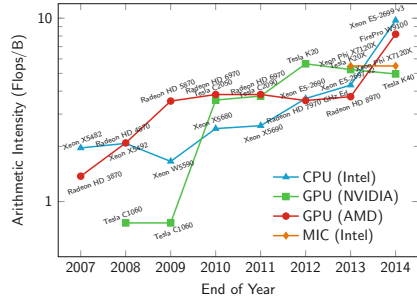


Fig. 2. Double-precision arithmetic intensity to fully utilize the floating-point capabilities of high-performance processing units; reproduced from [28].

Structured grid applications require relatively few arithmetic operations per byte read or written. Vector processor unit performance is thus limited by memory bandwidth rather than compute power, causing structured grid applications to perform poorly on current microarchitectures. The large number of components and machine complexity of future manycore-based architectures will further exacerbate the issue.

Thus, arithmetic intensity, the number of floating point operations executed per byte fetched from main memory, has become a more important metric to evaluate numerical algorithms than the number of required arithmetic operations. Figure 1 shows the relationship between attainable performance and arithmetic intensity on a recent microarchitecture. Below a critical value of arithmetic intensity, performance is limited by memory bandwidth. Stencil-based operations typically fall far below this value. Vectorization (e.g., AVX) allows higher peaks, but requires a higher arithmetic intensity to achieve them. Figure 2 shows the recent increase of the critical value on high-end hardware, plotting the required arithmetic intensity to transition between compute-limited and memory-bandwidth-limited regimes; this trend highlights the need for algorithms to allow simulation codes to adapt.

A given algorithm has a maximum arithmetic intensity, assuming perfect data reuse. Increasing the theoretical maximum is inconsequential if data reuse (with the help of hardware caches) is too suboptimal. Naïve implementations of stencil applications do not come close to achieving this maximum. Blocking strategies are more performant, yet the optimal implementation depends on the details of the cache(s) available. Hence, stencil-code engineering to increase arithmetic intensity has received increased attention in the last few years, evidenced by the appearance of a number of stencil-code programming languages and compilers, such as POCHOIR [31], PLUTO [6], PATUS [11], MODESTO [23], and GSCL [5].

In this paper, we make a novel combination of the two approaches to improving performance of stencil-based codes: we simultaneously increase the theoretical maximum arithmetic intensity by reformulating the Krylov solver, and utilize stencil compilers to approach this maximum on current hardware. Both ingredients are essential.

An iteration of a Krylov subspace method includes elementwise vector operations, vector reductions (dot products and norms), and matrix-vector multiplications that are tightly interwoven. This is illustrated in Figs. 3 and 4. In Fig. 3, a 7-point stencil results in the update of a single grid point, after which the stencil is applied repeatedly. In Fig. 4, however, in-between the stencil updates there are additional, interdependent dot products and vector updates. The resulting data dependencies prevent the solver from efficiently applying the subsequent in-place matrix-vector products required both to increase the theoretical arithmetic intensity and to take advantage of advanced stencil engineering techniques (“temporal blocking”).

```

1: for t = 1 to nu
2:   for i = 1 to nx
3:     for j = 1 to ny
4:       for k = 1 to nz
5:          $U_{i,j,k}^{t+1} \leftarrow 6 * U_{i,j,k}^t$ 
6:          $-U_{i-1,j,k}^t - U_{i+1,j,k}^t$ 
7:          $-U_{i,j-1,k}^t - U_{i,j+1,k}^t$ 
8:          $-U_{i,j,k-1}^t - U_{i,j,k+1}^t$ 

```

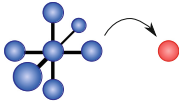


Fig. 3. Stencil example: three-dimensional (3D) Laplacian, 7-point stencil with constant coefficients.

```

1: Initialize X, R and P.
2: for t = 1 to nu do
3:   for i = 1 to nx do
4:     for j = 1 to ny do
5:       for k = 1 to nz do
6:          $S_{i,j,k}^{t+1} \leftarrow 6 * P_{i,j,k}^t$ 
7:          $-P_{i-1,j,k}^t - P_{i+1,j,k}^t$ 
8:          $-P_{i,j-1,k}^t - P_{i,j+1,k}^t$ 
9:          $-P_{i,j,k-1}^t - P_{i,j,k+1}^t$ 
10: dot products and vector updates
    with X, S, P, and R

```

Fig. 4. More complicated data dependencies inside the conjugate gradient (CG). Dot products and vector updates that use the results of the dot product limit the arithmetic intensity and make it impossible to use temporal blocking out of the box. See also Algorithm 1.

We use the concept of polynomial preconditioning to reformulate these isolated matrix-vector multiplications with matrix-polynomial-vector multiplications $p_m(A)$ with higher arithmetic intensity. The derived stencil computation can be optimized separately using an efficient stencil computation approach. From a numerical point of view, the search space constructed with a low-order

polynomial $p_m(A)$ might be suboptimal compared to the Krylov subspace based on regular powers of A and therefore needs more cumulative applications of the matrix A . However, due to the fact that data are better reused and that vector units can be used more effectively, overall time to solution can decrease.

This approach differs from s -step methods wherein intermediate vectors are stored and later orthogonalized using TSQR [26]. Our approach communicates only initial and final vectors, increasing the volume of computation and hence the arithmetic intensity. This difference reflects the different objectives of s -step preconditioners (removing the lower bounds on the amount of communication) and our approach (removing upper bounds on the arithmetic intensity achievable).

In this work, we leverage stencil compilers, specific code generation, and autotuning methodology to make Krylov subspace kernels both code- and performance-efficient, in a novel way that critically depends on the combination of polynomial reformulation and the use of advanced stencil compilers. After stencil computation and their challenges are discussed in Sect. 2, in Sects. 3–4, we present a careful performance-oriented reformulation of a polynomial Krylov subspace solver, analyze the performance of automatic stencil code generation in this context, and show scalability on current Intel microarchitecture. Section 5 presents results of the approach in two-dimensional experiments, and we conclude with Sect. 6 which demonstrates the implementation and performance of the technique in a three-dimensional, distributed-memory environment.

2 Node-Level Stencil Performance Engineering

Extracting good performance from stencil-based codes can be challenging because their maximum arithmetic intensity is rather low and thus performance is limited by the bandwidth between memory and compute units. Moreover, if the application requires that the stencil be applied multiple times to each of the spatial grids point, there is potential for performance increase by exploiting temporal data locality, i.e., reuse of the computed data before they are transferred back to memory. As an example, the image in Fig. 3 shows a visualization of the stencil structure of a second-order discretization of the 3D Laplacian.

A fair amount of research has addressed the algorithmic changes and code transformations needed. Loop-tiling approaches [17, 22] and compilers using the polyhedral model fall into this category. This is used to determine good tile sizes as well as for autoparallelization. However, only a few stencil compilers can carry out the requested, mostly nontrivial transformations for practical usage. We use the following tools in this work:

- **Patus** [10–12] decouples the algorithm from the scheduling of the computation, i.e., what is vectorized, which loops are tiled or unrolled, and which ones are parallelized. Thus, the “schedule” can be (auto)tuned for the target architecture. The goal is to enable a clean implementation of an algorithm (as opposed to C code cluttered by optimizations) while still delivering the

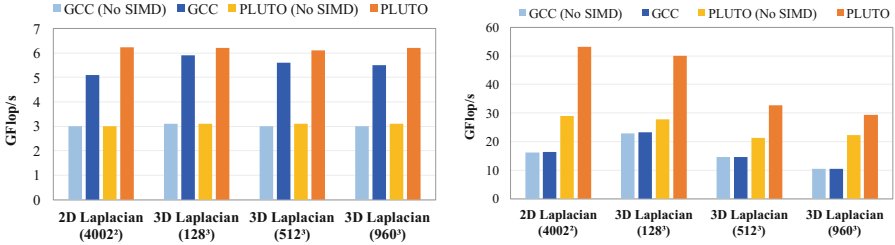


Fig. 5. Impact of AVX vectorization for various stencils using 1 (left) and 10 cores (right) on one socket of an Intel Xeon 2660 v2 Ivy Bridge.

performance of well-optimized implementations. PATUS takes advantage of spatial blocking but temporal blocking is not yet available.

- **Pluto** [7] is a research source-to-source (C-to-C) compiler using the polyhedral model. Recently, PLUTO was extended to automatically detect iterative stencil computations and to apply both spatial- and time-skewing, allowing concurrent processing of space-time tiles [34]. In this work, we use diamond-tiling loop transformations available in [7] which are more efficient on current microarchitectures. A recent advanced diamond-tiling implementation and performance analysis is also available in [25].

SIMD vectorization has proven to be the key optimization for numerous stencils in all data-centric stencil compiler programs, especially in view of growing SIMD vector width. Using the hardware’s SIMD units in an optimal way is critical for performance; even if one trusts the compiler to vectorize scalar code one can observe that explicitly vectorized code using SIMD intrinsics yields significantly better performance. General purpose compilers must be on the safe side and apply vectorization conservatively. PATUS and PLUTO can generate code using explicit SIMD intrinsics. The benefit is that in the non-unit-stride dimensions no loads are needed which are not aligned at SIMD vector boundaries, as unaligned loads may incur a latency penalty or may not even be supported on other architectures and therefore require a workaround.

However, it is well known that extra care is needed when using SIMD on multiple cores in the presence of limited bandwidth. This is illustrated in Fig. 5, showing AVX vectorization results using PLUTO and the GNU GCC compiler (4.9.2). Nonvectorized results are shown as PLUTO (No SIMD) and GCC (No SIMD), whereas AVX vectorization has been exploited in the GCC and PLUTO cases. Double-precision performance is shown for 5- and 7-point Laplacian stencils on a single socket Intel Xeon 2660 v2 Ivy Bridge node, using 1 and 10 hardware threads. While the automatic vectorization SIMD code generation methods based on the GNU compiler always work well on a single core (with an acceleration of $2.0\times$ for PLUTO), it is noticeable that these results cannot automatically be transferred to multiple cores on a socket. As shown in Fig. 6, on multiple cores, the GNU compiler reaches the memory bandwidth limitation of 41.1 GB/s for

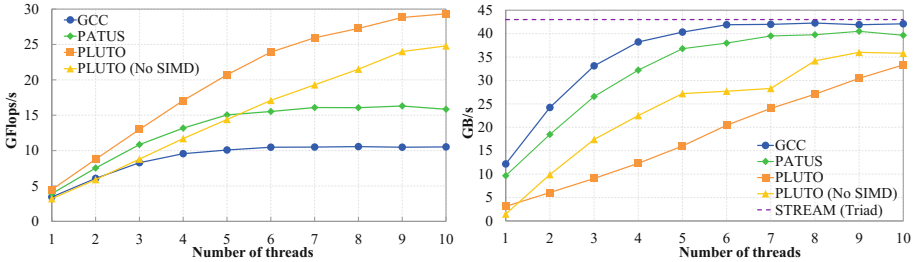


Fig. 6. Performance (left) and bandwidth (right) measurements of a 3D Laplacian of size $n = 960^3$ using a 7-point stencil with constant coefficients.

this memory-bound stencil computation¹. As a result, there is no improvement from SIMD vectorization. However, if we use techniques such as diamond-tiling loop transformations with PLUTO, we reduce the memory-bandwidth pressure and the beneficial impact of SIMD vectorization can be observed even by using up to 10 hardware threads. The trend is also visible in Fig. 6, which shows the performance and measured bandwidth of the Laplacian 7-point stencil with constant coefficients when the number of threads varies. PATUS is highly optimized software and includes optimization techniques beyond autovectorization, so is omitted from Fig. 5. Figure 6 shows that it yields better performance than the naïve approach but does not solve the scalability and bandwidth bottleneck issues. In the rest of the paper, we use PATUS as reference for the best spatial blocking implementation. The main question is how a Krylov subspace method, with much more complicated data dependencies, may benefit from SIMD vectorization, fully utilize all cores without saturating the bandwidth, and take advantage of efficient stencil compilers.

3 Increasing Arithmetic Intensity of Krylov Subspace Methods

Time spent in linear solvers often dominates total execution time in computational science applications, as processor count and problem size increase [9, 33]. Krylov subspace methods are commonly used within large-scale, distributed-memory codes. These involve applications of operators, such as the stencils we have considered, interspersed with global reductions (inner products and norms); mitigating the bottleneck created by these operations at extreme scale has prompted the development of methods to hide, as in pipelined Krylov methods [19, 20], or avoid, as in s -step methods [13, 14], these reductions. Reductions have another detrimental effect beyond the resources required to compute them: they constitute barriers which impose an upper bound on arithmetic intensity, even with perfect data reuse.

¹ We also compared the impact on the vectorization with the Intel *icc* compilers and obtained similar results.

Large, sparse linear systems of the form

$$Ax = b \tag{1}$$

can be solved with Krylov subspace methods [35] which accept an initial guess $x^{(0)}$ and generate a sequence of iterates $x^{(k)}$ such that the corrections $x^{(k)} - x^{(0)}$ lie in nested subspaces. The iterates satisfy a method-specific optimality criterion and converge toward the solution x . The Krylov subspace of degree k is the space of matrix polynomials of degree less than k applied to $r^{(0)} \doteq b - Ax^{(0)}$, the initial residual:

$$x^{(k)} - x^{(0)} \in \mathcal{K}_k(A, r^{(0)}) = \text{span}\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \dots, A^{k-1}r^{(0)}\}.$$

Depending on the properties of A , the minimization procedure in the search space can be simplified. For example, the CG method [27], shown in Algorithm 1, does not require explicit storage of basis vectors for $\mathcal{K}_k(A, r^{(0)})$ when A is symmetric and positive definite.

Algorithm 1. CG.

```

1:  $r \leftarrow b - Ax^{(0)}$ 
2:
3:  $p \leftarrow r$ 
4:  $\rho_{\text{old}} \leftarrow \|r\|^2$ 
5: for  $i = 0, 1, \dots$  do
6:    $s \leftarrow Ap$ 
7:    $\alpha \leftarrow \rho_{\text{old}} / \langle s, p \rangle$ 
8:    $x \leftarrow x + \alpha p$ 
9:    $r \leftarrow r - \alpha s$ 
10:   $\rho_{\text{new}} \leftarrow \|r\|^2$ 
11:
12:   $\beta \leftarrow \rho_{\text{new}} / \rho_{\text{old}}$ 
13:   $p \leftarrow r + \beta p$ 
14:   $\rho_{\text{old}} \leftarrow \rho_{\text{new}}$ 
15: end for

```

Algorithm 2. Preconditioned CG.

```

1:  $r \leftarrow b - Ax^{(0)}$ 
2:  $z \leftarrow M^{-1}r$ 
3:  $p \leftarrow z$ 
4:  $\rho_{\text{old}} \leftarrow \langle r, z \rangle$ 
5: for  $i = 0, 1, \dots$  do
6:    $s \leftarrow Ap$ 
7:    $\alpha \leftarrow \rho_{\text{old}} / \langle s, p \rangle$ 
8:    $x \leftarrow x + \alpha p$ 
9:    $r \leftarrow r - \alpha s$ 
10:   $\rho_{\text{new}} \leftarrow \langle r, z \rangle$ 
11:   $z \leftarrow M^{-1}r$ 
12:   $\beta \leftarrow \rho_{\text{new}} / \rho_{\text{old}}$ 
13:   $p \leftarrow z + \beta p$ 
14:   $\rho_{\text{old}} \leftarrow \rho_{\text{new}}$ 
15: end for

```

The residual after k steps, $r^{(k)} \doteq b - Ax^{(k)}$, can be written as a polynomial of A of degree k , applied to the initial residual $r^{(0)}$:

$$r^{(k)} = b - A \left(x^{(0)} + \sum_{j=0}^{k-1} \gamma_j A^j r^{(0)} \right) = r^{(0)} + \sum_{j=0}^{k-1} \gamma_j A^{j+1} r^{(0)} = P_k(A)r^{(0)}.$$

The operations in Algorithm 1 are typically limited by available memory bandwidth and thus cannot take advantage of additional arithmetic operations available in modern devices. Consider a matrix A with n_{nz} nonzero elements. Computing a matrix-vector product, as on line 6 of Algorithm 1, requires reading

n_{nz} matrix elements and at least n elements from the input vector and writing n elements of the output vector. The number of arithmetic operations is $2n_{\text{nz}} - n$.² Thus the arithmetic intensity is at most

$$q = \frac{2n_{\text{nz}} - n}{b(n_{\text{nz}} + 2n)} \leq \frac{2}{b}, \quad (2)$$

where b is the number of bytes required to represent a scalar.³ This represents a small fraction of the arithmetic intensity required to reach peak performance (see Fig. 2).

For matrices described by applying a stencil on a computational grid, matrix entries may be computed instead of read. If no entries need be retrieved from memory, neglecting any floating point operations required to compute the coefficients, the maximum arithmetic intensity of a *matrix-free* operator application can be expressed as

$$q_{\text{matfree}} = \frac{2n_{\text{nz}} - n}{2bn} \leq \frac{n_{\text{nz}}}{bn}. \quad (3)$$

This quantity is characterized by the average number of nonzero entries per row, divided by b . Thus, for some operators, q_{matfree} can be significantly greater than q , but is nonetheless a fixed function of A , and may not be suitable for a given microarchitecture.

However, when m matrix-vector products are done in-place, with the operator A applied repeatedly to v to produce $A^m v$, $m(2n_{\text{nz}} - n)$ operations are performed for a single read of the matrix, one read of the input vector, and one write of the output vector. This results in maximum arithmetic intensities of

$$q = \frac{m(2n_{\text{nz}} - n)}{b(n_{\text{nz}} + 2n)}, \quad q_{\text{matfree}} = \frac{m(2n_{\text{nz}} - n)}{2bn}. \quad (4)$$

These quantities increase linearly with m . The expressions assume that each element of v is read exactly once from main memory, typically impossible, and as such actual attained arithmetic intensity depends on the details of the cache hierarchy and the implementation of the polynomial kernel.

In light of Fig. 1, it is now apparent that one can attempt to improve the performance of Krylov subspace methods by introducing subsequent matrix-vector products instead of isolated matrix-vector products per iteration.

Polynomial preconditioning offers a natural, tunable way to accomplish this, reducing the number of iterations required for convergence of the Krylov subspace method without using additional memory bandwidth. It was first described shortly after CG [29] and became of greater interest with the advent of vector computers [15]. For more, see Sect. 5 of Saad's survey [30]. Multiply (1) on both sides by the polynomial $q_{m-1}(A)$,

$$p_m(A)x = \tilde{b} \doteq q_{m-1}(A)b, \quad (5)$$

² n_{nz} multiplies and $n_{\text{nz}} - n$ adds.

³ For instance, $b = 8$ for double-precision floating point numbers, giving $q < \frac{1}{4}$.

where $p_m(A) \doteq q_{m-1}(A)A$, producing an equivalent linear system with a new operator. Algorithm 2 shows the preconditioned version of the CG method (PCG), where a preconditioning step is explicitly computed on line 11 with the application of a preconditioning operator M^{-1} . This generalized algorithm is useful when the preconditioner is given by a (black box) routine such as algebraic multigrid or ILU decomposition and reduces to Algorithm 1 when $M = I$. However, as $p_m(A)$ is symmetric for symmetric A and can be chosen to be positive definite for positive definite A , polynomial preconditioning only requires standard CG, as in Algorithm 1; line 6 simply computes $s = p_m(A)p_i$ instead of $s = Ap_i$, and b is replaced with \tilde{b} . This corresponds to searching for an optimal error correction in a Krylov subspace based on powers of $p_m(A)$:

$$x^{(k)} - x^{(0)} \in \mathcal{K}_k(p_m(A), r^{(0)}) = \text{span}\{r^{(0)}, p_m(A)r^{(0)}, \dots, p_m(A)^{k-1}r^{(0)}\}.$$

Note that $\mathcal{K}_k(p_m(A), r_0) \subset \mathcal{K}_{m(k-1)+1}(A, r^{(0)})$ since $p_m(A)r_0$ can be written as a linear combination of the first m vectors of $\mathcal{K}_M(A, r^{(0)})$ for $M > m$. As a result, the residual can be expressed as a polynomial P of the matrix polynomial $p_m(A)$ applied to the initial residual, $r^{(k)} = P(p_m(A))r^{(0)}$.⁴

As discussed in Sect. 4, the polynomial $p_m(A)$ should be defined such that it reduces the number of iterations of the Krylov subspace method. Although the number of arithmetic operations per iteration increases, the overall time to solution is often lower than in the unpreconditioned case.

All other operations in Algorithm 1 are vector updates or dot products with a lower arithmetic intensity than matrix-vector multiplication; reducing the number of total iterations reduces the number of times these must be performed.

The routine to apply the polynomially preconditioned operator $p_m(A)$ can be provided as a black box to a CG solver, allowing it to be optimized with a stencil compiler. This further reduces the demand on memory bandwidth, allowing for arithmetic intensity closer to the ideal in (4) than for a single application of the matrix or a naïvely implemented polynomial kernel. To realize this benefit without the use of a stencil compiler would require expert, hardware-specific implementation. The total number of matrix-vector multiplies with A required to reach a given convergence tolerance typically increases with the use of a polynomially preconditioned system, as opposed to an unpreconditioned one, but time to solution can decrease since the average time per application decreases.

4 Selection and Implementation of the Polynomial Kernel

As test cases, we use simple constant-coefficient Poisson problems in 2 and 3 dimensions,

$$\nabla^2 u(x) = f(x), \quad x \in \mathbb{R}^{\{2,3\}}, \quad (6)$$

discretized with central finite differences, leading to the familiar 5- and 7-point stencils.

⁴ This contrasts with s -step Krylov methods [14] that compute iterates in standard Krylov spaces, s basis vectors at a time.

Algorithm 3. Recursive calculation of matrix vector product $w_m = p_m(A)v$.

```

1:  $\theta = (\lambda_{max} + \lambda_{min})/2$ ,  $\delta = (\lambda_{max} - \lambda_{min})/2$ 
2:  $\sigma_1 = \theta/\delta$ 
3:  $\rho_0 = 1/\sigma_1$ 
4:  $w_0 = 0$ ,  $w_1 = \frac{1}{\theta}Av$ 
5:  $\Delta w_1 = w_1 - w_0$ 
6: for  $k = 2, \dots, m$  do
7:    $\rho_{k-1} = 1/(2\sigma_1 - \rho_{k-2})$ 
8:    $\Delta w_k = \rho_{k-1} [\frac{2}{\delta}A(v - w_{k-1}) + \rho_{k-2}\Delta w_{k-1}]$ 
9:    $w_k = w_{k-1} + \Delta w_k$ 
10: end for

```

The choice of polynomials p_m is ultimately problem dependent, and has been studied in the literature [2]. A common goal is to minimize, in some norm, $1 - p_m(\lambda)$ over all eigenvalues λ of A . For symmetric systems, this corresponds to the fact that the condition number and clustering of the spectrum provide useful bounds on the convergence [35]. In the case of the simple Laplacian approximations we have chosen, the spectrum is known analytically [8]. It is well-separated over an interval, motivating the choice of Chebyshev polynomials.

If little is known about the spectrum of A , a good polynomial can be constructed based on the Ritz values of A . These can be estimated using a few iterations of CG and calculating the eigenvalues of the tridiagonal Lanczos matrix that contains α_i and β_i .

The recursive definition of Chebyshev polynomials can be used to define a function to apply $p_m(A)$. Algorithm 3 presents such a recursive routine which computes $w_m = p_m(A)v$, where p_m is a scaled and shifted Chebyshev polynomial that maps the spectrum of A around 1. The algorithm comprises only simple operations (and notably no reductions). Thus, when A can be defined as a stencil application, the entire polynomial application is amenable to automatic tiling and cache blocking. This possibility will be exploited in the experiments described in Sects. 5 and 6.

In order to provide data dependencies suitable for tiling with PLUTO, our polynomial kernel implementation used in Sects. 5 and 6 includes the option to use an auxiliary buffer for the vector to which to apply the polynomial.

5 Single-Node Experimental Results

Throughout this section, we compare three solver variants implementing a polynomial stencil application as described in Algorithm 3:

1. a naïve OPENMP implementation obtained by adding `#pragma omp parallel for` directives around the outer spatial loop to parallelize,
2. an improved approach based on the PLUTO compiler, and
3. a corresponding implementation using PATUS.

Table 1. Relevant hardware information for the Emmy cluster, used for all the experiments in this paper.

Nodes	560	Peak memory	59.7 GB/s
Sockets/node	2	bandwidth / socket	
Socket hardware	Xeon 2660v2 (Ivy Bridge)	Peak STREAMS memory bandwidth	41.1 GB/s
Cores/socket	10	DRAM per node	64 GB
Clock frequency	2.2 GHz	L1 cache per core	32 KB instruction + 32 KB data
Double-precision flops/cycle/core	8	L2 cache per core	256 KB
Peak flop rate/socket	176 GFlops/sec	L3 cache per socket	25 MB

5.1 Experimental Environment

All the experiments were performed on the Emmy cluster at RRZE Erlangen, as described in Table 1. We fix the maximum clock rate to 2.2 GHz, disabling “turbo” mode, do not use hyperthreading, and focus on single-socket performance. We use the LIKWID performance tools [32] to bind threads to cores on one socket and run the STREAMS benchmark to measure an attainable memory bandwidth of 41.1 GB/s. The measured bandwidth corresponds to the rate of data transfer between one socket and main memory. The experiments in this section solve a 3D Poisson problem of size $N = 512$ in each dimension, near the maximum size that can be allocated for a single socket. Here, Dirichlet boundary conditions are set in buffer grid points, and loops proceed over interior points only, with a uniform stencil. Code was compiled using *gcc* with `-O3` optimization and AVX enabled. “Degree m ” refers to the order of the polynomial $p_m(A)v$.

5.2 Krylov Subspace Method Convergence

Figures 7 and 8 illustrate that the reduction in communication costs results in faster times to solution. Figure 7 shows the convergence history of $\|r^{(k)}\|$ as a function of the number of matrix-vector multiplies for polynomial orders up to $m = 4$. Note that, for each value of m , CG minimizes the error in a different norm, though the residuals can still be used to monitor the convergence.

It is clear that the total number of matrix-vector products before convergence increases with m . This is to be expected, as the iterates lie in subspaces of the full Krylov subspace. Note, however, that the number of dot products to reach convergence (proportional to the number of plot markers) decreases, as does the number of sweeps over the vector. This may allow for lower communication overhead and better cache reuse.

Figure 8 presents the same results as Fig. 7 but plots the residual norm versus time. The polynomial of degree 2 converges faster than the polynomial of degree 1 and, subsequently, convergence time then increases with m . This can be explained by the increasing cost of the matrix-vector products m .

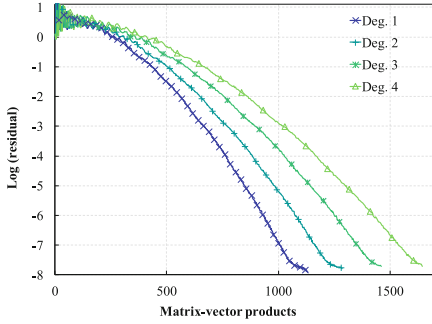


Fig. 7. Polynomial CG convergence as a function of effective matrix-vector multiplies.

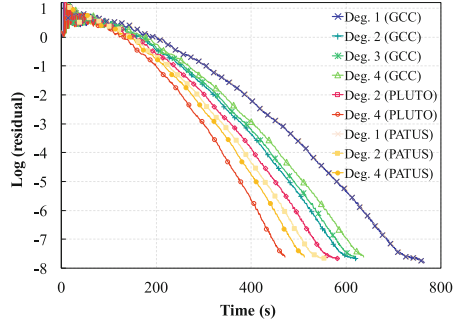


Fig. 8. Polynomial CG convergence using naïve OPENMP, PATUS, and PLUTO.

Figure 8 also shows the improvement in terms of convergence rate from using the PLUTO compiler. The efficient implementation of the polynomial of degree 2 surpasses all variants of the naïve implementation, and maximal speedup is obtained with a polynomial of degree 4.

Figure 9 shows the time per operation using the naïve OPENMP implementation, PLUTO, and PATUS. With the use of the stencil compilers, the matrix-vector product time tends to decrease considerably, resulting in a marked improvement in the time to solution. Using the naïve OPENMP implementation, the total time for the `axpy` and dot product operations decrease as expected with the number of iterations, while the total time for the matrix-vector multiply increases. By implementing the polynomial matrix-vector product $p_m(A)v$ as presented in Algorithm 3 using stencil-based compilers, a speedup of up to $1.5\times$ is attained by using the efficient spatial blocking algorithm PATUS, compared to the naïve implementation. The same algorithm using the PLUTO temporal blocking algorithm results in speedups of up to $2.6\times$.

5.3 Scalability

We present the performance scalability of our implementation when the number of threads varies. Figure 10 shows the total time to solution and the measured bandwidth. At around 6 cores, the naïve implementation reaches a measured bandwidth of 40 GB/s, comparable to the maximum attainable bandwidth as measured by the STREAMS benchmark.

By using PATUS, we observe a small decrease in the measured bandwidth and a speedup in the time to solution of up to $1.5\times$, compared to the naïve OPENMP implementation, thanks to the reuse of cache data. Since data reuse is optimized by PATUS, each core is likely to use the bus less often. However, each new iteration of the matrix-vector product requires a new load of all the associated data. This problem can be solved using PLUTO, thanks to its temporal reuse of data. As shown in the same figure, the PLUTO implementation reduces the pressure

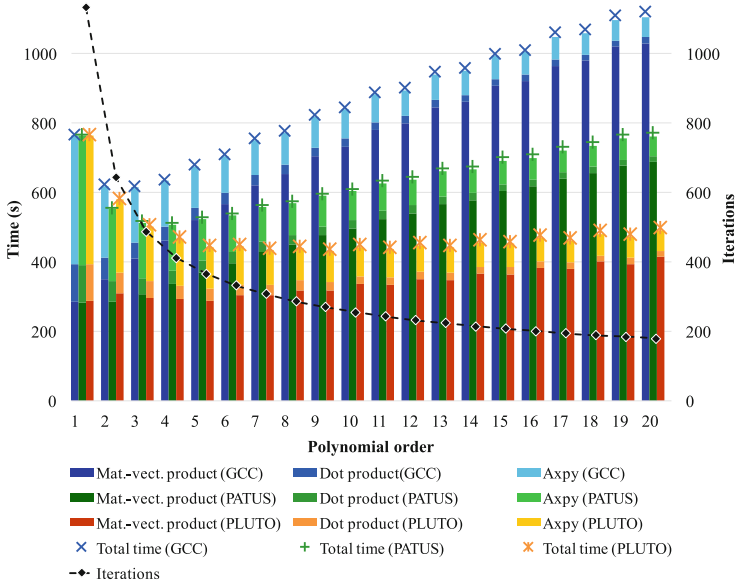


Fig. 9. Time spent for each operation within the OPENMP, PATUS, and PLUTO implementations.

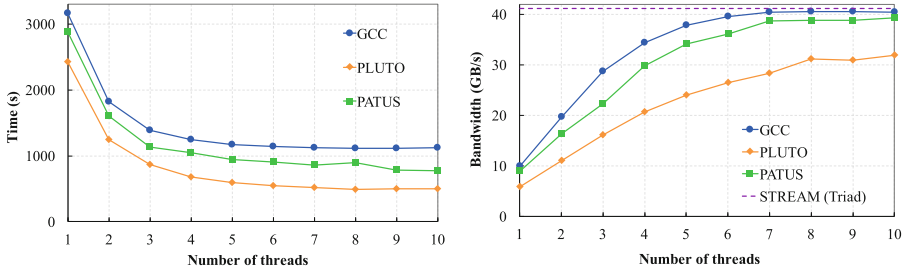


Fig. 10. Performance scalability of our implementation when the number of threads varies. On the left, the time for the solution and, on the right, the measured bandwidth.

on the memory bandwidth when the number of cores increases, and scalability of the algorithm increases considerably. This leads to an implementation up to $2.5\times$ faster than the naïve approach.

6 Multinode Experimental Results

We have shown that the use of a polynomial preconditioner can considerably reduce the iteration count in the CG algorithm. This includes a reduction in the total time spent performing `axpy` and dot product operations.

In this section, we present an implementation and evaluation of our algorithm on multiple nodes as a large-scale, distributed-memory application of our

algorithm in a hybrid MPI/Pthreads environment. In order to reduce the cost of the matrix-vector products, the most time-consuming part of the solver, we use PLUTO to increase the arithmetic intensity of the node-level portion of the underlying stencil-based matrix-vector product. We use a distributed version of the CG algorithm where the communications between different processors running on the nodes are performed by MPI, and within each node we use a multithreaded version of the basic operations such as `axpy`, dot product, and matrix-vector product. In order to parallelize the matrix-vector product, we use OPENMP and PLUTO. Because of the distributed-memory setting, all the data required to compute the local part of the matrix-vector product at each iteration are stored on the corresponding processor, including redundant “halo” data at the boundaries of per-processor subdomains. Boundaries are computed with specialized, nonoptimized stencils over the appropriate grid points.

For a polynomial of degree m , since we perform m matrix-vector products grouped together, it is necessary to store and exchange a halo region m times the width of the stencil. The extra memory required scales as m times the surface area of the subdomains, $mN^{\frac{d-1}{d}}$, which is small outside of the strong-scaling regime. This approach allows the use of OPENMP and PLUTO for the matrix-vector product within each node.

For our tests, we use code leveraging the PETSC library [3,4], linked with the multithreaded BLAS from the Intel MKL. We define our own matrix-free operator which applies the matrix polynomial to data stored on a distributed array (DMDA) object, and use it within PETSC’s CG. The code used to produce the results in this section, including tests, is available under an open-source license⁵. All results are computed on multiple nodes of the same machine used for the single-node experiments, as described in Table 1. Again, we only use a single socket per node. We compare a version of the code parallelized with OPENMP and a second version which uses PLUTO to optimize the polynomial application.

Figure 11 shows the solution time for 2, 4, and 8 nodes when the problem size is fixed at 2048 and the polynomial degree m varies. We observe that while we see strong scaling (and even superlinear strong scaling as the local problem size becomes small enough to fit into the L3 cache), the time to solution using OPENMP tends to increase with the degree of the polynomial. The use of PLUTO helps to remove this limitation, allowing further acceleration by increasing the polynomial degree. Figure 12 shows the solution time when the size of the problem increases for a polynomial of degree 10 using 16 and 32 nodes.

We observe that the OPENMP implementation on 32 nodes is up to 2 times faster than on 16 nodes, and that the use of PLUTO contributes to the best solution. Figure 13 shows strong scaling with a polynomial preconditioner of degree 10 for a small problem of size 1024. Initial superlinear scaling is observed, likely due to cache effects.

⁵ <https://bitbucket.org/psanan/polykrylovpetscexample>.

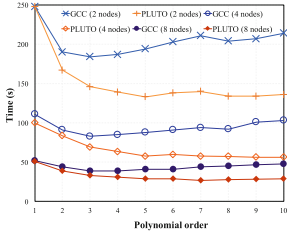


Fig. 11. Solution time for a fixed-size problem, using a polynomial preconditioner on 2–8 distributed-memory nodes.

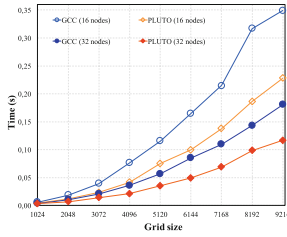


Fig. 12. Time to solution using degree 10 polynomial preconditioning on 16 and 32 distributed-memory nodes.

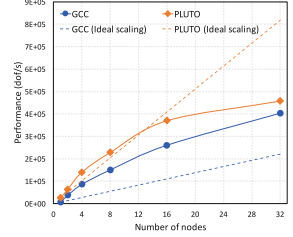


Fig. 13. Strong scaling behavior in time to solution using degree 10 polynomial preconditioning for a small problem of size 1024.

7 Conclusions and Outlook

Krylov subspace solvers use a sparse matrix-vector products, dot products, and vector updates to solve linear system. Each of these building blocks is a low arithmetic intensity operation, limited by available memory bandwidth and implementation’s ability to maximize data reuse. As a result, Krylov methods perform poorly on modern processing units that require, high arithmetic intensity algorithms to operate at peak performance, due to widening SIMD registers and increasing shared-memory concurrency.

Modern stencil compilers including temporal blocking can significantly reduce the bandwidth usage of algorithms with repeated application of the same stencil, for example in explicit time evolution.

In this paper we have shown that the same stencil compilers and temporal blocking techniques can also be used to accelerate Krylov methods, although these algorithms have a much more complicated data dependencies. Our insight is that by using polynomial preconditioning in combination with modern stencil compilers, one can simultaneously increase both the maximum and achieved arithmetic intensity, allowing for demonstrable speedup greatly superior to that of direct application on stencil compilers or polynomial preconditioning alone. Indeed, without advanced stencil compilers (or expert, machine-specific tuning), the polynomial approach here would not be beneficial for higher order polynomials, and without the polynomial approach, maximum arithmetic intensity would be limited to that of a single operator application.

The Poisson problem used as a proof of concept here has explicitly known extremal eigenvalues, required to choose a suitable polynomial. For more general operators, these values must be estimated.

A well-known limitation of polynomial preconditioning is that for most operators of interest and fixed m , the number of iterations to convergence of a Krylov method increases with problem size. This may be mitigated in the future as

communication-avoiding preconditioners [21] are developed, and the methods here may also be used with nested or hierarchical solves. A promising area of application is in those situations where a simple, diagonally-preconditioned CG solve, using $O(1)$ iterations, is used within a larger, scalable solver. Examples include multilevel Krylov methods [16]; indeed, an approach of this kind has been shown to be highly effective in an extreme-scale finite element solver, scaling to hundreds of billions of degrees of freedom on hundreds of thousands of cores, using a hierarchy of simply-preconditioned CG solves [18].

We have focused on finite-difference stencils, with regular access patterns allowing efficient optimization. Many problems of interest, such as the application of finite element operators on unstructured meshes, involve more complex access patterns. Stencil compilers for these cases are in their infancy [24], but clear hardware trends will strongly encourage their development, as indeed they will encourage the use and extension of the methods described in this work.

Acknowledgments. We thank Uday Bondhugula for helpful correspondence and upgrades of PLUTO, Karl Rupp for the data in Fig. 2, and Radim Janalik for initial results used in Fig. 5. We acknowledge the Swiss National Supercomputing Center (CSCS) and the University of Erlangen for computing resources. This research has been funded under the EU FP7-ICT project “Exascale Algorithms and Advanced Computational Techniques” (project reference 610741).

References

1. Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiawicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., Yelick, K.: A view of the parallel computing landscape. *Commun. ACM* **52**(10), 56–67 (2009)
2. Ashby, S.F., Manteuffel, T.A., Otto, J.S.: A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive definite linear systems. *SIAM J. Sci. Stat. Comput.* **13**(1), 1–29 (1992)
3. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zampini, S., Zhang, H.: PETSc users manual. Technical report ANL-95/11 - Revision 3.6, Argonne National Laboratory (2015)
4. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (eds.) *Modern Software Tools in Scientific Computing*, pp. 163–202. Birkhäuser Press, Boston (1997). https://doi.org/10.1007/978-1-4612-1986-6_8
5. Bianco, M., Varetto, U.: A generic library for stencil computations. arXiv preprint [arXiv:1207.1746](https://arxiv.org/abs/1207.1746) (2012)
6. Bondhugula, U., Hartono, A., Ramanujam, J., Sadayappan, P.: Pluto: a practical and fully automatic polyhedral program optimization system. In: *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation (PLDI 2008)*, June 2008. Citeseer, Tucson (2008)
7. Bondhugula, U., Hartono, A., Ramanujam, J., Sadayappan, P.: A practical automatic polyhedral parallelizer and locality optimizer. *ACM SIGPLAN Not.* **43**(6), 101–113 (2008)

8. Briggs, W.L., Henson, V.E., McCormick, S.F.: *A Multigrid Tutorial*, 2nd edn. SIAM, University City (2000)
9. Chow, E., Falgout, R.D., Hu, J.J., Tuminaro, R.S., Yang, U.M.: A survey of parallelization techniques for multigrid solvers. In: *Parallel Processing for Scientific Computing*, vol. 20, pp. 179–201 (2006)
10. Christen, M., Schenk, O., Burkhart, H.: Automatic code generation and tuning for stencil kernels on modern microarchitectures. In: *Proceedings of International Supercomputing Conference (ISC 2011)*, vol. 26, pp. 205–210 (2011)
11. Christen, M., Schenk, O., Burkhart, H.: PATUS: a code generation and auto-tuning framework for parallel iterative stencil computations on modern microarchitectures. In: *2011 IEEE International Conference on Parallel and Distributed Processing Symposium (IPDPS)*, pp. 676–687. IEEE (2011)
12. Christen, M., Schenk, O., Cui, Y.: PATUS for convenient high-performance stencils: evaluation in earthquake simulations. In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC 2012*, pp. 11:1–11:10. IEEE Computer Society Press, Los Alamitos (2012)
13. Chronopoulos, A.T., Swanson, C.D.: Parallel iterative s-step methods for unsymmetric linear systems. *Parallel Comput.* **22**(5), 623–641 (1996)
14. Chronopoulos, A.T., Gear, C.W.: s-Step iterative methods for symmetric linear systems. *J. Comput. Appl. Math.* **25**(2), 153–168 (1989)
15. Dubois, P.F., Greenbaum, A., Rodrigue, G.H.: Approximating the inverse of a matrix for use in iterative algorithms on vector processors. *Computing* **22**(3), 257–268 (1979)
16. Erlangga, Y.A., Nabben, R.: Multilevel projection-based nested Krylov iteration for boundary value problems. *SIAM J. Sci. Comput.* **30**(3), 1572–1595 (2008)
17. Feautrier, P., Lengauer, C.: The polyhedron model. In: *Encyclopedia of Parallel Computing*, pp. 1581–1592. Springer, Heidelberg (2011)
18. Fujita, K., Ichimura, T., Koyama, K., Inoue, H., Hori, M., Maddegedara, L.: Fast and scalable low-order implicit unstructured finite-element solver for earth’s crust deformation problem. In: *Proceedings of the Platform for Advanced Scientific Computing Conference, PASC 2017*, pp. 11:1–11:10. ACM, New York (2017)
19. Ghysels, P., Vanroose, W.: Hiding global synchronization latency in the preconditioned conjugate gradient algorithm. *Parallel Comput.* **40**(7), 224–238 (2014)
20. Ghysels, P., Ashby, T.J., Meerbergen, K., Vanroose, W.: Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM J. Sci. Comput.* **35**(1), C48–C71 (2013)
21. Grigori, L., Moufawad, S.: Communication avoiding ILU0 preconditioner. *SIAM J. Sci. Comput.* **37**(2), C217–C246 (2015)
22. Grosser, T., Größlinger, A., Lengauer, C.: Polly - performing polyhedral optimizations on a low-level intermediate representation. *Parallel Process. Lett.* **22**(4), 1250010 (2012)
23. Gysi, T., Grosser, T., Hoefler, T.: MODESTO: data-centric analytic optimization of complex stencil programs on heterogeneous architectures. In: *Proceedings of the 29th ACM on International Conference on Supercomputing, ICS 2015*, pp. 177–186. ACM, New York (2015)
24. King, J., Kirby, R.M.: A scalable, efficient scheme for evaluation of stencil computations over unstructured meshes. In: *2013 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–12, November 2013

25. Malas, T., Hager, G., Ltaief, H., Stengel, H., Wellein, G., Keyes, D.: Multicore-optimized wavefront diamond blocking for optimizing stencil updates. *SIAM J. Sci. Comput.* **37**(4), C439–C464 (2015)
26. Mohiyuddin, M., Hoemmen, M., Demmel, J., Yelick, K.: Minimizing communication in sparse matrix solvers. In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, p. 36. ACM (2009)
27. Stiefel, E., Hestenes, M.R.: Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* **49**(6) (1952)
28. Rupp, K.: CPU, GPU, and MIC hardware characteristics over time. <https://www.karlsruhp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/>
29. Rutishauser, H.: Theory of gradient methods. In: Engeli, M., Ginsburg, T., Rutishauser, H., Stiefel, E. (eds.) *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-adjoint Boundary Value Problems*, pp. 24–49. Springer, Heidelberg (1959). https://doi.org/10.1007/978-3-0348-7224-9_2
30. Saad, Y.: Krylov subspace methods on supercomputers. *SIAM J. Sci. Stat. Comput.* **10**(6), 1200–1232 (1989)
31. Tang, Y., Chowdhury, R.A., Kuszmaul, B.C., Luk, C.-K., Leiserson, C.E.: The Pochoir stencil compiler. In: *Proceedings of 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2011)*, pp. 117–128. ACM (2011)
32. Treibig, J., Hager, G., Wellein, G.: LIKWID: lightweight performance tools. In: Bischof, C., Hegering, H.G., Nagel, W., Wittum, G. (eds.) *Competence in High Performance Computing 2010*, pp. 165–175. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24025-6_14
33. U.S. Department of Energy, Office of Advanced Scientific Computing Research. Report on the workshop on Extreme-Scale Solvers: Transition to future Architectures, March 2012. <http://science.energy.gov/~media/ascr/pdf/program-documents/docs/reportExtremeScaleSolvers2012.pdf>. Accessed Mar 2013
34. Bondhugula, U., Bandishti, V., Pananilath, I.: Tiling stencil computations to maximize parallelism. In: *Proceedings of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2012)*, pp. 1–11 (2012)
35. Van der Vorst, H.A.: *Iterative Krylov Methods for Large Linear Systems*, vol. 13. Cambridge University Press, Cambridge (2003)



Applications of Trace Estimation Techniques

Shashanka Ubaru^(✉) and Yousef Saad

University of Minnesota at Twin Cities, Minneapolis, MN 55455, USA
{ubaru001,saad}@umn.edu

Abstract. We discuss various applications of trace estimation techniques for evaluating functions of the form $\text{tr}(f(A))$ where f is certain function. The first problem we consider that can be cast in this form is that of approximating the *Spectral density* or *Density of States* (DOS) of a matrix. The DOS is a probability density distribution that measures the likelihood of finding eigenvalues of the matrix at a given point on the real line, and it is an important function in solid state physics. We also present a few non-standard applications of spectral densities. Other trace estimation problems we discuss include estimating the trace of a matrix inverse $\text{tr}(A^{-1})$, the problem of counting eigenvalues and estimating the rank, and approximating the log-determinant (trace of log function). We also discuss a few similar computations that arise in machine learning applications. We review two computationally inexpensive methods to compute traces of matrix functions, namely, the Chebyshev expansion and the Lanczos Quadrature methods. A few numerical examples are presented to illustrate the performances of these methods in different applications.

1 Introduction

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric (real) matrix with an eigen-decomposition $A = U\Lambda U^T$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$, and λ_i , $i = 1, \dots, n$ are the eigenvalues of A and where the columns u_i , $i = 1, \dots, n$ of U are the associated eigenvectors. For the matrix function $f(A)$, defined as $f(A) = Uf(\Lambda)U^T$, where $f(\Lambda) = \text{diag}(f(\lambda_1), \dots, f(\lambda_n))$ [18], the trace estimation problems consist of computing an approximation of the trace of the matrix function $f(A)$, i.e.,

$$\text{tr}(f(A)) = \sum_{i=1}^n f(\lambda_i). \quad (1)$$

The problem of estimating the trace of a matrix function appears in a very broad range of applications that include machine learning, signal processing, scientific computing, statistics, computational biology and computational physics [2, 5, 12, 13, 17, 20, 22, 27, 29]. Clearly, a trace of a matrix function can be trivially computed from the eigen-decomposition of A . However, matrices in most of the applications just mentioned are typically very large and so computing the

complete eigen-decomposition can be expensive and sometimes even infeasible. Hence, the problem is to develop fast and scalable algorithms to perform such tasks without requiring the eigen-decomposition. This specific problem, which has been at the forefront of research in many distinct areas whether in physics or data-related applications, is the primary focus of this paper.

2 Trace Estimation Problems

We begin by first discussing a few trace estimation problems that arise in certain application areas.

2.1 Spectral Density

The first problem that we consider is that of computing the spectral density of a matrix [23], a very common problem in solid state physics. The spectral density of matrix, also known as Density of States (DOS) in physics, is a probability density distribution that measures the likelihood of finding eigenvalues of the matrix at a given point on the real line. Formally, the spectral density of a matrix is expressed as a sum of delta functions of the eigenvalues of the matrix. That is,

$$\phi(t) = \frac{1}{n} \sum_{i=1}^n \delta(t - \lambda_i),$$

where δ is the Dirac distribution or Dirac δ -function. This is not a proper function but a distribution and it is clearly not practically computable as it is defined. What is important is to compute a smoothed or approximate version of it that does not require computing eigenvalues, and several inexpensive methods have been proposed for this purpose, [23, 31, 35]. Recently, the DOS has been used in applications such as eigenvalue problem, e.g., for spectral slicing [38], for counting eigenvalues in intervals (‘eigencounts’) [11], and for estimating ranks [33, 34]. In Sect. 4, we present a few other (new) applications where the DOS can be exploited.

Article [23] reviews a set of inexpensive methods for computing the DOS. Here we briefly discuss two of these methods, namely the Kernel Polynomial method [35] and the Lanczos approximation method [23]. As was already mentioned the DOS is not a proper function. At best it can practically be viewed as a highly discontinuous function, which is difficult to handle numerically. The idea then is to replace the delta function by a surrogate Gaussian blurring function as is often done. A “blurred” version of the DOS is given by:

$$\phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^n h_\sigma(t - \lambda_i),$$

where $h_\sigma(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{t^2}{2\sigma^2}}$. The spectral density can then be approximated by estimating the trace of this blurred function, e.g., using the Lanczos algorithm. An older method consists of just expanding (formally) the DOS into Chebyshev polynomials. These two methods will be discussed later in Sect. 3.

2.2 Eigencount and Numerical Rank

The second trace estimation application we consider is that of counting eigenvalues located in a given interval (eigencount) and the related problem of estimating the numerical rank of a matrix. Estimating the number of eigenvalues $\eta_{[a,b]}$ located in a given interval $[a, b]$ (including the possible multiplicities) of a large sparse symmetric matrix is a key ingredient of effective eigensolvers [11], because these eigensolvers require an estimate of the dimension of the eigenspace to compute to allocate resources and tune the method under consideration. Estimating the numerical rank $r_\epsilon = \eta_{[\epsilon, \lambda_1]}$ is another closely related problem that occurs in machine learning and data analysis applications such as Principal Component Analysis (PCA), low rank approximations, and reduced rank regression [33, 34]. Both of these problems can be viewed from the angle of estimating the trace of a certain eigen-projector, i.e., the number of eigenvalues $\eta_{[a, b]}$ in $[a, b]$ satisfies:

$$\eta_{[a,b]} = \text{tr}(P), \text{ where } P = \sum_{\lambda_i \in [a, b]} u_i u_i^\top.$$

We can interpret P as a step function of A given by

$$P = h(A), \text{ where } h(t) = \begin{cases} 1 & \text{if } t \in [a, b] \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

The problem then is to find an estimate of the trace of $h(A)$. A few inexpensive methods are proposed in [11, 33, 34, 40] to approximately compute this trace. We can also compute the eigencount from the spectral density since

$$\eta_{[a, b]} = \int_a^b \sum_j \delta(t - \lambda_j) dt \equiv \int_a^b n\phi(t) dt. \quad (3)$$

2.3 Log-Determinant

Log-determinants have numerous applications in machine learning and related fields [17, 27, 29]. The logarithm of the determinant of a given positive definite matrix $A \in \mathbb{R}^{n \times n}$, is equal to the trace of the logarithm of the matrix, i.e.,

$$\log \det(A) = \text{tr}(\log(A)) = \sum_{i=1}^n \log(\lambda_i).$$

So, estimating the log-determinant of a matrix leads once more to the estimation of the trace of a matrix function in this case the logarithm function.

Various methods have been proposed for inexpensively computing logdeterminants. These methods differ in the approach used to approximate the log function. For example, the article [17] uses Chebyshev polynomial approximations of the log function, while [8, 39] uses Taylor series expansions. On the other hand, Aune et al. [3] developed a method based on rational approximations of

the logarithm. More recently, a method based on Lanczos Quadrature has been advocated for computing log determinants [32].

Log-Likelihood Estimation: One of the applications of log-determinants computation is in the in likelihood estimation problems that arise in Gaussian processes [27]. Maximum Likelihood Estimation (MLE) is a popular approach used for parameter estimation in high dimensional Gaussian models. The objective is to maximize the log-likelihood function with respect to a hyperparameter vector ξ :

$$\log p(z | \xi) = -\frac{1}{2}z^\top S(\xi)^{-1}z - \frac{1}{2}\log \det S(\xi) - \frac{n}{2}\log(2\pi), \quad (4)$$

where z is the data vector and $S(\xi)$ is the covariance matrix parameterized by ξ . As seen from the above expression, a log-determinant must be computed to obtain the log-likelihood, see [32].

2.4 Other Applications

Other frequent matrix function trace estimation problems include estimating the trace of a matrix inverse, the Schatten norms, and the Estrada index. These are discussed in turn.

Trace of a Matrix Inverse: The matrix inverse trace estimation problem amounts to computing the trace of the inverse function $f(t) = t^{-1}$ of a positive definite matrix $A \in \mathbb{R}^{n \times n}$, whose eigenvalues lie in the interval $[\lambda_{\min}, \lambda_{\max}]$ with $\lambda_{\min} > 0$. This problem appears in uncertainty quantification and in lattice quantum chromodynamics [20,37], where it is necessary to estimate the trace of the inverse of covariance matrices.

Schatten p -Norms: Given an input matrix $X \in \mathbb{R}^{d \times n}$, the Schatten p -norm of X is defined as

$$\|X\|_p = \left(\sum_{i=1}^r \sigma_i^p \right)^{1/p},$$

where the σ_i 's are the singular values of X and r its rank. The nuclear norm is the Schatten 1-norm so it is just the sum of the singular values. Estimating the nuclear norm and the Schatten p -norms of large matrices appears in matrix completion and in rank-constrained optimization problems, differential privacy and theoretical chemistry [25,32]. It is also used in SVD entropy computations [1] which has many applications [25]. Suppose we define a positive semidefinite matrix A as¹ $A = X^\top X$ or $A = XX^\top$. Then, the Schatten p -norm of X is defined as

$$\|X\|_p = \left(\sum_{i=1}^r \lambda_i^{p/2} \right)^{1/p} = \left(\text{tr}(A^{p/2}) \right)^{1/p}.$$

¹ The matrix product is not formed explicitly since the methods involved typically require only matrix vector products.

Hence, Schatten p -norms (the nuclear norm being a special case with $p = 1$) are the traces of matrix functions of A with $f(t) = t^{p/2}$, and they can be computed inexpensively using methods such as Lanczos Quadrature [32], Chebyshev expansions [16] and others [25].

Estrada Index: The Estrada index of graphs is a common tool in computational biology, and has applications that include protein indexing [12], statistical thermodynamics and information theory [9]. Estimating the Estrada index amounts to finding an approximation to the trace of the exponential function, i.e., we need to estimate $\text{tr}(\exp(A))$, where A is the adjacency matrix of the graph. Articles [16, 25, 32] discuss methods for a fast estimation of the Estrada index of graphs.

3 Methods

We now discuss two inexpensive techniques for estimating traces of matrix functions. The first approach is well-known among solid-state physicists. Termed ‘Kernel Polynomial Method’ (KPM) [34–36] or ‘Chebyshev expansion method’ [16, 17, 33], it consists simply of expanding the spectral density into Chebyshev polynomials. The second approach is based on the Lanczos algorithm [23, 32], where the relation between the Lanczos procedure and Gaussian quadrature formulas is exploited to construct a good approximation for the matrix function. We first discuss a standard tool known as the ‘stochastic trace estimator’, which is a key ingredient used in the methods to be discussed.

3.1 Stochastic Trace Estimator

The stochastic trace estimator [4, 19, 28] approximates the trace of a matrix function $f(A)$ by means of matrix-vector products with $f(A)$. This method takes a sequence of random vectors $v_l, l = 1, \dots, n_v$ whose components have zero mean $\mathbb{E}[v_l] = 0$ and whose 2-norm is one, $\|v_l\|_2 = 1$ and it then computes the average over the samples of $v_l^\top f(A) v_l$ to approximate the trace,

$$\text{tr}(f(A)) \approx \frac{n}{n_v} \sum_{l=1}^{n_v} v_l^\top f(A) v_l. \quad (5)$$

Convergence has been analyzed in [4, 28]. A variant of this method can be used to estimate the diagonal entries of $f(A)$, see [7]. Note that $f(A)$ need not be explicitly formed since we only need to efficiently compute the vectors $f(A)v_l$ for any v_l . This can be accomplished in a number of effective ways.

3.2 Chebyshev (Kernel) Polynomial Method

The Kernel Polynomial Method (KPM) proposed in [30, 35] computes an approximate DOS of a matrix using Chebyshev polynomial expansions, see, [23] for a

discussion. In KPM, the matrix is linearly transformed so as to map its eigenvalues from the initial interval $[\lambda_n, \lambda_1]$ into the interval $[-1, 1]$. This requires estimating the extreme eigenvalues, see [34]. KPM seek the expansion of:

$$\hat{\phi}(t) = \sqrt{1-t^2}\phi(t) = \sqrt{1-t^2} \cdot \frac{1}{n} \sum_{j=1}^n \delta(t - \lambda_j),$$

instead of the original $\phi(t)$ since the Chebyshev polynomials are orthogonal with respect to the weight function $(1-t^2)^{-1/2}$. Then, we write the partial expansion of $\hat{\phi}(t)$ as

$$\hat{\phi}(t) \approx \sum_{k=0}^m \mu_k T_k(t),$$

where $T_k(t)$ is the Chebyshev polynomial of degree k . A little calculation reveals that, formally at least, each expansion coefficient μ_k is given by $\mu_k = \frac{(2-\delta_{k0})}{\pi} \text{tr}(T_k(A))$. Here δ_{ij} is the Kronecker symbol, so $2 - \delta_{k0}$ is 1 when $k = 0$ and 2 otherwise. The trace of $T_k(A)$ can now be estimated with the help of the expansion coefficients the corresponding expansion coefficient μ_k are approximated as,

$$\mu_k \approx \frac{2 - \delta_{k0}}{\pi n_v} \sum_{l=1}^{n_v} (v_l)^\top T_k(A) v_l.$$

Scaling back by the weight function $(1-t^2)^{-1/2}$, we obtain the approximation for the spectral density function in terms of Chebyshev polynomial of degree m .

General Function $f(A)$: For a general function $f : [-1, 1] \rightarrow \mathbb{R}$, it is possible to obtain an approximation of the form

$$f(t) \approx \sum_{k=0}^m \gamma_k T_k(t).$$

using Chebyshev polynomial expansions or interpolations, see [24] for details. Here $T_k(t)$ is the Chebyshev polynomial of degree k and γ_k are corresponding coefficients specific to expanding the function $f(t)$. Hence, we can approximate the trace of $f(A)$ as

$$\text{tr}(f(A)) \approx \frac{n}{n_v} \sum_{l=1}^{n_v} \left[\sum_{k=0}^m \gamma_k (v_l)^T T_k(A) v_l \right], \quad (6)$$

using the Chebyshev expansions and the stochastic trace estimator (5). Articles [11, 33] discussed the expansion of step functions using Chebyshev polynomials to compute eigencounts and numerical ranks of matrices, respectively.

Han et al. [16] discussed the above Chebyshev expansion method to approximately estimate the traces $\text{tr}(f(A))$, when the function f is analytic over an

interval. They proposed using Chebyshev interpolations to obtain the coefficients γ_k . Problems such as estimating log-determinants, Estrada index, trace of matrix inverse and Schatten p norms were discussed. The functions $\log(t)$, $\exp(t)$, t^{-1} and $t^{p/2}$ are all analytic in the spectrum interval $[\lambda_{\min}, \lambda_{\max}]$, with $\lambda_{\min} > 0$.

When expanding discontinuous functions including the DOS and step functions using Chebyshev polynomials, oscillations known as *Gibbs Oscillations* appear near the discontinuities [11]. To reduce or suppress these oscillations, damping multipliers are often used, see [11, 23] for details. An important practical consideration is that we can economically compute vectors of the form $T_k(A)v$ using the three term recurrence of Chebyshev polynomials, see [16, 34]. The recent article [16] analyzed the convergence of methods for approximating $\text{tr}(f(A))$ with the Chebyshev method when the function $f(A)$ is analytic over the interval of interest.

3.3 Lanczos Quadrature

The Lanczos Quadrature method was developed in [14], and the idea of combining the stochastic trace estimator with the Lanczos Quadrature method appeared in [5, 6]. This method was recently analyzed and applied to matrix function trace estimation in [32]. In the Stochastic Lanczos Quadrature (SLQ) method, the scalar quantities $v^\top f(A)v$ in the trace estimator (5) are computed by treating them to Riemann-Stieltjes integral, and then using the Gauss quadrature rule to approximate this integral. Given the eigen-decomposition $A = Q\Lambda Q^\top$, we can write the scalar product as Riemann-Stieltjes integral given by,

$$v^\top f(A)v = v^\top Qf(\Lambda)Q^\top v = \sum_{i=1}^n f(\lambda_i)\mu_i^2 = \int_a^b f(t)d\mu(t), \quad (7)$$

where μ_i are the components of the vector $Q^\top v$ and the measure $\mu(t)$ is a piecewise constant function defined as

$$\mu(t) = \begin{cases} 0, & \text{if } t < a = \lambda_1, \\ \sum_{j=1}^{i-1} \mu_j^2, & \text{if } \lambda_{i-1} \leq t < \lambda_i, \quad i = 2, \dots, n, \\ \sum_{j=1}^n \mu_j^2, & \text{if } b = \lambda_n \leq t, \end{cases} \quad (8)$$

with λ_i ordered nondecreasingly. However, the complete eigen-decomposition of A will not be available, and will be very expensive to compute for large matrices. So, we consider estimating the integral using the Gauss quadrature rule [15]

$$\int_a^b f(t)d\mu(t) \approx \sum_{k=0}^m \omega_k f(\theta_k), \quad (9)$$

where $\{\omega_k\}$ are the weights and $\{\theta_k\}$ are the nodes of the $(m+1)$ -point Gauss quadrature. We can then compute these weights and nodes using the Lanczos algorithm [13].

Given symmetric matrix $A \in \mathbb{R}^{n \times n}$ and a starting vector w_0 of unit 2-norm, the Lanczos algorithm generates an orthonormal basis W_{m+1} for the *Krylov subspace* $\text{span}\{w_0, Aw_0, \dots, A^m w_0\}$ such that $W_{m+1}^\top A W_{m+1} = T_{m+1}$, where T_{m+1} is an $(m+1) \times (m+1)$ tridiagonal matrix. The columns w_k of W_{m+1} are related as

$$w_k = p_{k-1}(A)w_0, \quad k = 1, \dots, m,$$

where p_k are the Lanczos polynomials. The Lanczos polynomials are orthogonal with respect to the measure $\mu(t)$ in (8); see Theorem 4.2 in [13]. The nodes and the weights of the quadrature rule in (9) can be computed as the eigenvalues and the squares of the first entries of the eigenvectors of T_{m+1} . Thus, we have

$$v^\top f(A)v \approx \sum_{k=0}^m \tau_k^2 f(\theta_k) \quad \text{with} \quad \tau_k^2 = (e_1^\top y_k)^2, \quad (10)$$

where $(\theta_k, y_k), k = 0, 1, \dots, m$ are eigenpairs (eigenvalues and eigenvectors) of T_{m+1} by using v as the starting vector w_0 . Note that eigen-decomposition of T_{m+1} for small m is inexpensive to compute. Then, the trace of matrix function $f(A)$ can be computed as,

$$\text{tr}(f(A)) \approx \frac{n}{n_v} \sum_{l=1}^{n_v} \left(\sum_{k=0}^m (\tau_k^{(l)})^2 f(\theta_k^{(l)}) \right), \quad (11)$$

where $(\theta_k^{(l)}, \tau_k^{(l)}), k = 0, 1, \dots, m$ are eigenvalues and the first entries of the eigenvectors of the tridiagonal matrix $T_{m+1}^{(l)}$ corresponding to the starting vectors $v_l, l = 1, \dots, n_v$. A convergence analysis for this SLQ method was proposed in the recent article [32]. For analytic functions, it has been shown that the convergence of the Lanczos quadrature approximation is twice as fast as that of Chebyshev approximation methods. This stems from the fact that an m -point quadrature rule is exact for any polynomial of degree $2m$, see [32] for details.

Lanczos Approximation for the DOS. The Lanczos approximation technique for estimating spectral densities discussed in [23] is based on the above Lanczos Quadrature framework. Given a polynomial $p(t)$, we can use the Lanczos quadrature formula in Eq. (10), to compute the (Riemann-Stieltjes) integral $v^\top p(A)v$, see [13] for details. Since this is a Gaussian quadrature formula, it is exact when p is a polynomial of degree $\leq 2m + 1$.

As seen earlier, for an initial vector w_0 of the Lanczos sequence, expanded in the eigenbasis $\{u_i\}_{i=1}^n$ of A as $w_0 = \sum_{i=1}^n \beta_i u_i$ and we consider the discrete (Stieltjes) integral:

$$\int p(t) d\mu(t) = (p(A)w_0, w_0) = \sum_{i=1}^n \beta_i^2 p(\lambda_i). \quad (12)$$

This integral is a distribution ϕ_{w_0} applied to p , written as $(p(A)w_0, w_0) \equiv \langle \phi_{w_0}, p \rangle$. If we assume an idealistic situation where $\beta_i^2 = 1/n$ for all i , then ϕ_{w_0} will be exactly the distribution, the DOS function. In the sense of distributions,

$$\langle \phi_{w_0}, p \rangle \equiv (p(A)w_0, w_0) = \sum_{i=1}^n \beta_i^2 p(\lambda_i) = \sum_{i=1}^n \beta_i^2 \langle \delta_{\lambda_i}, p \rangle = \frac{1}{n} \sum_{i=1}^n \langle \delta_{\lambda_i}, p \rangle,$$

where δ_{λ_i} is a δ -function at λ_i . Then, from the Gaussian quadrature rule (10), we have: $\langle \phi_{w_0}, p \rangle \approx \sum_{k=1}^m \tau_k^2 p(\theta_k) = \sum_{k=1}^m \tau_k^2 \langle \delta_{\theta_k}, p \rangle$ and

$$\phi_{w_0} \approx \sum_{k=1}^m \tau_k^2 \delta_{\theta_k}.$$

Since the β_i 's are not equal in practice, we will need to use several starting vectors v_l and average the result of the above formula over them. This is the Lanczos approximation method for computing an approximate DOS [23].

If $(\theta_k^{(l)}, y_k^{(l)})$, $k = 1, 2, \dots, m$ are eigenpairs of the tridiagonal matrix T_m corresponding to the starting vector v_l , $l = 1, \dots, n_v$ and $\tau_k^{(l)}$ is the first entry of $y_k^{(l)}$, then the DOS function by Lanczos approximation is given by

$$\tilde{\phi}(t) = \frac{1}{n_v} \sum_{l=1}^{n_v} \left(\sum_{k=1}^m (\tau_k^{(l)})^2 \delta(t - \theta_k^{(l)}) \right). \quad (13)$$

The above function is a weighted spectral distribution of T_m , where τ_k^2 is the weight for the corresponding θ_k and it approximates the spectral density of A .

Computational Cost: The most expensive step in KPM is when computing the scalars $(v_l)^\top T_k(A) v_l$ for $l = 1, \dots, n_v$, $k = 0, \dots, m$. Hence, the computational cost for estimating matrix function traces by KPM will be $O(\text{nnz}(A) m n_v)$ for sparse matrices, where $\text{nnz}(A)$ is the number of nonzero entries of A . Similarly, the most expensive part of the Lanczos Quadrature procedure is to perform the m Lanczos steps with the different starting vectors. The computational cost for matrix function trace estimation by SLQ will be $O((\text{nnz}(A)m + nm^2)n_v)$ for sparse matrices where there is an assumed cost for reorthogonalizing the Lanczos vectors. As can be seen, these algorithms are inexpensive relative to methods that require matrix factorizations such as the QR or SVD.

4 Applications of the DOS

Among the applications of the DOS that were mentioned earlier, we will discuss two that are somewhat related. The first is a tool employed for estimating the rank of a matrix. This is only briefly sketched as it has been discussed in details earlier in [33, 34]. The second application, is in clustering and the problem of community detection in social graphs.

Threshold Selection for Rank Estimation: The *numerical rank* of a general matrix $X \in \mathbb{R}^{d \times n}$ is defined with respect to a positive tolerance ϵ as follows:

$$r_\epsilon = \min\{\text{rank}(B) : B \in \mathbb{R}^{d \times n}, \|X - B\|_2 \leq \epsilon\}. \quad (14)$$

To estimate r_ϵ we need to provide a good value for the threshold ϵ to be used to determine the rank. Recently, references [33,34] proposed a method for determining this threshold ϵ based on the plot of DOS. The idea is to detect a gap between the noisy and relevant eigenvalues (we are interested in the count of these relevant eigenvalues) by locating a local minima near zero in the DOS plot. The cutoff point is chosen to be where the derivative of the spectral density function becomes close to zero (local minimum) for the first time. Thus, the threshold ϵ can be selected as

$$\epsilon = \min\{t : \phi'(t) \geq \text{tol}, \lambda_n \leq t \leq \lambda_1\}, \quad (15)$$

for a small tolerance, e.g., $\text{tol} = -0.01$ and not zero for practical reasons.

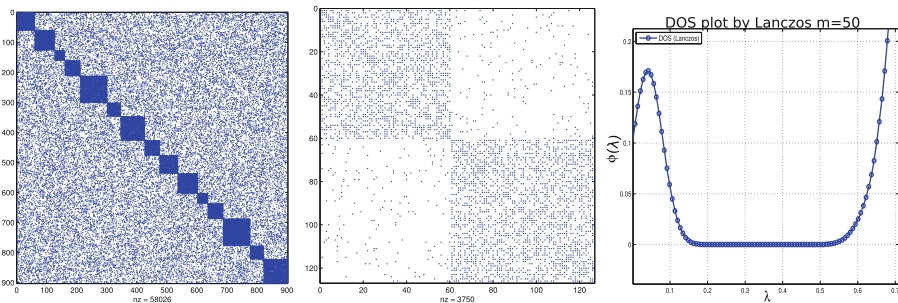


Fig. 1. Matrix of size $n = 902$ showing a block structure (left) and a zoom at the pattern focusing in the first 2 blocks (center). The DOS plot zooms in on the part between 0 and ≈ 0.7 (right).

Community Detection in Graphs: A problem of primary importance in various scientific and engineering disciplines is to determine dense subgraphs or *communities* within a given graph [21,26] Here we exploit the idea of spectral densities to determine the number of communities in a graph.

Let W denote the adjacency graph of a given (undirected) graph and let $L = D - W$ be the graph Laplacian with $D = \text{diag}(We)$ the diagonal matrix with diagonal entries as the sums of the entries of the corresponding row in W . In the ideal case of perfect clusters, where each cluster is a clique, the matrix L will have an eigenvalue of zero for each block, leading to a multiple zero eigenvalue, one for each block. By way of illustration consider left plot of Fig. 1. This represents the sparsity pattern of a small synthetic example made up of $k = 15$ sparse diagonal blocks that have a good density (about half dense) completed by some

sparse matrix that has a lower density. The sizes of the blocks vary from 105 to 190. The right side of the figure shows a zoom on the first two blocks to provide an illustration.

In an ideal scenario, each of the diagonal blocks is dense and there are no elements outside of these blocks. Then, the graph Laplacian will have exactly k zero eigenvalues where k is the number of blocks, which is 15 here. This is still true when the matrix is block diagonal but each diagonal block is a small sparse Laplacian, that contributes one zero eigenvalue. When we have certain off-block diagonal entries (equivalent to noise), the zero eigenvalues are perturbed, and we get a cluster of eigenvalues close to the origin. This is illustrated in the DOS plot corresponding to this matrix (zoomed in), see rightmost plot of Fig. 1. If we count the number of eigenvalues included in that cluster we will find that it is equal to the number of blocks. It is not too difficult to devise small subroutines that will consider the DOS curve and spot the point where the curve levels off after descending from its peak value to the left. A short matlab script we wrote finds the value $\tau = 0.136$ and the number of blocks detected with the help of spectral densities is close to the correct number of $k = 15$. Thus, this technique can be used to find an effective way to estimate the number of dense subgraphs.

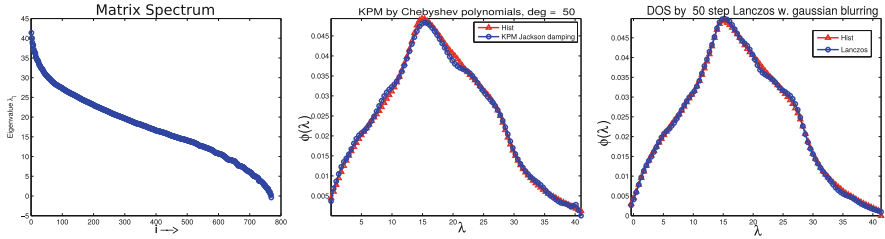


Fig. 2. The spectrum (Left), and the approximate DOS found by KPM (Middle), and by SLQ for the example S12.

Spectral densities can be used to extract a wealth of additional information. For example, the graphs may have multi-level clustering or sub-communities within the larger communities. Such situations are common in social networks, document and product categorization, etc. For such matrices, the multi-level clustering will result in a matrix with multiple clusters of eigenvalues in the DOS curve. It is possible to identify such clusters using the DOS and count the eigenvalues within these clusters, corresponding to the number of sub-communities.

5 Numerical Examples

In this section, we present a few examples to illustrate the performances of the Kernel polynomial and the Lanczos Quadrature methods for different problems of estimating traces of matrix functions.

Spectral Density. In the first experiment, to illustrate the performances of KPM and the Lanczos Approximation for approximating the DOS using a small example with matrix² `Si2` of size 769. Figure 2 plots the matrix spectrum (left), and the DOS plots obtained using KPM (Middle) and SLQ (right) with degree $m = 50$ and a number of samples $n_v = 30$, respectively. The red triangular lines in both plots represent the actual histogram (spectral density) of the matrix. The blue circle lines are the estimated DOS. Jackson damping [23] was used for KPM and Gaussian blurring with $\sigma = 0.25$ was used for Lanczos approximation plot. We note that the Lanczos algorithm gives slightly more accurate results for the same degree m compared to KPM. However, the Lanczos method is slightly more expensive due to the orthogonalization step.

Numerical Rank Estimation. The following experiment will illustrate the performances of the two techniques, KPM and SLQ, for estimating the numerical rank, see Fig. 3. We consider a 1961×1961 matrix named `netz4504` from the SuiteSparse collection (see footnote 2), see [10]. The matrix spectrum and the DOS plot obtained using KPM with degree $m = 50$ and a number of samples $n_v = 30$ are given in the left plots of Fig. 3. The threshold ϵ (the gap) estimated using this DOS plot was $\epsilon = 0.12$. The middle figure plots the estimated approximate ranks for different degrees m (top) and different number of starting vector n_v (bottom) using KPM (black solid line). Similarly, the right plots in Fig. 3 shows the estimated approximate ranks by the Lanczos approximation method using different degrees m (or the dimension of the Krylov subspace for the Lanczos method) and different number of sample vectors n_v . $n_v = 30$ in the top plot

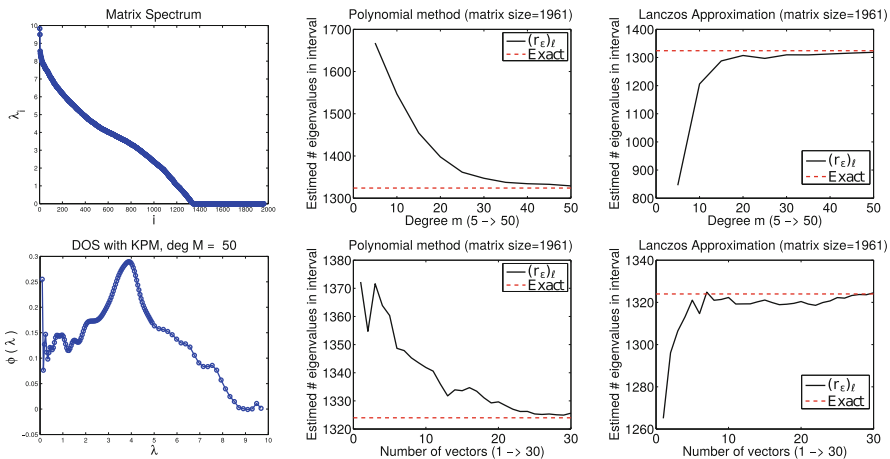


Fig. 3. The spectrum, the DOS found by KPM, and the approximate ranks estimation by KPM and by SLQ for the example `netz4504`.

² The matrices used in the experiments can be obtained from the SuiteSparse matrix collection: <https://sparse.tamu.edu/>.

and $m = 50$ in the bottom in both cases. The average approximate rank estimated over 30 sample vectors is equal to 1323.12 by KPM and by SLQ is equal to 1325.68. The exact number of eigenvalues in the interval is 1324, (indicated by the dash line in the plot). In this case ($m = 50, n_v = 30$), the number of matrix-vector multiplications required for both the rank estimator techniques is 1500.

6 Conclusion

The aim of this article, and the related presentation at HPCSE17, has been to provide a brief overview of the applications of traces of matrix functions and of effective related techniques for approximating them. In particular, we strived to highlight the broad applicability and the importance of spectral densities. These provide vital information in certain disciplines in physics and for this reason physicists were the first to develop effective methods such as KPM for computing them. On the other hand, the use of spectral densities, and more generally traces of matrix functions, in other areas such as machine learning and computational statistics, is more recent. There is no doubt that the interest in this topic will gain in importance given the rapid progress of disciplines that exploit large datasets.

Acknowledgments. This work was supported by NSF under grant CCF-1318597.

References

1. Alter, O., Brown, P.O., Botstein, D.: Singular value decomposition for genome-wide expression data processing and modeling. *Proc. Nat. Acad. Sci.* **97**(18), 10101–10106 (2000)
2. Andoni, A., Krauthgamer, R., Razenshteyn, I.: Sketching and embedding are equivalent for norms. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pp. 479–488. ACM (2015)
3. Aune, E., Simpson, D.P., Eidsvik, J.: Parameter estimation in high dimensional Gaussian distributions. *Stat. Comput.* **24**(2), 247–263 (2014)
4. Avron, H., Toledo, S.: Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM* **58**(2), 8 (2011)
5. Bai, Z., Fahey, G., Golub, G.: Some large-scale matrix computation problems. *J. Comput. Appl. Math.* **74**(1), 71–89 (1996)
6. Bai, Z., Golub, G.H.: Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices. *Ann. Numer. Math.* **4**, 29–38 (1996)
7. Bekas, C., Kokiopoulou, E., Saad, Y.: An estimator for the diagonal of a matrix. *Appl. Numer. Math.* **57**(11), 1214–1229 (2007)
8. Boutsidis, C., Drineas, P., Kambadur, P., Kontopoulou, E.-M., Zouzias, A.: A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra Appl.* **533**, 95–119 (2017)
9. Carbó-Dorca, R.: Smooth function topological structure descriptors based on graph-spectra. *J. Math. Chem.* **44**(2), 373–378 (2008)

10. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. *ACM Trans. Math. Softw. (TOMS)* **38**(1), 1 (2011)
11. Di Napoli, E., Polizzi, E., Saad, Y.: Efficient estimation of eigenvalue counts in an interval. ArXiv preprint [arXiv:1308.4275](https://arxiv.org/abs/1308.4275) (2013)
12. Estrada, E.: Characterization of 3D molecular structure. *Chem. Phys. Lett.* **319**(5), 713–718 (2000)
13. Golub, G.H., Meurant, G.: *Matrices, Moments and Quadrature with Applications*. Princeton University Press, Princeton (2009)
14. Golub, G.H., Strakoš, Z.: Estimates in quadratic formulas. *Numer. Algorithms* **8**(2), 241–268 (1994)
15. Golub, G.H., Welsch, J.H.: Calculation of gauss quadrature rules. *Math. Comput.* **23**(106), 221–230 (1969)
16. Han, I., Malioutov, D., Avron, H., Shin, J.: Approximating spectral sums of large-scale matrices using stochastic Chebyshev approximations. *SIAM J. Sci. Comput.* **39**(4), A1558–A1585 (2017)
17. Han, I., Malioutov, D., Shin, J.: Large-scale log-determinant computation through stochastic Chebyshev expansions. In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 908–917 (2015)
18. Higham, N.J.: *Functions of Matrices: Theory and Computation*. SIAM, University City (2008)
19. Hutchinson, M.F.: A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Commun. Stat.-Simul. Comput.* **19**(2), 433–450 (1990)
20. Kalantzis, V., Bekas, C., Curioni, A., Gallopoulos, E.: Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides. *Numer. Algorithms* **62**(4), 637–653 (2013)
21. Kaufman, L., Rousseeuw, P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, Hoboken (1990)
22. Li, Y., Nguyen, H.L., Woodruff, D.P.: On sketching matrix norms and the top singular vector. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1562–1581. SIAM (2014)
23. Lin, L., Saad, Y., Yang, C.: Approximating spectral densities of large matrices. *SIAM Rev.* **58**(1), 34–65 (2016)
24. Mason, J.C., Handscomb, D.C.: *Chebyshev Polynomials*. CRC Press, Boca Raton (2002)
25. Musco, C., Netrapalli, P., Sidford, A., Ubaru, S., Woodruff, D.P.: Spectrum approximation beyond fast matrix multiplication: algorithms and hardness. arXiv preprint [arXiv:1704.04163](https://arxiv.org/abs/1704.04163) (2017)
26. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**(3), 036104 (2006)
27. Rasmussen, C., Williams, C.: *Gaussian Processes for Machine Learning*. MIT Press, Cambridge (2006)
28. Roosta-Khorasani, F., Ascher, U.: Improved bounds on sample size for implicit matrix trace estimators. *Found. Comput. Math.* **15**(5), 1187–1212 (2015)
29. Rue, H., Held, L.: *Gaussian Markov Random Fields: Theory and Applications*. CRC Press, Boca Raton (2005)
30. Silver, R., Röder, H.: Densities of states of mega-dimensional Hamiltonian matrices. *Int. J. Mod. Phys. C* **5**(04), 735–753 (1994)
31. Turek, I.: A maximum-entropy approach to the density of states within the recursion method. *J. Phys. C: Solid State Phys.* **21**(17), 3251 (1988)

32. Ubaru, S., Chen, J., Saad, Y.: Fast estimation of $\text{tr}(f(A))$ via stochastic Lanczos quadrature. *SIAM J. Matrix Anal. Appl.* **38**, 1075–1099 (2017)
33. Ubaru, S., Saad, Y.: Fast methods for estimating the numerical rank of large matrices. In: *Proceedings of the 33rd International Conference on Machine Learning*, pp. 468–477 (2016)
34. Ubaru, S., Saad, Y., Seghouane, A.-K.: Fast estimation of approximate matrix ranks using spectral densities. *Neural Comput.* **29**(5), 1317–1351 (2017)
35. Wang, L.-W.: Calculating the density of states and optical-absorption spectra of large quantum systems by the plane-wave moments method. *Phys. Rev. B* **49**(15), 10154 (1994)
36. Weiße, A., Wellein, G., Alvermann, A., Fehske, H.: The kernel polynomial method. *Rev. Mod. Phys.* **78**(1), 275 (2006)
37. Wu, L., Laeuchli, J., Kalantzis, V., Stathopoulos, A., Gallopoulos, E.: Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *J. Comput. Phys.* **326**, 828–844 (2016)
38. Xi, Y., Li, R., Saad, Y.: Fast computation of spectral densities for generalized eigenvalue problems. arXiv preprint [arXiv:1706.06610](https://arxiv.org/abs/1706.06610) (2017)
39. Zhang, Y., Leithead, W.E.: Approximate implementation of the logarithm of the matrix determinant in Gaussian process regression. *J. Stat. Comput. Simul.* **77**(4), 329–348 (2007)
40. Zhang, Y., Wainwright, M.J., Jordan, M.I.: Distributed estimation of generalized matrix rank: efficient algorithms and lower bounds. In: *Proceedings of The 32nd International Conference on Machine Learning*, pp. 457–465 (2015)



Fourier Method for Approximating Eigenvalues of Indefinite Stekloff Operator

Yangqingxiang Wu and Ludmil Zikatanov^(✉)

Department of Mathematics, The Pennsylvania State University,
University Park, PA 16802, USA
ludmil@psu.edu
<http://personal.psu.edu/ltz1/>

Abstract. We introduce an efficient method for computing the Stekloff eigenvalues associated with the indefinite Helmholtz equation. In general, this eigenvalue problem requires solving the Helmholtz equation with Dirichlet and/or Neumann boundary condition repeatedly. We propose solving the discretized problem with Fast Fourier Transform (FFT) based on carefully designed extensions and restrictions operators. The proposed Fourier method, combined with proper eigensolver, results in an efficient and easy approach for computing the Stekloff eigenvalues.

Keywords: Stekloff eigenvalues · FFT · Helmholtz equation

1 Introduction

We consider the problem of computing the Stekloff eigenvalues corresponding to the indefinite Helmholtz equation. The efficient computation of such eigenvalues is needed in several numerical models. For example, in inverse scattering, as discussed in Cakoni et al. (2016), these eigenvalues carry information of the refractive index of an obstacle. To begin with, we introduce the following boundary value problem:

$$\mathcal{L}(\alpha, \lambda; \eta)w := \begin{cases} -\Delta w - \eta^2 w = f & \text{in } \Omega, \\ \alpha \frac{\partial w}{\partial n} + \lambda w = 0 & \text{on } \Gamma = \partial\Omega. \end{cases} \quad (1)$$

We call λ a Stekloff eigenvalue when the homogenous problem ($f = 0$ in (1)) has a non-trivial solution with $\alpha = 1$. For details on existence and properties of such eigenvalue problems we refer to Cakoni et al. (2016) (Sects. 2 and 4 for real and complex η respectively), Colton and Kress (2012).

As pointed out in Cakoni et al. (2016), the efficient computation of Stekloff eigenvalues is a challenging task. In addition, another important application of

This work was supported in part by NSF DMS-1720114.

the techniques that we propose here is of interest in computing transmission eigenvalues where the aim is to find the kernel of the difference of two indefinite Stekloff operators, see Cakoni and Kress (2017). The efficient solution of such problems, whether direct or inverse, requires fast solution of the Dirichlet problem, corresponding to $\mathcal{L}(0, 1; k(x))$, and the Neumann problem, corresponding to $\mathcal{L}(1, 0; k(x))$, as defined in (1). Here, $k(x)$ is the wave number and in case of non homogenous problem, the data $f(x)$ is the external force.

The difficulties associated with solving the indefinite Helmholtz equation numerically, especially in high frequency regimes, are well known (see Brandt and Livshits (1997), Ernst and Gander (2012)). Traditional iterative methods, such as Krylov subspace methods or standard MultiGrid (MG) and Domain Decomposition (DD) methods, are inefficient as discussed in Ernst and Gander (2012).

Over the last two decades, different preconditioners and solvers for the Helmholtz equation have been proposed. We refer to the classical works by Brandt and Livshits (1997), and Elman et al. (2001) for MG solvers and also to the more recent developments in Helmholtz preconditioning presented in Gander et al. (2015), Osei-Kuffuor and Saad (2010), Lahaye and Vuik (2017) and Sheikh et al. (2016). More recently, Engquist and Ying (2011) introduced the so called sweeping preconditioners which were further extended by Eslaminia and Guddati (2016) to double-sweeping preconditioners. Stolk (2013) proposed a DD preconditioner based on special transmission conditions between subdomains. Other DD methods are found in Chen and Xiang (2013), Zepeda-Núñez and Demanet (2016).

In our focus are the computations of Stekloff eigenvalues and the techniques which we propose here lead to efficient algorithms in many cases of practical interest and provide preconditioners for the Helmholtz problem. More importantly, our techniques easily extend to the Maxwell's system because they are based on the Fourier method.

The rest of the paper is organized as follows. We introduce the Fourier method for solving the constant coefficients boundary value problem in Sect. 2.1 (Dirichlet) and in Sect. 2.2 (Neumann). Further, in Sect. 3, we formulate the Stekloff eigenvalue problem and show how the FFT based Helmholtz solver can be applied. We conclude with several numerical tests on Stekloff eigenvalue computations as well as solution of the Helmholtz equation with variable wave number.

2 Periodic Extensions and Fourier Method

In this section, we focus on Fourier method for solving the Dirichlet problem and the Neumann problem with constant wave number $k(x) = k$. Non-constant case will also be considered in Sect. 4 as a numerical example.

2.1 Dirichlet Boundary Conditions

Dirichlet Problem in 1D. To explain the ideas, we consider the 1D version of (1) in the interval $(0, 1)$ with constant wave number k :

$$-u'' - k^2 u = f, \quad u(0) = u(1) = 0.$$

After discretization with central finite difference, we obtain the following linear system

$$A^D \mathbf{u} = \mathbf{f}, \quad A^D = T^D - k^2 h^2 I \in \mathbb{R}^{n \times n}, \quad (2)$$

where $T^D = \text{tridiag}(-1, 2, -1)$ is a tri-diagonal matrix, $\mathbf{u} = (u_1, \dots, u_n)^t$, $u_0 = u_{n+1} = 0$, $h = 1/(n+1)$, $\mathbf{f} = h^2(f_1, \dots, f_n)^t$, and $f_j = f(jh)$, $u_j \approx u(jh)$, $j = 1, \dots, n$. Here and in the following, the superscript t denotes transpose of a matrix or vector.

Let us now consider the same equation on a larger domain $(0, 2)$ and with periodic boundary conditions:

$$-v'' - k^2 v = g, \quad v(0) = v(2), \quad v'(0) = v'(2). \quad (3)$$

The finite difference discretization leads to a linear system for $\mathbf{v} = (v_1, \dots, v_N)^t$, $N = 2n + 2$, which is as follows:

$$A^P \mathbf{v} = \mathbf{g}, \quad A^P = T^P - k^2 h^2 I \in \mathbb{R}^{N \times N}. \quad (4)$$

Here, $e_1 = (1, 0, \dots, 0)^t$ and $e_N = (0, \dots, 0, 1)^t$ are the standard Euclidean basis vectors and $T^P = \text{tridiag}(-1, 2, -1) - e_1 e_N^t - e_N e_1^t$ is a circulant matrix. The right hand side $\mathbf{g} = h^2(g_1, \dots, g_N)^t$ is a given vector in \mathbb{R}^N depending on \mathbf{f} , which we specify later. The unknowns in this case are $v_j \approx v(2j/N)$, $j = 1, \dots, N$. Notice that from the periodic boundary conditions, we have $v_N \approx v(0) = v(2)$ and $v_1 \approx v(2/N)$ and this is reflected in the first and the last equation in the linear system (4).

The solution of systems with circulant matrices can be done efficiently using the Fast version (FFT) of the Discrete Fourier Transform (DFT) (see Cooley and Tukey (1965) for a description of FFT). The DFT is given by an operator $\mathcal{F} : \mathbb{C}^N \rightarrow \mathbb{C}^N$ represented by a matrix (denoted again with \mathcal{F}) defined as: $\mathcal{F}_{jm} = z^{(j-1)(m-1)}$, $z = e^{-\frac{2\pi i}{N}}$, with $j = 1, \dots, N$ and $m = 1, \dots, N$. As is well known, we have the DFT inversion formula:

$$\mathcal{F}^{-1} = \frac{1}{N} \mathcal{F}^* = \frac{1}{N} \overline{\mathcal{F}}.$$

Since A^P here is a circulant matrix it is diagonalized by \mathcal{F} , see Cooley and Tukey (1965), Golub and Van Loan (2012), and

$$\mathcal{F} A^P \mathcal{F}^{-1} = D^P = \text{diag}(d_l), \quad d_l = 4 \sin^2 \frac{\pi(l-1)}{N} - k^2 h^2, \quad l = 1, \dots, N. \quad (5)$$

As a consequence of this proposition, the solution \mathbf{v} to the problem (4) can be obtained by

$$\mathbf{v} = \mathcal{F}^{-1} (D^P)^{-1} \mathcal{F} \mathbf{g}. \quad (6)$$

Let us now consider the special case when \mathbf{g} in (4) corresponds to an “odd” extension of \mathbf{f} . That is $\mathbf{g} := E\mathbf{f}$ defined as

$$\mathbb{C}^N \ni \mathbf{g} = E\mathbf{f}, \quad g_j = \begin{cases} f_j, & j = 1, \dots, n, \\ 0, & j = n+1, \\ -f_{N-j}, & j = n+2, \dots, 2n+1, \\ 0, & j = N. \end{cases} \quad (7)$$

Here, $N = 2n + 2$. We have the following simple result.

Proposition 1. *If $N = 2n + 2$ and \mathbf{g} satisfies $g_j = -g_{N-j}$, for $j = n + 2, n + 3, \dots, 2n + 1$ and $g_{n+1} = g_{2n+2} = 0$, then the solution to (4) satisfies the relation:*

$$v_j = -v_{N-j}, \quad j = n + 2, \dots, 2n + 1. \quad (8)$$

Proof. We note that by assumption, \mathbf{g} is an “odd” function with respect to the middle of the interval $(0, 2)$. Since A^D is an invertible matrix, let \mathbf{u} satisfy $[A^D\mathbf{u}]_j = \mathbf{g}_j$, $j = 1, \dots, n$. Next, we define $\mathbf{v} = E\mathbf{u} \in \mathbb{C}^N$ where E is defined in (7) and it is immediate to verify that $A^P\mathbf{v} = \mathbf{g}$. Since A^P is also invertible, \mathbf{v} must be the unique solution of $A^P\mathbf{v} = \mathbf{g}$. From the definition of E , we conclude that \mathbf{v} satisfies (8).

Based on this observation, to solve the Dirichlet problem, we can define a linear operator B (which we will soon prove equals $(A^D)^{-1}$) as follows: we first define a restriction operator

$$\mathbb{C}^n \ni \mathbf{w} = R\mathbf{v}, \quad w_j = v_j, \quad j = 1, \dots, n, \quad \mathbf{v} \in \mathbb{R}^N. \quad (9)$$

We then set

$$B\mathbf{f} = R\mathcal{F}^{-1}(D^P)^{-1}\mathcal{F}E\mathbf{f}. \quad (10)$$

As the next proposition shows, B provides the exact solution to problem (2).

Proposition 2. *With B defined in (10) we have*

$$B = (A^D)^{-1}, \text{ and hence, } \mathbf{u} = B\mathbf{f}.$$

Proof. We notice that $R = (I_n, 0_{n, n+2}) \in \mathbb{R}^{n \times N}$, and $E = (I_n, 0_{n \times 1}, -\tilde{I}_n, 0_{n \times 1})^t \in \mathbb{R}^{N \times n}$, where I_n is the $n \times n$ identity matrix and $\tilde{I}_n = (\delta_{i, n+1-j})_{ij}$. Computing the product $A^D R (A^P)^{-1} E$ then shows that:

$$A^D R (A^P)^{-1} E = A^D (I, 0) (A^P)^{-1} E = (A^D, 0) (A^P)^{-1} E = (I, 0) E = I. \quad (11)$$

Indeed, the identities in (11) are verified by direct calculation:

$$\begin{aligned} & ((A^D, 0)(A^P)^{-1})_{ij} \\ &= -\frac{1}{N}(\mathcal{F}^{-1}(D^P)^{-1}\mathcal{F})_{i-1, j} + \frac{2}{N}(\mathcal{F}^{-1}(D^P)^{-1}\mathcal{F})_{i, j} - \frac{1}{N}(\mathcal{F}^{-1}(D^P)^{-1}\mathcal{F})_{i+1, j} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{N} \sum_{l=1}^N (-\bar{z}^{(i-2)(l-1)} + (2 - k^2 h^2) \bar{z}^{(i-1)(l-1)} - \bar{z}^{i(l-1)}) d_l z^{(k-1)(j-1)} \\
&= \frac{1}{N} \sum_{l=1}^N (2 - k^2 h^2 - 2 \cos(\frac{2\pi(l-1)}{N})) d_l z^{(k-1)(j-i)} \\
&= \frac{1}{N} \sum_{l=1}^N z^{(k-1)(j-i)} = \delta_{ij}. \tag{12}
\end{aligned}$$

This completes the proof.

Generalization to Higher Dimensions (Dirichlet Problem). We now consider the Helmholtz problem $\mathcal{L}(0, 1; k)$, defined in (1) in d -dimensions, i.e. we take $\Omega = (0, 1)^d$. With central finite difference discretization, we have the linear system:

$$A^D \mathbf{u} = \mathbf{f}, \quad A^D = \sum_{j=1}^d \left(I^{\otimes(j-1)} \otimes T^D \otimes I^{\otimes(d-j)} \right) - k^2 h^2 I^{\otimes d} \in \mathbb{R}^{n^d \times n^d}. \tag{13}$$

Here $M^{\otimes p} := \underbrace{M \otimes \dots \otimes M}_p$ for any matrix M and T^D as in (2) is the tri-diagonal matrix.

The extension and restriction operators in higher dimensions can be written as $E^{\otimes d}$ and $R^{\otimes d}$. The ‘‘odd’’ extension of \mathbf{f} is then $\mathbf{g} = E^{\otimes d} \mathbf{f}$. As in the 1D case, the linear system for the extended Helmholtz equation is:

$$A^P \mathbf{v} = \mathbf{g}, \quad A^P = \sum_{j=1}^d \left(I^{\otimes(j-1)} \otimes T^P \otimes I^{\otimes(d-j)} \right) - k^2 h^2 I^{\otimes d} \in \mathbb{R}^{N^d \times N^d}, \tag{14}$$

where T^P has been defined in (4). As in the one dimensional case (5), the matrix A^P is diagonalized by the multidimensional DFT $\mathcal{F}_d = \mathcal{F}^{\otimes d}$. The multidimensional version of (5) then is

$$\mathcal{F}_d A^P \mathcal{F}_d^{-1} = D_d^P := \sum_{j=1}^d \left(I^{\otimes(j-1)} \otimes D^P \otimes I^{\otimes(d-j)} \right) - k^2 h^2 I^{\otimes d}.$$

As a consequence, we obtain inversion formula similar to the one presented in Proposition 2. To show such representation, we need the following result.

Lemma 1. *Let E and R be extension and restriction operator defined in (7) and (9) respectively. Then the following identity holds*

$$A^D = R^{\otimes d} A^P E^{\otimes d}. \tag{15}$$

Proof. By using the standard properties of the tensor product, we have

$$\begin{aligned}
 R^{\otimes d} A^P E^{\otimes d} &= R^{\otimes d} \left(\sum_{j=1}^d \left(I^{\otimes(j-1)} \otimes T^P \otimes I^{\otimes(d-j)} \right) - k^2 h^2 I^{\otimes d} \right) E^{\otimes d} \\
 &= R^{\otimes d} \left(\sum_{j=1}^d \left(E^{\otimes(j-1)} \otimes T^P E \otimes E^{\otimes(d-j)} \right) - k^2 h^2 E^{\otimes d} \right) \\
 &= \sum_{j=1}^d \left(I^{\otimes(j-1)} \otimes R T^P E \otimes I^{\otimes(d-j)} \right) - k^2 h^2 I^{\otimes d}.
 \end{aligned}$$

It is straightforward to check that $R T^P E = T^D$. Thus, $R^{\otimes d} A^P E^{\otimes d} = A^D$.

The following theorem gives the representation of the inverse of the discretized Dirichlet problem in the multidimensional case.

Theorem 1. *The inverse of A^D can be written as*

$$(A^D)^{-1} = R^{\otimes d} (A^P)^{-1} E^{\otimes d}, \quad \text{and hence, } \mathbf{u} = R^{\otimes d} (A^P)^{-1} E^{\otimes d} \mathbf{f}. \quad (16)$$

Proof. Clearly, it is straight forward to see the following

$$A^D R^{\otimes d} (A^P)^{-1} E^{\otimes d} = R^{\otimes d} A^P E^{\otimes d} R^{\otimes d} (A^P)^{-1} E^{\otimes d} = I,$$

by the using properties of the matrix tensor product, Lemma 1 and identity $RE = I$.

Notice here, all the above results can be extended to rectangle (non-square) domain without too much difference. As a conclusion, we can solve the constant coefficient Helmholtz equation with Dirichlet boundary condition by FFT with complexity $O(n \log n)$, where n is the problem size.

2.2 Neumann Boundary Conditions

Neumann Problem in 1D. Let us consider the following 1D Helmholtz equation with one side Neumann boundary condition in $(0, 1)$:

$$-u'' - k^2 u = f, \quad u(0) = 0, \quad u'(0) = g. \quad (17)$$

After discretization with central finite difference, we have the linear system

$$A\mathbf{u} = \mathbf{f}$$

where

$$A = \text{tridiag}(-1, 2 - k^2 h^2, -1) - (1 - \frac{1}{2} k^2 h^2) e_{n+1} e_{n+1}^t \in \mathbb{R}^{(n+1) \times (n+1)}, \quad (18)$$

and $\mathbf{u} = (u_1, \dots, u_{n+1})^t$, $h = 1/(n+1)$, $u_j \approx u(jh)$, $f_j = f(jh)$, for $j = 1, \dots, n+1$, and $\mathbf{f} = (h^2 f_1, \dots, h^2 f_n, \frac{1}{2}h^2 f_{n+1} + hg)^t$.

With Fourier method for the Dirichlet problem in mind, we do “even” extension of the system (18) to get a Toeplitz system similar to (2):

$$A^e \mathbf{u}^e = \mathbf{f}^e, \quad A^e = \text{tridiag}(-1, 2 - k^2 h^2, -1) \in \mathbb{R}^{M \times M}, \quad (19)$$

where $M = 2n + 1$, $\mathbf{u}^e = (u_1, \dots, u_M)^t$, and $\mathbf{f}^e = (h^2 f_1, \dots, h^2 f_n, h^2 f_{n+1} + 2hg, h^2 f_n, \dots, h^2 f_1)^t$. By symmetry, the solution of system (19), when restricted on interval $(0, 1)$, will be the same as solution of system (18). Even though the problem size has been doubled, the extended system is Toeplitz, thus can be solved by Fourier method from Sect. 2.

In summary, we have the following inverse of the Neumann operator, that is the solution of:

$$A^{-1} \mathbf{f} = R_n \mathcal{F}^{-1} (D^P)^{-1} \mathcal{F} E_o E_e \mathbf{f}.$$

The operators involved in the definition above are (from right to left): even extension, odd extension followed by the inverse of the periodic problem and then restriction to Neumann problem. More precisely, for the even extension E_e we have,

$$\mathbb{C}^M \ni \mathbf{f}^e = [E_e \mathbf{f}]_j = \begin{cases} f_j, & j = 1, \dots, n, \\ 2f_{n+1}, & j = n + 1, \\ f_{M+1-j}, & j = n + 2, \dots, M. \end{cases} \quad (20)$$

Next, for the extension as to odd functions/vectors we have:

$$\mathbb{C}^{2M+2} \ni \mathbf{f}^o = [E_o \mathbf{f}^e]_j = \begin{cases} f_j^e, & j = 1, \dots, M, \\ 0, & j = M + 1, \\ -f_{N-j}^e, & j = M + 2, \dots, 2M + 1, \\ 0, & j = 2M + 2. \end{cases} \quad (21)$$

Finally, as in (5), we have the diagonal matrix $D^P = \text{diag}(d_l)$ with $d_l = 4 \sin^2 \frac{\pi(l-1)}{2M+2} - k^2 h^2$ for $l = 1, \dots, 2M + 2$, and the restriction R_n operator (to Neumann problem (18)):

$$\mathbf{v} \in \mathbb{R}^{2M+2}, \quad \mathbb{C}^{n+1} \ni \mathbf{u} = R_n \mathbf{v}, \quad u_j = v_j, \quad j = 1, \dots, n + 1.$$

Generalization to Higher Dimensions (Neumann problem). The Fourier method for Neumann problem can also be generalized to any dimension d in a fashion similar to the procedure given earlier for the Dirichlet problem. Just for illustration, we consider the following problem in 2D:

$$\begin{cases} -\Delta u - \eta^2 u = f & \text{in } \Omega = (0, 1)^2, \\ \frac{\partial u}{\partial x} = g & \text{on } \Gamma_1 = \{x = 1\} \times (0, 1), \\ u = 0 & \text{on } \partial\Omega/\Gamma_1. \end{cases} \quad (22)$$

We first do an “even” extension of the linear system and arrive at

$$A^e \mathbf{u}^e = \mathbf{F}^e, \quad A^e = I_M \otimes T_n^d + T_M^d \otimes I_n - k^2 h^2 I_{Mn} \in \mathbb{R}^{Mn \times Mn}, \quad (23)$$

where $T_j^d = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{j \times j}$ for any j , $\mathbf{u}^e = (\mathbf{u}_1, \dots, \mathbf{u}_M)^t$, and $\mathbf{F}^e = (h^2 \mathbf{f}_1, \dots, h^2 \mathbf{f}_n, h^2 \mathbf{f}_{n+1} + 2hg, h^2 \mathbf{f}_n, \dots, h^2 \mathbf{f}_1)^t$. Clearly, the restriction of \mathbf{u}^e on $\Omega \cup \Gamma_1$ is the same as the solution of the Neumann problem. Now, for the solution of (23) we can apply the method we have already described in Sect. 2.1.

3 Stekloff Eigenvalue Computation with Fourier Method

3.1 Variational Formulation

Multiplying the first equation in (1) with $\alpha = 1$ by $v \in H^1(\Omega)$ and integrating by parts, we get:

$$\int_{\Omega} \nabla w \nabla v - \eta^2 \int_{\Omega} wv = -\lambda \int_{\Gamma} wv.$$

Define $A(\eta) : H^1(\Omega) \rightarrow H^{-1}(\Omega)$ as

$$\langle A(\eta)w, v \rangle := (\nabla w, \nabla v)_{\Omega} - \eta^2 (w, v)_{\Omega} \quad \forall v \in H^1(\Omega), \quad (24)$$

where $(\cdot, \cdot)_{\Omega}$ is the L^2 inner product and $\langle \cdot, \cdot \rangle$ is duality pairing between $H^{-1}(\Omega)$ and $H^1(\Omega)$. The Stekloff operator, or Dirichlet-to-Neumann (DtN), $S(\eta)$ can be defined in two steps:

Firstly, for any $f \in H^{1/2}(\Gamma)$, define $f_0 \in H_0^1(\Omega)$ as the unique function satisfying:

$$\langle A(\eta)f_0, v_0 \rangle = -\langle A(\eta)(Ef), v_0 \rangle, \quad \forall v_0 \in H_0^1(\Omega),$$

where Ef is H^1 -bounded extension of f , e.g. harmonic extension.

Secondly, define the action of $S(\eta) : H^{1/2}(\Gamma) \rightarrow H^{-1/2}(\Gamma)$ as

$$\langle S(\eta)f, g \rangle_{1/2} = \langle A(\eta)(f_0 + Ef), Eg \rangle, \quad \forall g \in H^{1/2}(\Gamma), \quad (25)$$

where $\langle \cdot, \cdot \rangle_{1/2}$ is the duality pairing between $H^{1/2}(\Gamma)$ and $H^{-1/2}(\Gamma)$.

Lemma 2. *The Eq. (1) with $\alpha = 1$ has a non-trivial solution if and only if $-\lambda$ is an eigenvalue of the Stekloff operator $S(\eta)$.*

Proof. We prove one side of this equivalence here since the other half is similar. Suppose u is solution to (1) with $\alpha = 1$, then

$$\langle A(\eta)u, v_0 \rangle = 0, \quad \forall v_0 \in H_0^1(\Omega).$$

From the above equation, we have

$$\langle A(\eta)(u - Eu_{\Gamma}), v_0 \rangle = -\langle A(\eta)Eu_{\Gamma}, v_0 \rangle, \quad \forall v_0 \in H_0^1(\Omega),$$

where $u_{\Gamma} := u|_{\Gamma}$ is the trace of u on Γ . Denote $u_0 := u - Eu_{\Gamma}$. It is easy to see that $u_0 \in H_0^1(\Omega)$ and the following equation holds:

$$\langle S(\eta)u_{\Gamma}, g \rangle_{1/2} = \langle A(\eta)(u_0 + Eu_{\Gamma}), Eg \rangle = \langle A(\eta)u, Eg \rangle = -(\lambda u_{\Gamma}, g).$$

This shows that $-\lambda$ must be the eigenvalue of $S(\eta)$.

This lemma implies that solving problem (1) is equivalent to finding the eigenvalue for $S(\eta)$ with given η . In the next section we describe an efficient method for this task, namely the Fourier method for Stekloff eigenvalues.

3.2 Neumann-to-Dirichlet and Dirichlet-to-Neumann Operators

We start with the definition of Neumann-to-Dirichlet operator $T : L^2(\Gamma) \rightarrow L^2(\Gamma)$. Let $\mu \in L^2(\Gamma)$ and define $w_\mu \in H^1(\Omega)$ to be the solution of Neumann problem $\mathcal{L}(1, 0; \eta)w = (0, \mu)^t$. Equivalently, $w_\mu \in H^1(\Omega)$ satisfies

$$\langle A(\eta)w_\mu, v \rangle = \langle \mu, v \rangle_{1/2}, \quad \text{for any } v \in H^1(\Omega). \quad (26)$$

Taking the trace of w_μ , we can define Neumann-to-Dirichlet (NtD) operator T as $T\mu = w_\mu|_\Gamma$.

After discretization the Neumann problem $\mathcal{L}(1, 0; \eta)w = (0, \mu)^t$ with finite differences¹, we have the following linear system:

$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{w}_I \\ \mathbf{w}_B \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mu} \end{bmatrix}, \quad (27)$$

where $\boldsymbol{\mu}$ is discretized version of μ , \mathbf{w}_I and \mathbf{w}_B are approximations to solution of (26) restricted inside Ω and on boundary Γ respectively. Notice here we order the grids by their positions in the domain for illustration purpose. As we discussed in Sect. 2.2, Eq. (27) can be solved efficiently with Fourier method.

The discretized NtD operator T_h corresponding to the NtD operator T is

$$T_h \boldsymbol{\mu} = [\mathbf{0}, I] \begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mu} \end{bmatrix}. \quad (28)$$

To discretize the Dirichlet-to-Neumann operator, we consider the Dirichlet problem $\mathcal{L}(0, 1; \eta)w = (0, g)^t$. With analogous discretization, we have the following linear system:

$$\begin{bmatrix} A_{II} & A_{IB} \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} \mathbf{w}_I \\ \mathbf{w}_B \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{g} \end{bmatrix}. \quad (29)$$

Here \mathbf{g} is discretization of g . Clearly, $\mathbf{w}_B = \mathbf{g}$ as expected and the discrete version of (25) gives the action of S_h as

$$S_h \mathbf{g} = (A_{BB} - A_{BI}A_{II}^{-1}A_{IB})\mathbf{g}. \quad (30)$$

We have the following Lemma, which shows that T_h is the inverse of S_h .

Lemma 3. *For the operators T_h and S_h defined in (28) and (30), respectively, we have $S_h T_h = I$.*

¹ The discretization with piece-wise linear continuous Lagrange finite elements produces the same, albeit scaled, matrix. Thus, the consideration that follow apply to finite element discretizations as well.

Proof. We just need to show $S_h T_h \boldsymbol{\mu} = \boldsymbol{\mu}$ for any $\boldsymbol{\mu}$. This can be easily proved by Block-LU factorization.

$$\begin{aligned} S_h T_h \boldsymbol{\mu} &= S_h [\mathbf{0}, I] \begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mu} \end{bmatrix} \\ &= S_h [\mathbf{0}, I] \left(\begin{bmatrix} I & \mathbf{0} \\ A_{BI} A_{II}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{II} & A_{IB} \\ \mathbf{0} & S_h \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mu} \end{bmatrix} = \boldsymbol{\mu}. \end{aligned} \quad (31)$$

The results from the previous two sections show that we can efficiently compute the Stekloff eigenvalues of small magnitude as well as large magnitude. For example, the eigenvalues of small magnitude of S_h can be approximated by the reciprocal of the eigenvalues of NtD operator T_h . The action of T_h , which is needed repeatedly in such a procedure, can be efficiently computed by the Fourier method applied to the solution of the Helmholtz-Neumann problem as we have discussed earlier. For eigenvalues of the largest magnitude the same applies, except that we need the action of S_h , which requires fast solution of the corresponding Dirichlet problems, which we also described earlier.

4 Numerical Examples

In this section, we present two applications of the Fourier method described in Sect. 2. In the first example, the Fourier method is utilized in computing the Stekloff eigenvalues as discussed in Sect. 3. For the second example, given the fact that we can compute the numerical solution of constant coefficient Helmholtz equation efficiently, we apply such solver as preconditioner for varying efficient problem. Both examples are implemented in MATLAB and all tests have been performed on the same computer with dual-core 2.5 GHz CPU.

4.1 Computing Stekloff Eigenvalues

In this numerical example, we calculate the Stekloff eigenvalues corresponding to the problem:

$$\begin{cases} -\Delta w - \eta^2 w = 0 & \text{in } \Omega = (0, 1)^2, \\ \frac{\partial w}{\partial n} + \lambda w = 0 & \text{on } \Gamma_1 = \{x = 1\} \times (0, 1), \\ w = 0 & \text{on } \partial\Omega/\Gamma_1. \end{cases} \quad (32)$$

As is discussed in Sect. 3, the Stekloff eigenvalues can be numerically approximated by the eigenvalues of discretized DtN operator S_h in (30). To find the eigenvalues of S_h with small magnitude, for example, we need an efficient algorithm to compute the action of $T_h = S_h^{-1}$ on a vector. As in (28), one Neumann problem needs to be solved for computing each action. Thus, we create a function handle that solves the Neumann boundary problem in (28) with FFT, which is then used in MATLAB eigensolver `eigs`. For different η and grid size $n \times n$, the smallest six eigenvalues in magnitude are reported in Table 1.

Table 1. Smallest six Stekloff eigenvalues in magnitude for different η on $n \times n$ grid with $n = 50, 100$.

η	λ_1		λ_2		λ_3		λ_4		λ_5		λ_6	
	$n = 50$	$n = 100$	$n = 50$	$n = 100$	$n = 50$	$n = 100$	$n = 50$	$n = 100$	$n = 50$	$n = 100$	$n = 50$	$n = 100$
1	2.994	2.993	6.210	6.205	9.397	9.378	12.588	12.542	15.795	15.707	19.028	18.876
2	2.461	2.461	5.962	5.958	9.233	9.216	12.464	12.421	15.696	15.610	18.943	18.795
4	-3.159	-3.154	4.846	4.846	8.548	8.537	11.959	11.925	15.291	15.216	18.603	18.468
8	-1.245	-1.218	3.914	3.968	4.963	4.978	9.692	9.691	13.562	13.529	17.1817	17.098

4.2 Helmholtz Equation with Varying Wave Number

We consider the Helmholtz equation with homogeneous Dirichlet boundary condition and varying coefficient on domain $(0, 1)^2$, i.e.

$$\mathcal{L}(0, 1; k(x, y))u = (f(x, y), 0)^t.$$

The external force $f(x, y)$ is set to be constant 1. The wave number fields are $k_i(x, y) = \omega/c_i(x, y)$ with constant angular frequency ω and varying velocity fields $c_i(x, y)$ for $i = 1, 2$ as follows:

1. $c_1(x, y) = \frac{4}{3}[1 - 0.5 \exp(-0.5(x - 0.5)^2)]$,
2. $c_2(x, y) = \frac{4}{3}[1 - 0.5 \exp(-0.5(x - 0.5)^2 + (y - 0.5)^2)]$.

After discretizing the equation with central finite difference, we use GMRES as iterative solver for the linear system (relative residual tolerance 10^{-3}). The preconditioner is implemented by solving the constant wave number problem with k equaling the average value of number field over the entire domain.

To study how the preconditioning performance is depending on the magnitude of $k_i(x, y)$, we gradually increase ω and grid size n (in each dimension), while keeping the ratio ω/n fixed. In fact, this is the most computationally challenging case, since the relative percentage of positive eigenvalues for the linear system is fixed. We record the number of iteration N_i and CPU time T_i for the preconditioned GMRES to converge in Table 2. As a special case, for $\omega = 6.4\pi$, the wave number fields $k_i(x, y)$ and resulted wave fields $u_i(x, y)$ in each case have been shown in Fig. 1.

Table 2. Varying ω and n , GMRES iterations number and time.

ω	n^2	N_1	T_1	N_2	T_2
1.6π	50^2	4	0.068	4	0.062
3.2π	100^2	5	0.074	5	0.076
6.4π	200^2	6	0.12	6	0.12
12.8π	400^2	13	0.60	10	0.46

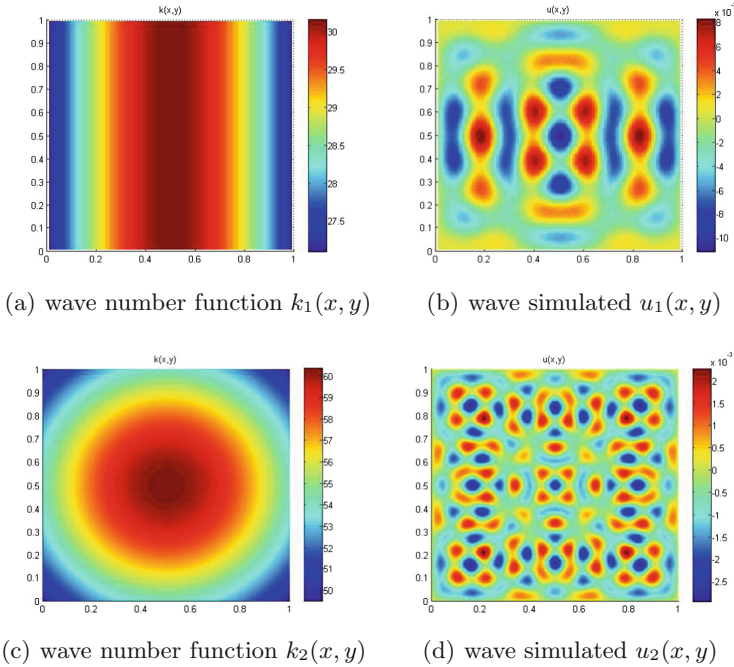


Fig. 1. Simulation results for different velocity fields.

5 Conclusions

In this paper, we proposed an efficient method for finding eigenvalues of indefinite Stekloff operators. The main tool that we developed is a fast Fourier method for solving constant coefficient Helmholtz equation with Dirichlet or Neumann boundary condition discretized on a uniform mesh. The resulting algorithm is efficient, transparent, and easy to implement. Our numerical experiments show that such algorithm works also as a solver for the Helmholtz problem with mildly varying coefficients (non-constant wave number). Another pool of important applications will be the computation of transmission eigenvalues, where our method has the potential to provide an efficient computational tool.

References

- Brandt, A., Livshits, I.: Wave-ray multigrid method for standing wave equations. *Electron. Trans. Numer. Anal.* **6**(162–181), 91 (1997)
- Cakoni, F., Colton, D., Meng, S., Monk, P.: Stekloff eigenvalues in inverse scattering. *SIAM J. Appl. Math.* **76**(4), 1737–1763 (2016). <https://doi.org/10.1137/16M1058704>. ISSN 0036–1399
- Cakoni, F., Kress, R.: A boundary integral equation method for the transmission Eigenvalue problem. *Appl. Anal.* **96**(1), 23–38 (2017). <https://doi.org/10.1080/00036811.2016.1189537>. ISSN 0003–6811

- Chen, Z., Xiang, X.: A source transfer domain decomposition method for Helmholtz equations in unbounded domain. *SIAM J. Numer. Anal.* **51**(4), 2331–2356 (2013)
- Colton, D., Kress, R.: *Inverse Acoustic and Electromagnetic Scattering Theory*, vol. 93. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-1-4614-4942-3>
- Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* **19**, 297–301 (1965). ISSN 0025–5718
- Elman, H.C., Ernst, O.G., O’leary, D.P.: A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations. *SIAM J. Sci. Comput.* **23**(4), 1291–1315 (2001)
- Engquist, B., Ying, L.: Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Model. Simul.* **9**(2), 686–710 (2011)
- Ernst, O.G., Gander, M.J.: Why it is difficult to solve Helmholtz problems with classical iterative methods. In: Graham, I., Hou, T., Lakkis, O., Scheichl, R. (eds.) *Numerical Analysis of Multiscale Problems*. LNCSE, vol. 83, pp. 325–363. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-22061-6_10
- Eslaminia, M., Guddati, M.N.: A double-sweeping preconditioner for the Helmholtz equation. *J. Comput. Phys.* **314**, 800–823 (2016)
- Gander, M.J., Graham, I.G., Spence, E.A.: Applying GMRES to the Helmholtz equation with shifted Laplacian preconditioning: what is the largest shift for which wavenumber-independent convergence is guaranteed? *Numer. Math.* **131**(3), 567–614 (2015). <https://doi.org/10.1007/s00211-015-0700-2>. ISSN 0029–599X
- Golub, G.H., Van Loan, C.F.: *Matrix Computations*, vol. 3. JHU Press, Baltimore (2012)
- Lahaye, D., Vuik, C.: How to choose the shift in the shifted Laplace preconditioner for the Helmholtz equation combined with deflation. In: Lahaye, D., Tang, J., Vuik, K. (eds.) *Modern Solvers for Helmholtz Problems*. GSMA, pp. 85–112. Birkhäuser, Cham (2017). https://doi.org/10.1007/978-3-319-28832-1_4
- Osei-Kuffuor, D., Saad, Y.: Preconditioning Helmholtz linear systems. *Appl. Numer. Math.* **60**(4), 420–431 (2010)
- Sheikh, A.H., Lahaye, D., Garcia Ramos, L., Nabben, R., Vuik, C.: Accelerating the shifted Laplace preconditioner for the Helmholtz equation by multilevel deflation. *J. Comput. Phys.* **322**, 473–490 (2016). <https://doi.org/10.1016/j.jcp.2016.06.025>. ISSN 0021–9991
- Stolk, C.C.: A rapidly converging domain decomposition method for the Helmholtz equation. *J. Comput. Phys.* **241**, 240–252 (2013)
- Zepeda-Núñez, L., Demanet, L.: The method of polarized traces for the 2D Helmholtz equation. *J. Comput. Phys.* **308**, 347–388 (2016)



Proportionality-Based Gradient Methods with Applications in Contact Mechanics

Zdeněk Dostál^{1,4}, Gerardo Toraldo², Marco Viola³, and Oldřich Vlach^{1,4}(✉)

¹ Department of Applied Mathematics,
Faculty of Electrical Engineering and Computer Science,
VŠB-Technical University of Ostrava, Ostrava, Czech Republic
oldrich.vlach@gmail.com

² Department of Mathematics and Applications, University of Naples Federico II,
Naples, Italy

³ Department of Computer Control and Management Engineering,
Sapienza University of Rome, Rome, Italy

⁴ IT4Innovations, National Supercomputing Center,
VŠB-Technical University of Ostrava, Ostrava, Czech Republic

Abstract. Two proportionality based gradient methods for the solution of large convex bound constrained quadratic programming problems, MPRGP (Modified Proportioning with Reduced Gradient Projections) and P2GP (Proportionality-based Two-phase Gradient Projection) are presented and applied to the solution of auxiliary problems in the inner loop of an augmented lagrangian algorithm called SMALBE (Semi-monotonic Augmented Lagrangian for Bound and Equality constraints). The SMALBE algorithm is used to generate the Lagrange multipliers for the equality constraints. The performance of the algorithms is tested on the solution of the discretized contact problems by means of TFETI (Total Finite Element Tearing and Interconnecting).

Keywords: QP optimization · Contact problems · P2GP · MPRGP

1 Introduction

This work is focused on the solution of quadratic programming problems characterized by bound constraints and linear equality constraints, i.e. problems of the form

$$\begin{aligned} \min f(\mathbf{x}) &:= \frac{1}{2}\mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b} \\ \text{s.t. } \mathbf{B}\mathbf{x} &= \mathbf{0} \\ \mathbf{x} &\in \Omega \end{aligned} \quad (1)$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes a symmetric positive definite matrix, $\mathbf{B} \in \mathbb{R}^{m \times n}$ is full rank and $m < n$, so $\text{Ker } \mathbf{B} \neq \{\mathbf{0}\}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. The set Ω is a closed convex set of the form

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : l_i \leq x_i \leq u_i, \quad i = 1, \dots, n\}.$$

We admit $l_i = -\infty$ and/or $u_i = \infty$, so that Ω_i can be defined by box or bound constraints.

A number of methods have been proposed for the solution of (1), basically based either on interior point (see for instance [5, 6] and references therein) or active set strategies [15]. In this paper we will just consider algorithms which belong to the latter class. More specifically, we are interested in SMALBE, a variant of Augmented Lagrangian methods which generates approximations for the Lagrange multipliers for the equality constraints in the outer loop and proportionality-based gradient methods for the bound constrained minimization of the lagrangian in the inner loop. A unique feature of SMALBE is a bound on the number of outer iterations which is independent on the conditioning of the constraints. Two algorithms are used for the solution of the auxiliary inner loop problems. The first one is the MPRGP algorithm (Modified Proportioning with Reduced Gradient Projections). A nice feature of MPRGP, which combines conjugate gradient method with gradient projections, is the bound on the rate of convergence in terms of the condition number of the Hessian, so that it can solve a class of problems (1) with the spectrum of \mathbf{A} in a given positive interval in a number of iterations which is independent of n [21]. In particular, such property is enjoyed by a class of problems (1) obtained by various discretizations of variational inequalities that describe the equilibrium of a system of bodies in mutual contact, so that the cost of a solution is asymptotically proportional to n (see Dostál et al. [18, 20], or [19, Chap. 11]). The second algorithm is P2GP (Proportionality-based Two-phase Gradient Projection), which combines monotonic spectral method with gradient projections and turned out to be very efficient for the solution of several problems [9]. Here we briefly describe the above algorithms and compare their performance on the solution of discretized contact problems of elasticity.

2 The SMALBE Framework

The SMALBE algorithm [12] works with the augmented Lagrangian in the form

$$L(\mathbf{x}, \boldsymbol{\lambda}, \varrho) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b} + \mathbf{x}^\top \mathbf{B}^\top \boldsymbol{\lambda} + \frac{\varrho}{2} \|\mathbf{B}\mathbf{x}\|^2,$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of the lagrangian multipliers associated with the linear equality constraints and $\varrho > 0$ is a fixed regularization parameter. In the inner loop, the algorithm carries out an approximate bound constrained minimization of the augmented lagrangian with the precision controlled by a multiple of the feasibility error and the projected gradient

$$g_i^P(\mathbf{x}) \equiv g_i^P(\mathbf{x}, \boldsymbol{\lambda}, \varrho) := \begin{cases} \nabla L_{x_i}(\mathbf{x}, \boldsymbol{\lambda}, \varrho) & \text{if } i \in \mathcal{F}(\mathbf{x}), \\ \min\{0, \nabla L_{x_i}(\mathbf{x}, \boldsymbol{\lambda}, \varrho)\} & \text{if } i \in \mathcal{A}_l(\mathbf{x}), \\ \max\{0, \nabla L_{x_i}(\mathbf{x}, \boldsymbol{\lambda}, \varrho)\} & \text{if } i \in \mathcal{A}_u(\mathbf{x}), \end{cases}$$

where

$$\mathcal{F}(\mathbf{x}) = \{i : x_i \in (\ell_i, u_i)\}, \quad \mathcal{A}_l(\mathbf{x}) = \{i : x_i = \ell_i\}, \quad \mathcal{A}_u(\mathbf{x}) = \{i : x_i = u_i\}.$$

The minimization in the inner loop can be carried out by any convergent algorithm. Here we use the gradient methods described in the next section. The SMALBE algorithm reads as follows.

Algorithm 1. (SMALBE)

```

1:  $tol \geq 0; \eta > 0; 1 > \vartheta > 0; \varrho > 0; M_0 = M_1 \in \mathbb{R}; \boldsymbol{\lambda}_0 \in \mathbb{R}^m; \mathbf{x}^0 \in \mathbb{R}^n; k = 0;$ 
2: while ( $\|\mathbf{g}^P(\mathbf{x}^k, \boldsymbol{\lambda}^k, \varrho)\| \leq tol$  and  $\|\mathbf{B}\mathbf{x}^k\| \leq tol$ ) do ▷ MAIN LOOP
3:   Find  $\mathbf{x}^{k+1} \in \Omega$  such that
4:    $\|\mathbf{g}^P(\mathbf{x}^{k+1}, \boldsymbol{\lambda}^k, \varrho)\| \leq \min\{M_{k+1}\|\mathbf{B}\mathbf{x}^{k+1}\|, \eta\}$  ▷ BQP SUBPROBLEM
5:    $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \varrho\mathbf{B}\mathbf{x}^{k+1}$ 
6:   if ( $M_{k+1} = M_k$  and  $L(\mathbf{x}^{k+1}, \boldsymbol{\lambda}^{k+1}, \varrho) < L(\mathbf{x}^k, \boldsymbol{\lambda}^k, \varrho) + \frac{\varrho}{2}\|\mathbf{B}\mathbf{x}^{k+1}\|$ ) then
7:      $M_{k+2} = \vartheta M_{k+1}$  ▷ TIGHTEN PRECISION CONTROL
8:   else
9:      $M_{k+2} = M_{k+1}$ 
10:  end if
11:   $k = k + 1$ 
12: end while
    
```

3 Gradient Methods Based on Proportionality Measures

In the previous section we saw that solving problem (1) by means of the SMALBE framework leads to the solution of a series of quadratic programs subject to bound constraints only (BQPs), i.e. problems of the form

$$\begin{aligned} & \min \hat{f}(\mathbf{x}) \\ & \text{s.t. } l_i \leq x_i \leq u_i, \quad i = 1, \dots, n, \end{aligned}$$

where $\hat{f}(\mathbf{x}) = L(\mathbf{x}, \boldsymbol{\lambda}, \varrho)$.

The minimization of the Lagrangian in Step 4 does not affect $\boldsymbol{\lambda}, \varrho$, so we shall consider in this section cost functions with only one variable.

Many gradient-based methods have been proposed which alternate steps aimed at identifying the variables which are on the bound at the solution (identification) and steps aimed at reducing the objective function on the face determined by the current active set. The latter, in particular, are usually unconstrained minimization steps for the solution of an *auxiliary problem* of the form

$$\begin{aligned} & \min \hat{f}(\mathbf{x} + \mathbf{d}) \\ & \text{s.t. } d_i = 0, \quad i \in \mathcal{A}(\mathbf{x}), \end{aligned} \tag{2}$$

where we have defined the active set as $\mathcal{A}(\mathbf{x}) := \mathcal{A}_l(\mathbf{x}) \cup \mathcal{A}_u(\mathbf{x})$. If the minimizer of (2) is unfeasible, a projected line-search is usually used to recover a feasible point on the face under analysis.

A crucial issue in the development of such kind of algorithms is to define suitable criteria to decide how much a face is worth to be explored. Indeed

finding an approximate solution for the auxiliary problem with an high level of accuracy may be a computational expensive task, which ends up to be useless if the current active set is far from being the optimal one. To overcome the inefficiencies associated with the use of heuristic criteria such as imposing a maximum number of consecutive minimization steps, some practical termination condition have been proposed in literature. An effective one is the one based on the concept of *proportionality* (see, e.g., Dostál [11], Dostál and Schöberl [21], Friedlander and Martínez [24], or Bielchowsky et al. [3]). An iterate \mathbf{x}^k is called *proportional* if, for a suitable constant $\Gamma > 0$,

$$\|\boldsymbol{\beta}(\mathbf{x}^k)\|^2 \leq \Gamma \tilde{\varphi}^\top(\mathbf{x}^k) \boldsymbol{\varphi}(\mathbf{x}^k), \quad (3)$$

where $\boldsymbol{\varphi}(\mathbf{x})$, $\tilde{\varphi}^\top(\mathbf{x})$, and $\boldsymbol{\beta}(\mathbf{x})$ are the so-called free, reduced free, and chopped gradients, respectively, defined component-wise as

$$\varphi_i := \begin{cases} \nabla L_{\mathbf{x}_i} & \text{if } i \in \mathcal{F}, \\ 0 & \text{if } i \in \mathcal{A}_l, \\ 0 & \text{if } i \in \mathcal{A}_u, \end{cases} \quad \beta_i := \begin{cases} 0 & \text{if } i \in \mathcal{F}, \\ \min\{0, \nabla L_{\mathbf{x}_i}\} & \text{if } i \in \mathcal{A}_l, \\ \max\{0, \nabla L_{\mathbf{x}_i}\} & \text{if } i \in \mathcal{A}_u, \end{cases}$$

$$\tilde{\varphi}_i = \begin{cases} \max\{\varphi_i, \frac{u_i - x_i}{\alpha}\} & \text{if } \varphi_i < 0, \\ \min\{\varphi_i, \frac{u_i - x_i}{\alpha}\} & \text{if } \varphi_i > 0, \end{cases}$$

where for ease of notation the dependence from \mathbf{x} or $(\mathbf{x}, \boldsymbol{\lambda}, \rho)$ has been omitted. We note that \mathbf{x}^* is stationary for the BQP problem if and only if

$$\|\boldsymbol{\beta}(\mathbf{x}^*)\| + \|\boldsymbol{\varphi}(\mathbf{x}^*)\| = 0,$$

moreover it is easy to show that, for each $\mathbf{x} \in \Omega$, $\boldsymbol{\beta}(\mathbf{x}) + \boldsymbol{\varphi}(\mathbf{x}) = \mathbf{g}^P(\mathbf{x})$. The vector $\boldsymbol{\varphi}$ coincides with the restriction of the gradient to the reduced space (the one containing the current face), therefore it provides a measure of the optimality of the current point with respect to the current face. The vector $\boldsymbol{\beta}$ instead provides a measure of the optimality over the complementarity space. The ratio between the two is in fact the ratio of the norms of the violation of the Karush-Kuhn-Tucker conditions at free and active variables. It has been proved that if the Hessian of the objective function is positive definite, disproportionality of \mathbf{x}^k guarantees that the solution of the BQP problem does not belong to the face determined by the active variables at \mathbf{x}^k , and thus exploration of that face is stopped [11]. The idea of proportional iteration has been recently extended to a more general class of linearly constrained problems, i.e. those subject to bound constraints and a single linear constraint [9].

In the following we will briefly introduce two examples of algorithm proposed in literature which can be ascribed to the class of proportionality and gradient-based active-set algorithms, namely the MPRGP [21] and the P2GP [9].

3.1 The MPRGP Algorithm

The MPRGP algorithm is an active set based algorithm which explores the current face by the conjugate gradient method as long as feasible steps are generated

and the violation of the KKT conditions on the free set dominates that on the active set, i.e., as long as (3) holds true. If the iteration is not proportional, then the algorithm releases some indices from the active set using the decrease direction $-\beta$, and if the iteration is not feasible, the algorithm proceeds along the conjugate direction as long as possible (feasible halfstep) and then carries out the reduced gradient step with a fixed steplength. Notice that in case of bound constraints, it is possible to take an optimal unconstrained steplength in order to release the active constraints in the direction $-\beta$. The steplength does not change during the computation and its length is based on the analysis of the decrease of the cost function along the gradient path (see, e.g., [14, 16, 21]).

Algorithm 2. (MPRGP)

```

1:  $\mathbf{x}^0 \in \Omega$ ;  $tol > 0$ ;  $\alpha \in (0, 2\|\mathbf{A}\|^{-1})$ ;  $\Gamma > 0$ ;  $k = 0$ ;  $\mathbf{g} = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$ ;  $\mathbf{p} = \varphi(\mathbf{x}^0)$ 
2: while ( $\|\mathbf{g}^P(\mathbf{x}^k)\| > tol$ ) do ▷ MAIN LOOP
3:   if ( $\|\beta(\mathbf{x}^k)\|^2 \leq \Gamma \tilde{\varphi}^\top(\mathbf{x}^k)\varphi(\mathbf{x}^k)$ ) then ▷ PROPORTIONAL  $\mathbf{x}^k$ 
4:      $\alpha_{cg} = \mathbf{g}^\top \mathbf{p} / \mathbf{p}^\top \mathbf{A} \mathbf{p}$ ,  $\mathbf{y} = \mathbf{x}^k - \alpha_{cg} \mathbf{p}$  ▷ TRIAL CG STEP
5:      $\alpha_f = \max\{\alpha \mid \mathbf{x}^k - \alpha \mathbf{p} \in \Omega\}$ 
6:     if ( $\alpha_{cg} \leq \alpha_f$ ) then
7:        $\mathbf{x}^{k+1} = \mathbf{y}$ ;  $\mathbf{g} = \mathbf{g} - \alpha_{cg} \mathbf{A} \mathbf{g}$  ▷ CG STEP
8:        $\beta = \varphi^\top(\mathbf{y}) \mathbf{A} \mathbf{p} / \mathbf{p}^\top \mathbf{A} \mathbf{p}$ ;  $\mathbf{p} = \varphi(\mathbf{y}) - \beta \mathbf{p}$ 
9:     else ▷ EXPANSION OF ACTIVE SET
10:       $\mathbf{x}^{k+\frac{1}{2}} = \mathbf{x}^k - \alpha_f \mathbf{p}$ ;  $\mathbf{g} = \mathbf{g} - \alpha_f \mathbf{A} \mathbf{p}$  ▷ FEASIBLE HALFSTEP
11:       $\mathbf{x}^{k+1} = P_\Omega(\mathbf{x}^{k+\frac{1}{2}} - \alpha \varphi(\mathbf{x}^{k+\frac{1}{2}}))$  ▷ FIXED STEPLENGTH EXPANSION STEP
12:       $\mathbf{g} = \mathbf{A} \mathbf{x}^{k+1} - \mathbf{b}$ ;  $\mathbf{p} = \varphi(\mathbf{x}^{k+1})$ 
13:    end if
14:  else
15:     $\mathbf{d} = \beta(\mathbf{x}^k)$ ;  $\alpha_{cg} = \mathbf{g}^\top \mathbf{d} / \mathbf{d}^\top \mathbf{A} \mathbf{d}$  ▷ PROPORTIONING STEP
16:     $\alpha_{fcg} = \min\{\max\{\alpha \mid \mathbf{x}^k - \alpha \mathbf{d} \in \Omega\}, \alpha_{cg}\}$ ,
17:     $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_{fcg} \mathbf{d}$ ;  $\mathbf{g} = \mathbf{g} - \alpha_{fcg} \mathbf{A} \mathbf{d}$ ;  $\mathbf{p} = \varphi(\mathbf{x}^{k+1})$ 
18:  end if
19:   $k = k + 1$ 
20: end while
    
```

The expansion is implemented by means of Euclidean projection P_Ω , i.e.,

$$P_\Omega(\mathbf{x}) = \arg \min_{\mathbf{y} \in \Omega} \|\mathbf{y} - \mathbf{x}\|.$$

3.2 The P2GP Algorithm

Gradient projection methods have been widely used for the solution of problems characterized by constraints for which the projection onto the feasible set can be computed cheaply, as in the case of bound constraints. Apart from being easy to implement, gradient projection methods have been shown to have nice identification properties [7], which together with their capability of adding/removing

multiple variables to/from the active set in a single iteration, made them a natural choice for the identification of the active constraints in active-set methods. A well-known method based on this approach is the GPCG [25] by Moré and Toraldo, developed for strictly convex quadratic programs subject to bound on the variables. The acronym GPCG stands for “Gradient Projection - Conjugate Gradient”, the algorithm indeed alternates between two phases: an identification phase, which performs GP iterations until a suitable face of the feasible set is identified or no reasonable progress toward the solution is achieved, and a minimization phase, which exploits the Conjugate Gradient (CG) method to find an approximate minimizer of the objective function in the reduced space resulting from the previous phase. The global convergence of the GPCG method relies on the global convergence of GP with steplengths satisfying a suitable sufficient decrease conditions [7]. In their recent work, di Serafino et al. [9] developed a new active-set method based on gradient projection, called the P2GP (Proportionality-based Two-phase Gradient Projection). The P2GP can be seen not only as an extension of the original GPCG framework to the solution of a wider class of linearly constrained problems, but also an improvement of the original two-phase algorithm with the introduction of spectral steplengths proposed in literature for gradient methods [8], and the replacement of the original heuristic used for the switch between minimization and identification, with a deterministic rule based on the concept of proportional iterate. This last improvement, from the theoretical point of view, allowed to prove finite termination of the algorithm for strictly convex problems, even in the case of dual degeneracy of the solution. Anyway, apart from the improved theoretical properties, the introduction of the proportionality-based stopping criterion for the termination of the minimization phase, together with the use of Barzilai Borwein-like steplengths [2] of Frassoldati [23] for the gradient projection, which have shown to be effective in several application areas [1, 4, 10], led to better computational performances with respect to the original GPCG algorithm in the solution of BQPs.

The algorithm for the strictly convex case is sketched in Algorithm 3.

P2GP alternates identification phases, where GP steps satisfying sufficient decrease conditions are performed, and minimization phases, where an approximate solution to the auxiliary problem is searched. Unless a point satisfying the stopping condition is found, the identification phase proceeds either until a promising active set \mathcal{A}^{k+1} is identified (i.e., an active set that remains fixed in two consecutive iterations) or no reasonable progress is made in reducing the objective function, i.e.,

$$\hat{f}^k - \hat{f}^{k+1} \leq \eta \max_{\bar{k} \leq l < k} (\hat{f}^l - \hat{f}^{l+1}), \quad (4)$$

where $\eta \in (0, 1)$ is a suitable constant and \bar{k} is the first iteration of the current identification phase. In the minimization phase, an approximate solution to problem (2) is searched for. In detail, the minimization of the auxiliary unconstrained problem is abandoned if an approximation to a stationary point is computed satisfying a criterion similar to (4). The proportionality criterion (3) is used to decide when the minimization phase has to be terminated.

Algorithm 3. (P2GP)

```

1:  $\mathbf{x}^0 \in \Omega$ ;  $tol \geq 0$ ;  $\eta \in (0, 1)$ ;  $\Gamma > 0$ ;  $k = 0$ ;
2:  $conv = (\|\mathbf{g}^P(\mathbf{x}^k)\| \leq tol)$ ;  $phase1 = .true.$ ;  $phase2 = .true.$ 
3: while ( $\sim conv$ ) do ▷ MAIN LOOP
4:    $m = k$ ;
5:   while ( $phase1$ ) do ▷ IDENTIFICATION PHASE
6:      $\mathbf{x}^{k+1} = P_\Omega(\mathbf{x}^k - \alpha^k \nabla \hat{f}^k)$  where  $\alpha^k$  satisfies suff. decrease conditions [7];
7:      $conv = (\|\mathbf{g}^P(\mathbf{x}^{k+1})\| \leq tol)$ ;
8:      $phase1 = (\mathcal{A}^{k+1} \neq \mathcal{A}^k) \wedge (\hat{f}^k - \hat{f}^{k+1} > \eta \max_{m \leq l < k} (\hat{f}^l - \hat{f}^{l+1})) \wedge (\neg conv)$ ;
9:      $k = k + 1$ ;
10:  end while
11:  if ( $conv$ ) then
12:     $phase2 = .false.$ ;
13:  end if
14:  while ( $phase2$ ) do ▷ MINIMIZATION PHASE
15:    Compute an approximate solution  $\mathbf{d}^k$  to the auxiliary problem (2);
16:     $\mathbf{x}^{k+1} = P_\Omega(\mathbf{x}^k + \alpha^k \mathbf{d}^k)$  with  $\alpha^k$  such that  $\hat{f}^{k+1} \leq \hat{f}^k$ ;
17:     $conv = (\|\mathbf{g}^P(\mathbf{x}^{k+1})\| \leq tol)$ ;
18:     $phase2 = (\|\boldsymbol{\beta}^{k+1}\| \leq \Gamma \|\boldsymbol{\varphi}^{k+1}\|) \wedge (\neg conv)$ ;
19:     $k = k + 1$ ;
20:  end while
21:   $phase1 = .true.$ ;  $phase2 = .true.$ ;
22: end while

```

4 Numerical Results

We have implemented our algorithms in MATLAB and used them to solve stationary 2D contact problems with optional Tresca friction and 3D frictionless contact problems. Let us recall that using the duality, the discretized contact problem reduces to the problem of finding the minimizer of

$$\begin{aligned} \min \Theta(\boldsymbol{\lambda}) &:= \frac{1}{2} \boldsymbol{\lambda}^\top \mathbf{F} \boldsymbol{\lambda} - \boldsymbol{\lambda}^\top \mathbf{d} \\ \text{s.t. } \mathbf{G} \boldsymbol{\lambda} &= \mathbf{0}, \\ \boldsymbol{\lambda} &\in \tilde{\Lambda}(\boldsymbol{\Psi}), \end{aligned}$$

where $\tilde{\Lambda}(\boldsymbol{\Psi}) = \Lambda(\boldsymbol{\Psi}) - \tilde{\boldsymbol{\lambda}}$, with

$$\Lambda(\boldsymbol{\Psi}) = \left\{ (\boldsymbol{\lambda}_n^\top, \boldsymbol{\lambda}_t^\top, \boldsymbol{\lambda}_E^\top)^\top \mid \boldsymbol{\lambda}_n \geq 0, \|\boldsymbol{\lambda}_{t,i}\| \leq \Psi_i, i = 1, \dots, n_t \right\}.$$

For 3D frictionless problem and 2D problem with a given (Tresca) friction, the feasible set is defined by bound and/or box inequality constraints as (1) (see, e.g., [17, 19]). The same type of problem arises in implementation of any time-step of implicit scheme for transient contact problems [18]. In 3D problems with friction, the box constraints are replaced by separable circular constraints, but the resulting QCQP problem can be solved by similar algorithms. A detailed formulation of contact problems and their effective discretization is out of the scope of this article therefore we refer the reader to and [19] from which we borrowed the notation.

4.1 2D Beam with Material Insets and Coulomb Friction

Our first benchmark is the 2D beam problem depicted in left part of Fig. 1, where inside the “soft” ($E = 4.4 \text{e}+5$, $\sigma = 0.34$) rectangular beam there are 8 stiff ($E = 1.6 \text{e}+7$, $\sigma = 0.32$) circular insets. The whole set of bodies is subject to a force applied to the right side of the soft beam as shown in the figure. The discretization of the problem leads to a problem with 2222 variables, 1024 of which are subject either to lower bounds or to both lower and upper bounds, and 60 linear equality constraints. The difficulty of this problem lies in the need for the iterative solver to distribute the global information across several nonlinear interfaces.

We solved six different instances of the problem, characterized by different choices for the boundary forces (F_x, F_y). The results obtained by solving the problem with the SMALBE algorithm equipped either with P2GP or with MPRGP are summarized in Table 1. In particular, the number of variables which are on a bound ($|\mathcal{A}|$), number of outer iterations and Hessian multiplications are shown.

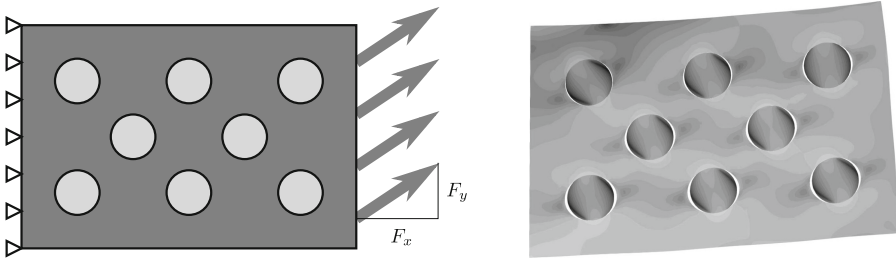


Fig. 1. 2D beam with insets - setting (left) and HMH stress (right)

4.2 Hertz 3D Problem Without Friction

The second benchmark is a 3D two-body contact problem depicted in Fig. 2 (left). The problem was solved by a variant of the FETI method introduced by Farhat and Roux [22] and adapted for contact problems by Dostál et al., see, e.g., [19]. Let us recall that FETI methodology transforms the minimization of convex energy function subject to general inequality and equality constraints in primal formulation to dual problem in Lagrange multipliers for “gluing” of the subdomains and non-penetration. The equality constraints in the dual formulation enable a correct reconstruction of the displacements, while the bound constrained multipliers for the nonpenetration are related to contact forces (pressure).

The stiff upper body ($E = 1.6 \text{e}+6$, $\sigma = 0.32$, $\rho = 5.08 \text{e}-9$) is pressed toward the softer lower one ($E = 4.4 \text{e}+5$, $\sigma = 0.34$, $\rho = 1.04 \text{e}-9$). The upper body has been divided in $3 \times 3 \times 2$ subdomains, while the lower one has been divided in $3 \times 3 \times 3$ subdomains; each subdomain has been divided in $10 \times 10 \times 10$ parts.

Table 1. 2D beam with insets - 6 benchmarks, P2GP \times MPRGP comparison

			P2GP				MPRGP			
F_x	F_y	$ \mathcal{A} $	$\ \mathbf{g}^P\ $	$\ \mathbf{Bx}\ $	out_it	Hess	$\ \mathbf{g}^P\ $	$\ \mathbf{Bx}\ $	out_it	Hess
100	0	890	1.5e-9	7.0e-7	15	1298	2.7e-14	5.0e-7	19	1667
75	15	879	7.7e-9	1.3e-6	18	1701	5.3e-11	9.9e-7	16	877
75	-15	878	4.0e-9	8.3e-7	16	1427	5.9e-13	8.0e-7	19	1494
-100	0	618	3.8e-9	9.9e-7	14	1432	4.1e-13	5.2e-8	14	1321
-75	15	667	5.8e-9	1.1e-7	16	1625	1.1e-12	1.6e-7	14	1380
-75	-15	666	1.6e-9	6.7e-7	14	1396	2.5e-12	3.9e-7	14	1332

For our test we fixed the radius of the lower body at -50 which translates into a concave surface and we chose two different radii for the upper body, namely 30 and 45. The first problem is characterized by 34854 variables, 900 of them are subject to lower bounds, while the second one is characterized by 34914 variables, 960 of them subject to lower bounds; both problems are subject to 270 linear equality constraints. The problem is not easy as on the solution comprises many dual degenerate components on the boundary of the active contact interface. The performance of both algorithms is summarized in Table 2.

4.3 3D Ball Bearing Without Friction

As last benchmark we chose an example of a real-life application, i.e the 3D multi-body contact problem describing the interaction between the various components of a ball bearing; in particular only a segment of the ball bearing is considered (see the right side of Fig. 3). The problem has been solved by means of the same variant of the FETI method used for the previous case. The discretization leads to a QP problem characterized by 19976 variables and 120 linear equality constraints. The active set at the optimal solution consists of 1199 variables among the 1212 subject to a lower bound. In this case the algorithm equipped with P2GP as inner solver took 51 iterations for the solution of the problem, with a total amount of 849 Hessian multiplications, thus outperforming the algorithm equipped with MPRGP which took 60 iterations and a total amount of 1188 Hessian products.

4.4 Comments

The result of the performed tests show that in some cases P2GP appears to be competitive with MPRGP, which is a standard choice for contact problems, and is sometimes able to outperform it. We conjecture that its more aggressive strategy for the expansion the active set performs better in problems where the percentage of active constraints is higher (see for example the case of the ball bearing and the case of the 2D beam). On the other hand, the results in Table 2 indicate that MPRGP is more efficient in treating problems with many dual degenerate components of the solution.

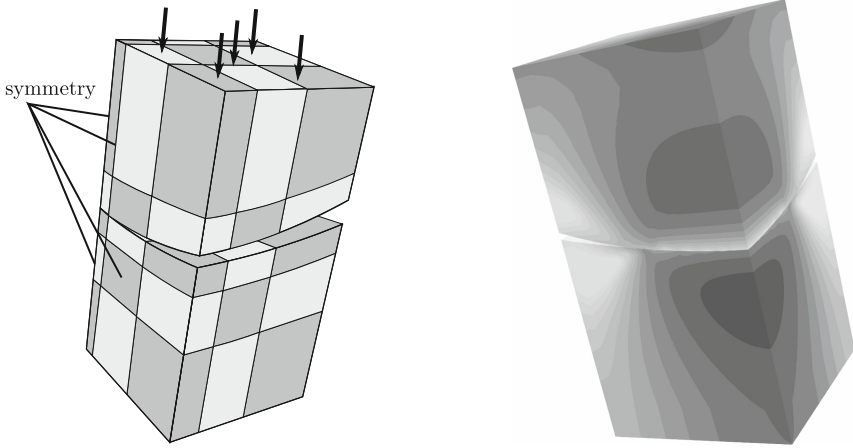


Fig. 2. Herz 3D setting (left) and HMH stress (right)

Table 2. 3D Hertz problem - 2 shape configuration with 4 different pressure for each one - P2GP \times MPGRP comparison

r_2	P	$ \mathcal{A} $	P2GP				MPGRP			
			$\ g^P\ $	$\ Bx\ $	out_it	Hess	$\ g^P\ $	$\ Bx\ $	out_it	Hess
30	1	856/900	4.1e-7	3.4e-6	78	1954	4.6e-7	3.6e-6	61	1687
	10	822/900	9.4e-7	3.7e-6	25	2103	8.2e-7	3.6e-6	28	1562
	100	729/900	3.3e-6	3.2e-6	17	2176	3.5e-6	3.3e-6	16	1544
	1000	556/900	1.4e-6	2.2e-6	14	3461	1.3e-6	3.6e-6	12	2447
45	1	887/960	1.5e-7	5.2e-7	34	1900	1.5e-7	5.1e-7	27	1617
	10	821/960	5.1e-7	5.8e-7	20	2306	4.6e-7	6.0e-7	18	1713
	100	659/960	3.3e-7	6.1e-7	13	3301	1.6e-7	5.4e-7	13	2778
	1000	324/960	1.1e-6	9.7e-7	13	3622	9.4e-7	8.5e-7	11	2496

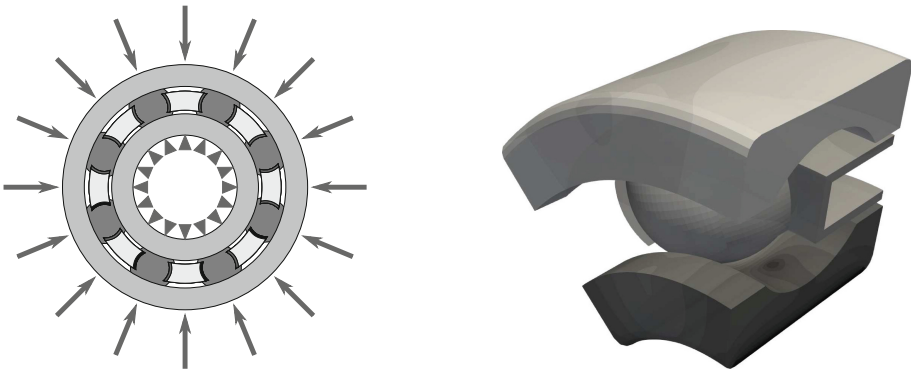


Fig. 3. Ball bearing setting (left) and displacement stress (right)

5 Conclusions

We have presented two methods for the solution of bound or box constrained convex quadratic programming problems. The methods combine monotone spectral gradient method, conjugate gradients, and gradient projections with different steplength choices. The methods were applied to the solution of auxiliary problems in the inner loop of the augmented Lagrangian algorithm for the analysis of 2D and 3D discretized contact problems. The experimental results confirm effectiveness of both algorithms. It seems that a suitable combination of both algorithms can result in still faster solver.

Acknowledgement. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPS II) project “IT4Innovations excellence in science - LQ1602” and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project “IT4Innovations National Supercomputing Center - LM2015070”. The work was partially supported by Gruppo Nazionale per il Calcolo Scientifico - Istituto Nazionale di Alta Matematica (GNCS-INdAM).

References

1. Antonelli, L., De Simone, V., di Serafino, D.: On the application of the spectral projected gradient method in image segmentation. *J. Math. Imaging Vis.* **54**(1), 106–116 (2015)
2. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. *IMA J. Numer. Anal.* **8**(1), 141–148 (1988)
3. Bielschowski, R.H., Friedlander, A., Gomes, F.A.M., Martínez, J.M., Raydan, M.: An adaptive algorithm for bound constrained quadratic minimization. *Invest. Oper.* **7**, 67–102 (1997)
4. Bonettini, S., Zanella, R., Zanni, L.: A scaled gradient projection method for constrained image deblurring. *Inverse Probl.* **25**, 015002 (2009)
5. Cafieri, S., D’Apuzzo, M., Marino, M., Mucherino, M., Toraldo, G.: Interior-point solver for large-scale quadratic programming problems with bound constraints. *J. Optim. Theory Appl.* **129**(1), 55–75 (2006)
6. Cafieri, S., D’Apuzzo, M., De Simone, V., di Serafino, D., Toraldo, G.: Convergence analysis of an inexact potential reduction method for convex quadratic programming. *J. Optim. Theory Appl.* **135**(3), 355–366 (2007)
7. Calamai, P.H., Moré, J.J.: Projected gradient methods for linearly constrained problems. *Math. Program.* **39**, 93–116 (1987)
8. di Serafino, D., Ruggiero, V., Toraldo, G., Zanni, L.: On the steplength selection in gradient methods for unconstrained optimization. *Appl. Math. Comput.* (2017). <https://doi.org/10.1016/j.amc.2017.07.037>
9. di Serafino, D., Toraldo, G., Viola, M., Barlow, J.: A two-phase gradient method for quadratic programming problems with a single linear constraint and bounds on the variables. *SIAM Journal on Optimization* (2018, to appear)
10. De Asmundis, R., di Serafino, D., Landi, G.: On the regularizing behavior of the SDA and SDC gradient methods in the solution of linear ill-posed problems. *J. Comput. Appl. Math.* **302**, 81–93 (2016)

11. Dostál, Z.: Box constrained quadratic programming with proportioning and projections. *SIAM J. Optim.* **7**(3), 871–887 (1997)
12. Dostál, Z.: Inexact semi-monotonic augmented Lagrangians with optimal feasibility convergence for convex bound and equality constrained quadratic programming. *SIAM J. Numer. Anal.* **43**(1), 96–115 (2005)
13. Dostál, Z.: Semi-monotonic inexact augmented Lagrangians for quadratic programming with equality constraints. *Optim. Methods Softw.* **20**(6), 715–727 (2005)
14. Dostál, Z.: On the decrease of a quadratic function along the projected-gradient path. *ETNA* **31**, 25–29 (2008)
15. Dostál, Z.: *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*. Springer, Heidelberg (2009). <https://doi.org/10.1007/b138610>
16. Dostál, Z., Domorádová, M., Sadowská, M.: Superrelaxation in minimizing quadratic functions subject to bound constraints. *Comput. Optim. Appl.* **48**(1), 23–44 (2011)
17. Dostál, Z., Kozubek, T., Markopoulos, A., Brzobohatý, T., Vondrák, V., Horyl, P.: Scalable TFETI algorithm for two dimensional multibody contact problems with friction. *J. Computat. Appl. Math.* **235**(2), 403–418 (2010)
18. Dostál, Z., Kozubek, T., Brzobohatý, T., Markopoulos, A., Vlach, O.: Scalable TFETI with preconditioning by conjugate projector for transient frictionless contact problems of elasticity. *CMAME* **247–248**, 37–50 (2012)
19. Dostál, Z., Kozubek, T., Sadowská, M., Vondrák, V.: *Scalable Algorithms for Contact Problems*. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-1-4939-6834-3>
20. Dostál, Z., Kozubek, T., Vondrák, V., Brzobohatý, T., Markopoulos, A.: Scalable TFETI algorithm for the solution of multibody contact problems of elasticity. *Int. J. Numer. Methods Eng.* **82**(11), 1384–1405 (2010)
21. Dostál, Z., Schöberl, J.: Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Comput. Optimiz. Appl.* **30**(1), 23–44 (2005)
22. Farhat, C., Roux, F.-X.: A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Methods Eng.* **32**, 1205–1227 (1991)
23. Frassoldati, G., Zanni, L., Zanghirati, G.: New adaptive stepsize selections in gradient methods. *J. Ind. Manag. Optim.* **4**, 299–312 (2008)
24. Friedlander, A., Martínez, J.M.: On the maximization of a concave quadratic function with box constraints. *SIAM J. Optim.* **4**, 177–192 (1994)
25. Moré, J.J., Toraldo, G.: On the solution of large quadratic programming problems with bound constraints. *SIAM J. Optim.* **1**, 93–113 (1991)



Schur Complement-Schwarz DD Preconditioners for Non-stationary Darcy Flow Problems

Radim Blaheta¹(✉), Tomáš Lubér¹, and Jakub Kružík^{1,2}

¹ Institute of Geonics CAS, Ostrava, Czech Republic
radim.blaheta@ugn.cas.cz

² IT4Innovations National Supercomputing Centre,
VSB - Technical University, Ostrava, Czech Republic

Abstract. The paper concerns development of highly parallelizable preconditioners for solving nonstationary Darcy flow problems. The discretization of the solved problem is done by mixed finite elements in space and by first order implicit Euler discretization in time. The systems with generalized saddle point matrices, which appear in each time step of the implicit Euler method, are then solved by FGMRES method with a block type preconditioner. Moreover, highly parallelizable, one-level additive Schwarz method is used for preconditioning of the velocity block. Both analysis and numerical experiment show that this application of the Schwarz method is highly efficient for a class of flow problems with parameters corresponding to many applications in geosciences.

1 Introduction

The fluid flow in porous media appears in many applications in geomechanics, environmental problems, biomechanics etc. This paper is devoted to a model of nonstationary flow in fully saturated media. The model is based on Darcy law and an assumption of nonzero storativity - ability to increase fluid amount in a volume with increasing fluid pressure. The storativity results from a slight compressibility of the fluid and deformability of the solid matrix.

The discretization of the porous media flow problem is done by a mixed finite elements in space, namely the lowest order Raviart-Thomas are used, see [10]. The Euler type systems with generalized saddle point matrices, which appear in each time step of the implicit Euler method, are then solved by MINRES or FGMRES method with Schur complement type preconditioner.

Numerical complexity is concentrated into solving the Euler type systems and especially into the preconditioning of the velocity block of the preconditioner. To this aim, highly parallelizable one-level additive Schwarz method can be used, see the analysis in [1]. Both analysis and numerical experiments show that this Schwarz method is highly efficient for a class of flow problems with material parameters corresponding to many applications in geosciences.

The content of this paper is as follows. Section 2 describes the nonstationary Darcy problem and its discretization. The block preconditioners for the systems arising in each time step of the implicit Euler method are described in Sect. 3. The Schwarz method for solving the velocity block systems is then described in Sect. 4 with analysis, which strengthens the results from [1]. Section 5 discusses implementation on parallel computers and provides numerical experiments on a massively parallel computer.

The main conclusion is that the combination of the block preconditioners and the additive Schwarz method provides efficient and highly parallelizable preconditioners for a class of nonstationary Darcy flow problems with parameters corresponding to many applications in geosciences and other fields.

2 Nonstationary Darcy Flow Problem and Its Discretization

The nonstationary Darcy problem for very slightly compressible liquid and matrix can be written in the following mixed form

$$\begin{aligned} K^{-1}v + \nabla p &= 0 \text{ in } \Omega, \\ \nabla \cdot v + c_{pp}\partial_t v &= f \text{ in } \Omega, \end{aligned} \tag{1}$$

where $\Omega \subset R^d$ is the problem domain, p is the fluid pressure, v is the Darcy velocity, $K \in R^{d \times d}$ is the permeability represented in a general anisotropic media by a symmetric positive definite matrix and c_{pp} is the storativity constant. Note that in Sects. 4 and 5, we restrict ourselves to isotropic media $K = kI$, where $k = k(x) \geq k_0 > 0$ and I is the identity matrix. The model is described in detail e.g. in [11].

The weak formulation of the problem (1) leads to finding the pair (v, p) , $v = v(x, t)$ and $p = p(x, t)$, which fulfils a mixed variational identity in $V \times X = H(\text{div}, \Omega) \times L_2(\Omega)$, see e.g. [13].

We assume discretization in $V_h \times X_h$ with Raviart-Thomas finite elements on squares [10] for velocity and piecewise constant functions for pressure. The choice of a basis $V_h = \text{span}\{\psi_i\}$ and $X_h = \text{span}\{\phi_i\}$ and induced isomorphisms $V_h \leftrightarrow \mathbf{V}_h \equiv R^N$, $v_h \leftrightarrow \mathbf{v}$, $X_h \leftrightarrow \mathbf{X}_h \equiv R^Z$, $p_h \leftrightarrow \mathbf{p}$ then provides differential algebraic system of the form

$$\begin{aligned} \mathcal{A}_1 \frac{d}{dt} \mathcal{U} + \mathcal{A} \mathcal{U} &= \mathcal{F} \\ \mathcal{A}_1 &= \begin{bmatrix} 0 & 0 \\ 0 & -C \end{bmatrix}, \quad \mathcal{A} = \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}, \quad \mathcal{U} = \begin{bmatrix} \mathbf{v} \\ \mathbf{p} \end{bmatrix}, \end{aligned}$$

where $M \in R^{N \times N}$, $B \in R^{Z \times N}$ and $C \in R^{Z \times Z}$ are matrices defined by the following identities

$$\langle M \mathbf{u}, \mathbf{v} \rangle = \int_{\Omega} K^{-1} u_h \cdot v_h \, d\Omega, \quad \langle B \mathbf{v}, \mathbf{p} \rangle = - \int_{\Omega} \text{div}(v_h) p_h \, d\Omega \quad \forall v_h \in V_h, p_h \in X_h, \tag{2}$$

$$\langle C\mathbf{p}, \mathbf{q} \rangle = \int_{\Omega} c_{pp} p_h q_h d\Omega, \quad p_h, q_h \in X_h, \quad (3)$$

where $\langle \cdot, \cdot \rangle$ denotes Euclidean inner product.

Note, that the regularity of the solution will be low when there are large jumps in the permeability, so we restrict ourselves only to the lowest order Raviart-Thomas elements for discretization of velocity.

Implicit Euler method [13] uses time discretization $0 = t_0 < \dots < t_k < \dots$ and computes the values $\mathcal{U}^k = \begin{bmatrix} v^k \\ p^k \end{bmatrix}$ in the time steps t_k , $k \geq 1$ by solving the systems with the matrices \mathcal{A}_E ,

$$\mathcal{A}_E \mathcal{U}^{k+1} = \mathcal{F}^{k+1} + \frac{1}{\tau_k} \mathcal{A}_1 \mathcal{U}^k, \quad (4)$$

$$\mathcal{A}_E = \frac{1}{\tau_k} \mathcal{A}_1 + \mathcal{A} = \begin{bmatrix} M & B^T \\ B & -\frac{1}{\tau_k} C \end{bmatrix}. \quad (5)$$

The time step $\tau_k = t_{k+1} - t_k$ can be variable or fixed. In the analysis of preconditioners, we use the notation $\tau_k \equiv \tau$ without a loss of generality since we always consider solving system with matrix (5) during one specific timestep.

3 Preconditioning of the Euler Type Systems

The preconditioners for \mathcal{A}_E are based on the two by two partition shown in (5). Since the matrix C is diagonal for the piecewise constant approximation for pressure, we can consider the Schur complement $M_C = M + \tau B^T C^{-1} B$ and either block diagonal or block triangular preconditioners with the Schur complement block M_C ,

$$\mathcal{P}_D = \begin{bmatrix} M_C & 0 \\ 0 & \frac{1}{\tau} C \end{bmatrix} \quad \text{and} \quad \mathcal{P}_T = \begin{bmatrix} M_C & B^T \\ 0 & -\frac{1}{\tau} C \end{bmatrix}$$

The block diagonal preconditioner \mathcal{P}_D is positive definite and in the ideal case (when the problems connected with M_C and C are solved exactly) can be combined with MINRES method. The convergence is then driven by spectral properties, specifically by the following localization of the spectrum of the preconditioned matrix

$$\sigma(\mathcal{P}_D^{-1} \mathcal{A}_E) \subset \left\langle \frac{-1 - \sqrt{5}}{2}, -1 \right\rangle \cup \left\langle \frac{-1 + \sqrt{5}}{2}, 1 \right\rangle.$$

The proof of this localization can be found e.g. in [6, 9]. Note that this result depends only on the algebraic structure of the matrix \mathcal{A}_E and the localization result is robust with respect to the material parameters (permeability, storativity) of the model and discretization parameters (h , τ).

Even stronger localization of spectrum of the preconditioned system occurs for the block triangular preconditioner,

$$\mathcal{P}_T^{-1}\mathcal{A}_E = \mathcal{P}_T^{-1} \begin{bmatrix} M & B^T \\ B & -\frac{1}{\tau}C \end{bmatrix} = \mathcal{P}_T^{-1} \begin{bmatrix} M_C & B^T \\ 0 & -\frac{1}{\tau}C \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ -\tau C^{-1}B & I_2 \end{bmatrix}.$$

Thus in the ideal case (exact solvers for M_C and C , no influence of finite arithmetic) it holds that

$$\mathcal{P}_T^{-1}\mathcal{A}_E = \begin{bmatrix} I_1 & 0 \\ -\tau C^{-1}B & I_2 \end{bmatrix}$$

and consequently $\sigma(\mathcal{P}_T^{-1}\mathcal{A}_E) = \{1\}$ and $\mathcal{P}_T^{-1}\mathcal{A}_E$ has minimal polynomial of order two $(\mathcal{P}_T^{-1}\mathcal{A}_E - I)^2 = 0$. The block triangular preconditioner spoils the symmetry, which requires to use e.g. GMRES method. Two iterations of any Krylov space method are sufficient to solve the system.

In the implementation described in Sect. 5, we use an exact solver for the block C but inexact solver for the velocity block M_C . This solver uses conjugate gradient (CG) method with one level additive Schwarz preconditioning described in the next section. Numerical experiments show that when the accuracy of the inner solver is reasonably good, the outer iterations realized by flexible GMRES behave similarly to the ideal case.

4 Additive Schwarz Method for the Velocity Block

Both preconditioners presented in the previous section require the solution of the system with the Schur complement matrix M_C . This matrix is symmetric and positive definite and therefore can be solved by the conjugate gradient method. This section describes one level additive Schwarz preconditioner for the Schur complement system including the theory based on element-by-element analysis. The results presented in this section extend the results from [1] by considering nonzero block C and by utilising the elementwise computed maximum contrast $c_{pp}^{-1}k$.

The preconditioner P_{AS} is defined via a decomposition of the finite element space $V_h = V_1 + \dots + V_m$, where the subspaces V_k are defined via an overlapping decomposition of the domain Ω , $\bar{\Omega} = \bar{\Omega}_1 \cup \dots \cup \bar{\Omega}_m$. We assume that $\bar{\Omega}_k$ are aligned with the finite element division \mathcal{T}_h , which is used for definition of V_h (lowest order Raviart-Thomas RT(0) elements). Then

$$V_k = \{v_h \in V_h, v_h \equiv 0 \text{ in } \Omega \setminus \Omega_k\}.$$

Functions $v_h \in V_h$ are represented by algebraic vectors $\mathbf{v} \in \mathbf{V}_h \equiv R^N$ through isomorphism $V_h \leftrightarrow \mathbf{V}$, the same isomorphism provides relations $V_k \leftrightarrow \mathbf{V}_k \equiv R^{N_k}$. The inclusion $V_k \subset V_h$ induces restriction $\mathbf{V} \rightarrow \mathbf{V}_k$ represented by the matrix $R_k \in R^{N \times N_k}$. Then the preconditioner P_{AS} to M_C can be defined as

$$P_{AS}^{-1} = \sum_{k=1}^m R_k^T M_{C_k}^{-1} R_k, \quad M_{C_k} = R_k M_C R_k^T. \quad (6)$$

The matrix $M_C = M + \tau B^T C^{-1} B$ is created from the matrices defined variationally in (2) and (3). For a subsequent analysis, it is important that M_C can be also defined variationally, in particular

$$\langle M_C \mathbf{u}, \mathbf{v} \rangle = m(u_h, v_h) + \tau d(u_h, v_h) = a(u_h, v_h),$$

where $m(u_h, v_h) = \int_{\Omega} k^{-1} u_h v_h dx$, we presume that k is constant on mesh elements from \mathcal{T}_h , $k = k_E$ on $E \in \mathcal{T}_h$, and $d(u_h, v_h)$ is defined as follows

$$\begin{aligned} d(u_h, v_h) &= \langle B^T C^{-1} B \mathbf{u}, \mathbf{v} \rangle = \langle C^{-1} B \mathbf{u}, B \mathbf{v} \rangle \\ &= \sum_i c_{ii}^{-1} \left(\int_{\Omega} \operatorname{div}(u_h) \psi_i dx \right) \left(\int_{\Omega} \operatorname{div}(v_h) \psi_j dx \right) \\ &= \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \operatorname{div}(u_h) dx \right) \left(\int_E \operatorname{div}(v_h) dx \right). \end{aligned}$$

The summation above is over $E \in \mathcal{T}_h$ and we use the fact that the basis functions ψ_i of the space X_h are equal to 1 on $E = E_i \in \mathcal{T}_h$ and to zero on the other elements $E \neq E_i$. Moreover, $c_{ij} = \delta_{ij} \int_{E_i} c_{pp} = \delta_{ij} c_{pp} |E|$, where δ_{ij} is the Kronecker's symbol. As $\operatorname{div}(u_h)$ is constant on E for RT(0) function u_h ,

$$\begin{aligned} d(u_h, v_h) &= \sum_E c_{pp}^{-1} |E|^{-1} \operatorname{div}(u_h) \operatorname{div}(v_h) |E|^2 \\ &= \sum_E \int_E c_{pp}^{-1} \operatorname{div}(u_h) \operatorname{div}(v_h) dx \end{aligned}$$

and we conclude that $a(u_h, v_h)$ is a weighted $H(\operatorname{div})$ inner product, which guarantees positive definiteness of a .

The condition number of the preconditioned matrix $P_{AS}^{-1} M_C$ can be bounded by

$$\operatorname{cond}(P_{AS}^{-1} M_C) \leq c_0 c_1, \quad (7)$$

see e.g. [7, 8], where the constants c_0, c_1 come from the conditions

$$\forall v_h \in V_h, \exists v_k \in V_k, \quad v_h = \sum_{k=1}^m v_k : \sum_{k=1}^m a(v_k, v_k) \leq c_0 a(v_h, v_h), \quad (8)$$

$$\forall v_h \in V_h, \forall v_k \in V_k, \quad v_h = \sum_{k=1}^m v_k : a(v_h, v_h) \leq c_1 \sum_{k=k_0}^m a(v_k, v_k). \quad (9)$$

The rest of this section is devoted to determining the values of c_0, c_1 . It is easy to show that c_1 can be taken as maximal number of subdomains which mutually intersect c_{nis} .

The estimate of c_0 is more complicated and requires a suitable construction of the decomposition of the elements $v \in \mathbf{V}$. To derive the estimate we analyse the decomposition provided by

$$v = \sum_{k=1}^m v_k, \quad v_k = \Pi_{RT}(\theta_k v),$$

where θ_k are functions of a decomposition of unity [18],

$$1 = \sum_{k=1}^m \theta_k, \quad \text{supp}(\theta_k) = \bar{\Omega}_k, \quad 0 \leq \theta_k \leq 1, \quad \|\text{grad}(\theta_k)\| \leq c\delta^{-1},$$

where δ is an overlap (usually a nonoverlapping decomposition $\bar{\Omega} = \bar{\Omega}_1^0 \cup \dots \cup \bar{\Omega}_m^0$ is enlarged to overlapping one by construction of subdomains $\Omega_k = \{x \in \Omega, \text{dist}(x, \Omega_k^0) \leq \delta\}$).

Our analysis will make use of the Raviart-Thomas interpolation $\Pi_h^{RT} : C(\Omega) \rightarrow RT_0$ given by

$$\Pi_h^{RT} v = \sum_i \left(\frac{1}{|e_i|} \int_{e_i} v \cdot n_{e_i} ds \right) \psi_i,$$

where the summation goes over the degrees of freedom (located on edges of the elements), see [15].

We will show that

$$\sum_k m(v_k, v_k) \leq \kappa c_{nis} m(v_h, v_h), \quad (10)$$

$$\sum_k d(v_k, v_k) \leq 2c_{nis} d(v_h, v_h) + 2\tau c_{nis} \delta^{-2} \max_E \{c_{pp}^{-1} k_E\} m(v_h, v_h) \quad (11)$$

The constant κ will be determined later in the analysis.

To derive the estimate (10) we consider

$$\begin{aligned} \sum_k m(v_k, v_k) &= \sum_k \sum_{E \subset \bar{\Omega}_k} \int_E k_E^{-1} v_k \cdot v_k = \sum_k \sum_{E \subset \bar{\Omega}_k} k_E^{-1} \int_E v_k \cdot v_k \\ &\leq c_{nis} \sum_{E \in \mathcal{T}_h} k_E^{-1} \int_E \|\Pi_h^{RT}(\theta_k v_h)\|^2 \leq c_{nis} \kappa \sum_E \int_E k_E^{-1} \|v_h\|^2 \\ &\leq c_{nis} \kappa m(v_h, v_h). \end{aligned}$$

Above, we used fact that $\Pi_h^{RT}(\theta_k v_h)|_E = \sum_i z_i \hat{\psi}_i$ where $\hat{\psi}_i$ are local (element) RT_0 basis functions and

$$z_i = \frac{1}{|e_i|} \int_{e_i} (\theta_k v_h) \cdot n_{e_i} ds = \frac{1}{|e_i|} \int_{e_i} \theta_k (v_h \cdot n_{e_i}) ds = (v_h \cdot n_{e_i}) \frac{1}{|e_i|} \int_{e_i} \theta_k ds \leq v_h \cdot n_{e_i}$$

as $v_h \cdot n_{e_i}$ is constant on e_i for $v \in RT_0$. Therefore,

$$\begin{aligned} \int_E \|\Pi_h^{RT}(\theta_k v_h)\|^2 &= \int_E \left\| \sum_i z_i \hat{\psi}_i \right\|^2 = \langle M_E \mathbf{z}, \mathbf{z} \rangle \leq \mu_{\max}(M_E) \|\mathbf{z}\|^2 \\ &\leq \mu_{\max}(M_E) \|v\|^2 \leq \frac{\mu_{\max}(M_E)}{\mu_{\min}(M_E)} \langle M_E v, v \rangle \\ &= \kappa \int_E \|v\|^2, \end{aligned}$$

where M_E is the velocity mass matrix, $(M_E)_{ij} = \int_E \hat{\psi}_i \cdot \hat{\psi}_j$ and $\kappa = \frac{\mu_{\max}(M_E)}{\mu_{\min}(M_E)}$, i.e. the condition number of the local mass matrix.

To prove (11), we investigate $\sum_k d(v_k, v_k)$

$$\begin{aligned} \sum_k d(v_k, v_k) &= \sum_k \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \operatorname{div}(\Pi_h^{RT}(\theta_k v_h)) \, dx \right)^2 \\ &= \sum_k \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \operatorname{div}(\theta_k v_h) \, dx \right)^2 \end{aligned} \quad (12)$$

$$\begin{aligned} &= \sum_k \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \theta_k \operatorname{div}(v_h) + \operatorname{grad}(\theta_k) \cdot v_h \, dx \right)^2 \\ &\leq 2c_{nis} \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \theta_k \operatorname{div}(v_h) \, dx \right)^2 \\ &\quad + 2c_{nis} \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \operatorname{grad}(\theta_k) \cdot v_h \, dx \right)^2 \\ &\leq 2c_{nis} \sum_E c_{pp}^{-1} |E|^{-1} \left(\operatorname{div}(v_h) \int_E \theta_k \, dx \right)^2 \end{aligned} \quad (13)$$

$$\begin{aligned} &\quad + 2c_{nis} \sum_E c_{pp}^{-1} |E|^{-1} \left(\int_E \|\operatorname{grad}(\theta_k)\|^2 \, dx \right) \left(\int_E \|v_h\|^2 \, dx \right) \\ &\leq 2c_{nis} \sum_E c_{pp}^{-1} |E|^{-1} \operatorname{div}(v_h)^2 |E|^2 \\ &\quad + 2c_{nis} \sum_E c_{pp}^{-1} |E|^{-1} (\delta^{-2} |E|) k_E \left(\int_E k_E^{-1} \|v_h\|^2 \, dx \right) \\ &= 2c_{nis} d(v_h, v_h) + 2c_{nis} \delta^{-2} \max_E \{c_{pp}^{-1} k_E\} m(v_h, v_h) \end{aligned}$$

In (13) we use the fact that $\operatorname{div}(v_h)$ is constant on elements from \mathcal{T}_h , (12) follows from

$$\int_E \operatorname{div}(\Pi_h^{RT}(\theta_k v)) \, dx = \int_E \operatorname{div}(\theta_k v) \, dx,$$

i.e.

$$\begin{aligned} 0 &= \int_E \operatorname{div}(\theta_k v - \Pi_h^{RT}(\theta_k v)) \, dx \\ &= \int_{\partial E} (\theta_k v - \Pi_h^{RT}(\theta_k v)) \cdot n \, ds \\ &= \sum_i \int_{e_i} \theta_k v \cdot n \, ds - \int_{e_i} \left(\frac{1}{|e_i|} \int_{e_i} \theta_k v \cdot n_{e_i} \, ds \right) n_{e_i} \cdot n \, ds \\ &= \sum_i \int_{e_i} \theta_k v \cdot n \, ds - \int_{e_i} \left(\frac{1}{|e_i|} \int_{e_i} \theta_k v \cdot (n_{e_i} \cdot n) \, ds \right) n_{e_i} \cdot n \, ds \\ &= \sum_i \int_{e_i} \theta_k v \cdot n \, ds - (n_{e_i} \cdot n)^2 \int_{e_i} \theta_k v \cdot n \, ds = 0 \end{aligned}$$

Note that n_{e_i} is an a priori selected normal, which is used for definition of the degrees of freedom and n is the outer normal to the element E , $n_{e_i} \cdot n = \pm 1$.

The whole estimate is now

$$\begin{aligned} \sum_k a(v_k, v_k) &\leq \left(\kappa c_1 + 2c_{nis} \tau \delta^{-2} \max_E \{c_{pp}^{-1} k_E\} \right) m(v_h, v_h) + 2c_{nis} d(v_h, v_h) \\ &\leq c_{nis} \max \left\{ 2, \kappa + 2\tau \delta^{-2} \max_E \{c_{pp}^{-1} k_E\} \right\} a(v_h, v_h) \end{aligned}$$

and as c_{nis} and κ are independent of physical and discretization parameters, the efficiency and robustness of the estimate depends mostly on the term

$$c_{AS} = \tau \delta^{-2} \max_E \{c_{pp}^{-1} k_E\}. \quad (14)$$

The above results can be summarized in the following theorem.

Theorem. *Let us consider the time step matrix A_E from (5) with time step τ and the Schur complement $M_C = M + \tau B^T C^{-1} B$. Let P_{AS} be the additive Schwarz preconditioner from (6). Then*

$$\operatorname{cond}(P_{AS}^{-1} M_C) \leq c_{nis}^2 \max \left\{ 2, \kappa + 2\tau \delta^{-2} \max_E \{c_{pp}^{-1} k_E\} \right\}.$$

We remind the notation in which c_{nis} is the maximum number of mutually overlapping subdomains, δ is the overlap of the decomposition, κ is maximum condition number of the element mass matrices M_E , c_{pp} and k_E are storativity and permeability assumed to be constant on the finite elements and $\max_E \{c_{pp}^{-1} k_E\}$ is taken over all elements of the finite element division.

Remark. Note that for 2D and RT(0) elements on squares $\langle x_{01}, x_{01} + h \rangle \times \langle x_{02}, x_{02} + h \rangle$, we get

$$M_E = \frac{1}{6}h^2 \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ -1 & 0 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{bmatrix}, \quad \sigma(M_E) = \frac{1}{6}h^2 \{1; 3\}, \quad \kappa = 3.$$

5 Implementation and Numerical Experiments

The numerical experiments are computed with our own code available at [19] written in C on top of PETSc [5, 16]. Matrices M , M_C , B and C are assembled element by element from local contributions. All of these matrices are created and stored in a distributed form using PETSc MatCreateAIJ operation. The system matrix \mathcal{A}_E and the preconditioner matrix \mathcal{P} are then formed implicitly from blocks using PETSc MATNEST matrix type. The action of the preconditioner, which combines separate preconditioners for individual fields, is provided by PETSc PCFIELDSPLIT operation.

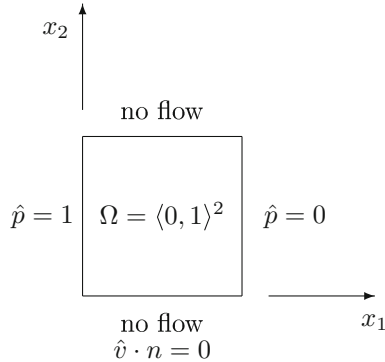


Fig. 1. Model problem

For numerical experiments we use a model problem described by (1) with the zero volume source $f \equiv 0$, boundary conditions as shown in Fig. 1 and initial condition $v = 0$ and $p = 0$ in Ω . The problem domain $\Omega = \langle 0, 1 \rangle^2$ is regularly divided into square elements with the meshsize characterized by the parameter $n = 1/h$ being the number of segments on the side. Timestep $\tau = 0.1$ is used for all experiments. The permeability of the material is supposed to be isotropic and elementwise constant. Values on each element are in the form $k = k_s k_r$, $\log k = \log k_s + \log k_r$, where k_r are sampled from lognormal distribution with parameters $\mu = 0$ and $\sigma = 2$, $\log k \in N(\log k_s, \sigma^2)$.

FGMRES with the block preconditioner is used to solve the outer system with the matrix \mathcal{A}_E and conjugate gradients with Schwarz preconditioner are used to solve the inner system corresponding to the block M_C . The stopping criterion for both outer and inner iterations is the reduction of relative unpreconditioned residual to be equal or less than 10^{-6} .

The Schwarz preconditioner for the matrix M_C uses the PETSc PCASM functionality. The decomposition to subdomains for the model problem corresponds to the splitting of the domain into horizontal strips which for the regular mesh and natural numbering of nodes corresponds to row-wise matrix decomposition. In PETSc, the overlap is imposed by adding a proper number of matrix rows to implicitly performed nonoverlapping row-wise matrix splitting. The preconditioner uses LU decomposition for the solution of systems on subdomains. The LU decomposition is computed during the setup of the preconditioner and then it is repeatedly applied during the iterations. Possible generalization of this approach to 3D problem discretized with regular mesh and numbering aligned with domain decomposition on layers is straightforward.

Table 1 shows scalability of the implementation. We investigate “weak” scaling with problem size increasing with increasing number of subdomains (processors), i.e. the size of the subproblems is kept not strongly decreasing. Note that the subproblems arise by decomposition of $\mathcal{A}_E \in R^{N_t \times N_t}$ and $M_C \in R^{N \times N}$. For the problem on $n \times n$ mesh and RT(0)-P0 elements, $N_t \sim 3n^2$ and $N \sim 2(nsd \cdot n)^2$, $nsd = 2no + n/np$, no is the size of overlap, np is the number of processors. The parameter no corresponds to number of rows of $n \times n$ mesh common to neighbouring subdomains, the geometrical overlap δ is obtained by multiplication no by the width of the row/strip. The number no is changing to keep the value δ not strongly dependent on the mesh size (h).

The values of outer iterations in Tables 1–3 report the average number of both outer FGMRES and inner CG iterations over one time step (averaged over the first ten timesteps, with zero initial guess within each timestep). Division of the number of inner iterations by number of outer iterations shows how many inner iterations are needed for the inner systems. Note that in a class of parameters, which we investigate there is frequently just one inner iteration per outer one. The time in all tables express the time spent by solvers that means without including times for matrix assembly and initialization of preconditioner. The computations were performed on the Salomon supercomputer, see [17].

The results in Table 1 correspond to the material parameters $k_s = 10^{-15}$, $c_{pp} = 10^{-10}$, $k_s/c_{pp} = 10^{-5}$ and use of more efficient triangular preconditioner. Tables 2 and 3 report number of outer/inner iterations in dependence on the material parameters, especially on the ratio k_s/c_{pp} . It can be seen that for both triangular preconditioner (Table 2) and diagonal preconditioner (Table 3) the efficiency is excellent if $k_s/c_{pp} \leq 10^{-4}$. This condition is fulfilled for many applications in geomechanics and biomechanics, see Table 5.

Tables 2 and 3 illustrate the dependence of the iterative processes on material parameters k_s and c_{pp} . The first number in each cell is the average number of outer iterations and the second is the average number of inner iterations over one time step both averaged over the first ten timesteps, with zero initial guess within

Table 1. Test on scaling: mesh size $n \times n$, number of DOFs for RT(0)-P0 elements $\sim 3n^2$. Material parameters $k_s = 10^{-15}$, $c_{pp} = 10^{-10}$, $k_s/c_{pp} = 10^{-5}$.

n	Processors/np	Overlap/no	Outer it.	Inner it.	Time[s]
406	2	2	2.0	2.0	3.0
574	4	2	2.0	2.0	2.8
805	8	4	2.0	2.0	2.7
1122	16	6	2.0	2.0	2.6
1550	32	8	2.0	2.0	2.5
2106	64	10	2.0	2.9	2.7
2722	128	14	2.0	4.7	3.8
3390	256	17	2.0	6.3	6.0
4800	512	21	2.0	8.5	18.3

each timestep to assess the convergence without the influence good initial guess from previous timestep. The Table 4 shows dependence on the overlap measured by number of rows. The overlap is increasing in the rows, decreasing in the column. The test is done for unfavourable ratio $k_s/c_{pp} = 1$. For favourable ratio $k_s/c_{pp} = 1 \leq 10^{-4}$ the dependence on the overlap is very weak.

Table 2. Dependence of number of iterations for FGMRES - \mathcal{P}_T (triangular preconditioner) and CG - Schwarz on material parameters k_s , $\sigma = 2$, c_{pp} . Other parameters $n = 1000$, 24 subdomains, overlap 8 are kept constant.

$\downarrow k_s \setminus c_{pp} \rightarrow$	10^{-3}	10^{-4}	10^{-5}	10^{-6}
10^{-7}	2.1/3.7	20/9.7	2.1/21.0	2.1/78.6
10^{-8}	2.0/2.0	2.1/3.7	2.0/9.7	2.1/21.0
10^{-9}	2.0/2.0	2.0/2.0	2.1/3.7	2.0/9.7
10^{-10}	2.0/2.0	2.0/2.0	2.0/2.0	2.1/3.7
10^{-11}	2.0/2.0	2.0/2.0	2.0/2.0	2.0/2.0

Table 3. Dependence of number of iterations for FGMRES - \mathcal{P}_D (diagonal preconditioner) and CG - Schwarz on material parameters k_s , $\sigma = 2$, c_{pp} . Other parameters $n = 1000$, 24 subdomains, overlap 8 are kept constant.

$\downarrow k_s \setminus c_{pp} \rightarrow$	10^{-3}	10^{-4}	10^{-5}	10^{-6}
10^{-7}	15.9/44.6	14.9/87.1	14.2/193.9	11.3/437.1
10^{-8}	16.5/24.9	15.9/44.6	14.9/87.1	14.2/193.9
10^{-9}	16.3/16.3	16.5/24.9	15.9/44.6	14.9/87.1
10^{-10}	15.4/15.4	16.3/16.3	16.5/24.9	15.9/44.6
10^{-11}	12.2/12.2	15.4/15.4	16.3/16.3	16.5/24.9

Table 4. Dependence on the geometrical overlap δ for triangular preconditioner, $k_S = 10^{-10}$, $c_{pp} = 10^{-10}$, $\sigma = 0$, 24 processes.

$n \backslash \text{overlap}$	4	8	16
100	2.1/43.6	2.1/31.4	2.1/24.0
200	20/63.0	2.1/47.3	2.1/34.4
400	2.1/79.2	2.0/65.9	2.1/47.8
800	2.1/101.6	2.0/82.1	2.0/65.8
1600	2.1/141.2	2.1/106.7	2.1/83.5

Table 5. Ranges of material parameters

	k [m ²]	c_{pp} [Pa ⁻¹]
Unweathered clay	$10^{-20} - 10^{-16}$	$10^{-8} - 10^{-6}$
Jointed rocks	$10^{-15} - 10^{-11}$	$10^{-10} - 10^{-8}$
Sound crystalline rocks	$10^{-20} - 10^{-16}$	$10^{-11} - 10^{-9}$
Water		$4.4 \cdot 10^{-10}$

6 Conclusions

This paper presents iterative technique for solving the systems arising from non-stationary Darcy flow problems discretized by mixed finite elements in space and implicit Euler method in time. The technique combines outer iteration by FGMRES and block preconditioner with the velocity block solved by inner CG iteration with Schwarz type preconditioner.

It is shown that the convergence of the outer iterations is practically independent on the material parameters and inner iterations converge extremely fast for the ratio of permeability to storativity small enough ($k_s/c_{pp} \leq 10^{-4}$). This observation derived from the numerical tests is also in good agreement with our theoretical result (14), which follows from extending the analysis provided in [1]. As another robust inner iterative method, we can mention e.g. [14].

Such suitable ratio of permeability to storativity is characteristic in many geo applications dealing with semi-pervious and impervious materials, see the following values provided e.g. by references [11, 12]:

The presented iterative solution technique is also highly parallelizable and numerical experiments demonstrate its scalability.

The technique can be also used in the case that time discretization is done by higher order scheme, such as e.g. the Radau IIA method [3]. The systems arising within time steps of Radau method can be preconditioned by block preconditioner involving Euler type matrices as blocks, see [4]. The results of this paper can be also used for solving poroelasticity problems, cf. [2–4].

Acknowledgement. The work was done within the projects LD15105 “Ultrascale computing in geosciences” and LQ1602 “IT4Innovations excellence in science” supported by the Ministry of Education, Youth and Sports of the Czech Republic. The third author acknowledges the support of the Czech Science Foundation (GACR) project no. 15-18274S. Authors would also like to thank the referees for their helpful comments.

References

1. Axelsson, O., Blaheta, R.: Preconditioning of matrices partitioned in 2×2 block form: eigenvalue estimates and Schwarz DD for mixed FEM. *Numer. Linear Algebra Appl.* **17**, 787–810 (2010)
2. Axelsson, O., Blaheta, R., Byczanski, P.: Stable discretization of poroelasticity problems and efficient preconditioners for arising saddle point type matrices. *Comput. Vis. Sci.* **15**, 191–207 (2012). <https://doi.org/10.1007/s00791-013-0209-0>
3. Axelsson, O., Blaheta, R., Kohut, R.: Preconditioned methods for high order strongly stable time integration methods with an application for a DAE problem. *Numer. Linear Algebra Appl.* **22**, 930–949 (2015)
4. Axelsson, O., Blaheta, R., Lubner, T.: Preconditioners for Mixed FEM solution of stationary and nonstationary porous media flow problems. In: Lirkov, I., Margenov, S.D., Waśniewski, J. (eds.) *LSSC 2015*. LNCS, vol. 9374, pp. 3–14. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26520-9_1
5. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object-oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (eds.) *Modern Software Tools for Scientific Computing*, pp. 163–202. Birkhäuser, Boston (1997). https://doi.org/10.1007/978-1-4612-1986-6_8
6. Bean, M., Lipnikov, K., Yi, S.-Y.: A block-diagonal preconditioner for a four-field mixed finite element method for Biot’s equations. *Appl. Numer. Math.* **122**, 1–13 (2017)
7. Björstad, P.E., Mandel, J.: On the spectra of sums of orthogonal projections with applications to parallel computing. *BIT* **31**, 76–88 (1991)
8. Blaheta, R.: Space decomposition preconditioners and parallel solvers. In: Feistauer, M., Dolejší, V., Knobloch, P., Najzar, K. (eds.) *Numerical Mathematics and Advanced Applications*, pp. 20–38. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-642-18775-9_2
9. Blaheta, R., Lubner, T.: Algebraic preconditioning for Biot-Barenblatt poroelastic systems. *Appl. Math.* **62**(6), 561–577 (2017)
10. Boffi, D., Brezzi, F., Fortin, M.: *Mixed Finite Element Methods and Applications*. Springer Series in Computational Mathematics, vol. 44. Springer, Berlin (2013). <https://doi.org/10.1007/978-3-642-36519-5>
11. Chen, Z., Huan, G., Ma, Y.: *Computational Methods for Multiphase Flows in Porous Media*. SIAM, Philadelphia (2006)
12. Duffield, G.M.: Representative Values of Hydraulic Properties. *AQTESOLV*. http://www.aqtesolv.com/aquifer-tests/aquifer_properties.htm
13. Ern, A., Guermond, J.-L.: *Theory and Practice of Finite Elements*. Springer, New York (2004). <https://doi.org/10.1007/978-1-4757-4355-5>
14. Kraus, J., Lazarov, R., Lymbery, M., Margenov, S., Zikatanov, L.: Preconditioning heterogeneous $H(\text{div})$ problems by additive schur complement approximation and applications. *SIAM J. Sci. Comput.* **38**(2), A875–A898 (2016)

15. Kwak, D.Y., Pyo, H.C.: Mixed finite element methods for general quadrilateral grids. *Appl. Math. Comput.* **217**, 6556–6565 (2011)
16. PETSc - Portable: Extensible Toolkit for Scientific Computation, Argone National Laboratory. <http://www.mcs.anl.gov/petsc/>
17. SALOMON - IT4Innovations National Supercomputing Centre, Ostrava CR. <https://docs.it4i.cz/salomon>
18. Toselli, A., Widlund, O.B.: *Domain Decomposition Methods — Algorithms and Theory*. Springer, Berlin (2005). <https://doi.org/10.1007/b137868>
19. www.ugm.cas.cz/publish/software/HDIV/hdiv.zip



Relating Computed and Exact Entities in Methods Based on Lanczos Tridiagonalization

Tomáš Gergelits^{1,2}, Iveta Hnětynková¹, and Marie Kubínová^{1,2}(✉)

¹ Faculty of Mathematics and Physics, Charles University,
121 16 Prague, Czech Republic

{gergelits,hnetynko,kubinova}@karlin.mff.cuni.cz

² Institute of Computer Science, The Czech Academy of Sciences,
182 07 Prague, Czech Republic

Abstract. Krylov subspace methods based on short recurrences such as CGL or MINRES represent an attractive way of solving large and sparse systems of linear algebraic equations. Loss of orthogonality in the underlying Lanczos process delays significantly their convergence in finite-precision computation, whose connection to exact computation is still not fully understood. In this paper, we exploit the idea of simultaneous comparison of finite-precision and exact computations for CGL and MINRES, by taking advantage of their relationship valid also in finite-precision arithmetic. In particular, we show that finite-precision CGL residuals and Lanczos vectors have to be aggregated over the intermediate iterations to form a counterpart to vectors from the exact computation. Influence of stagnation in exact MINRES computation is also discussed. Obtained results are supported by numerical experiments.

Keywords: Krylov subspace · CGL · MINRES
Finite-precision computations · Loss of orthogonality
Delay of convergence · Lanczos vectors

1 Introduction

Large and sparse linear algebraic problems of a general form

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, \quad b \in \mathbb{R}^n,$$

can be often solved efficiently by Krylov subspace methods. Many of these rely mathematically on computation of an orthonormal basis of the Krylov subspaces

$$\mathcal{K}_k(A, r_0) \equiv \text{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}, \quad k = 1, 2, \dots, \quad (1)$$

where $r_0 = b - Ax_0$, with x_0 being the initial approximation. For a symmetric A , such basis can be efficiently computed by *short recurrences*, represented by

the Lanczos algorithm [11]. However, in finite-precision arithmetic, the global orthogonality and subsequently also the linear independence of the computed Lanczos vectors \bar{v}_j is usually quickly lost, and the subspaces spanned by \bar{v}_j are not Krylov subspaces defined by the input data. As a result, the convergence of methods such as the Conjugate gradients implemented via the Lanczos algorithm (CGL) [9, 11] or the Minimal residual method (MINRES) [17], is *significantly delayed*, and the computed entities, including approximate solutions or residuals, can deviate substantially from their mathematical counterparts; see [6]. For some problems, the computation may not be affected by this delay, for example because the desired accuracy of the approximate solution is reached before the severe loss of orthogonality emerges (e.g., when efficient preconditioning can be used). However, short recurrences in principle cannot guarantee the linear independence of the computed vectors when rounding errors are present. Various techniques for preserving orthogonality, such as the full or selective reorthogonalization (see, e.g., [21, 22] or [13, Sect. 4.5]) have been developed, but for large scale problems, reorthogonalization in the Lanczos algorithm is typically unaffordable since it heavily increases computational time and storage requirements.

The first significant step in explaining the behavior of the Lanczos algorithm in finite-precision arithmetic was made in [15, 16]. It was proved that the loss of orthogonality among the computed Lanczos vectors is possible only in the directions of eigenvectors of the matrix A (more specifically, in the directions of Ritz vectors associated with converged Ritz values). Another fundamental step was done in [6, 7] showing that the behavior in the first k steps of the finite-precision Lanczos computations is identical to the behavior of exact Lanczos computations applied to a possibly larger matrix $\hat{A}(k)$, whose eigenvalues lie within tiny intervals around the eigenvalues of A ; see also [13] for an overview. In [19] the finite-precision Lanczos process in step k is described via the exact Lanczos process applied on augmented system containing both the matrix A and the currently computed tridiagonal Jacobi matrix. Sensitivity of Krylov subspace to small perturbations of the input data was studied in [1, 10, 20]. However, these results assume linear independence of the computed Lanczos vectors, which is often quickly lost.

Despite the wide attention, the properties of the methods based on the Lanczos process are in finite-precision computations still not fully understood. In particular, it is not clear how the subspaces generated by the computed Lanczos vectors differ from the exact Krylov subspaces, or how the computed approximation or residual vectors resemble their counterparts from exact computation with the *same matrix and starting vector*. The approaches in [6, 7, 19] do not allow direct comparison of the solution, residual, or Lanczos vectors, since they involve extended or augmented matrices.

However, combining [6, 7] together with the analysis of the convergence of the exact CGL in [14] gives sufficient reasoning to relate A -norm of the error in the k -th iteration of finite-precision CGL computation with (an earlier) l -th iteration of exact computation with the same data as

$$\|\bar{x}_k^{\text{L}} - x\|_A \approx \|x_l^{\text{L}} - x\|_A.$$

The gap $k - l$ corresponds to the notion of the rank-deficiency of the computed matrix of Lanczos vectors or to the delay of convergence, see Fig. 1. Even though this idea has appeared in the literature repeatedly (see, e.g., [12, Sect. 5.9], [5, Chap. 3], [8, Sect. 6.7.4]) determination of the corresponding iterations $[k, l]$ is still an open question and can be highly problem-dependent. Furthermore, to the best of our knowledge, the possibility of comparison of other entities, especially those whose size does not decay monotonically, has not been addressed in literature.

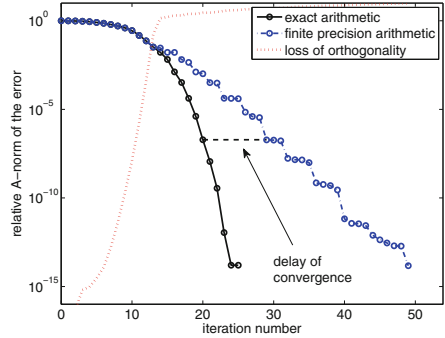


Fig. 1. Illustration of the loss of orthogonality and delay of convergence in CGL.

In this paper, we consider A symmetric positive-definite¹ and we exploit the idea of *simultaneous comparison* of finite-precision and exact computations for two related methods – CGL and MINRES. Since they form a pair of the norm-minimizing and the Galerkin method, we take advantage of their relationships proved in [2] to be approximately valid also in finite-precision computations. Because some finite-precision iterations k are a redundant consequence of reappearing information due to the delay of convergence, we do not consider all k . We rather assume we are given a subsequence $\{k_l\}_{l=1}^m$, $m \leq n$, where k_l is the finite-precision iteration related to the exact iteration l in the sense that the minimized quantities, i.e., A -norm of the error for CGL and residual norm for MINRES, are comparable between the two computations.² We show that some of the other entities cannot be compared directly. In particular, finite-precision CGL residuals (as well as their norms) and Lanczos vectors have to be *aggregated* over the intermediate iterations to form a counterpart to exact entities. We discuss influence of stagnation of MINRES on this comparison. Next, we discuss approaches to determine the subsequence $\{k_l\}$. Validity of obtained results is illustrated on numerical examples with matrices with various eigenvalue distribution.

The paper is organized as follows. Section 2 summarizes the Lanczos process and the two methods based on it - CGL and MINRES. Section 3 studies the relations between finite-precision and exact entities. Section 4 proposes some approaches to construct the subsequence $\{k_l\}$. Section 5 provides numerical experiments. Section 6 gives the conclusions. Throughout the paper, we assume $x_0 = 0$, i.e., $r_0 = b$; $\|\cdot\|, \|\cdot\|_A$ denotes the Euclidean and the energy norm respectively; e_j denotes the j -th column of the identity matrix of a suitable size. The entities computed in finite-precision arithmetic are denoted by bar.

¹ We only assume positive-definite matrices, so that the CGL iterations are well-defined in each step, although MINRES is well-defined also for indefinite matrices.

² The length of the subsequence, i.e., the index m , is typically determined by the iteration in which the finite-precision computation reaches the maximum attainable accuracy; see [12, Sect. 5.9.3].

2 Methods Based on Lanczos Tridiagonalization

Let $A \in \mathbb{R}^{n \times n}$ be a non-singular symmetric positive-definite matrix. Starting from a vector $v_1 = b/\delta_1$, $\delta_1 = \|b\|$, and initializing $v_0 = 0$, the tridiagonalization [11] computes, for $k = 1, 2, \dots$,

$$\begin{aligned} \gamma_k &= (Av_k, v_k); \\ v_{k+1} &= Av_k - \gamma_k v_k - \delta_k v_{k-1}; \\ \delta_{k+1} &= \|v_{k+1}\|, \quad \text{if } \delta_{k+1} = 0, \text{ then stop;} \\ v_{k+1} &= v_{k+1}/\delta_{k+1}. \end{aligned} \tag{2}$$

Vectors v_1, \dots, v_k form an orthonormal basis of the Krylov subspace (1). For simplicity of notation, we assume that the process (2) does not terminate before the iteration n , i.e., $\delta_{j+1} > 0$, $j = 1, \dots, n-1$. Denoting $V_k \equiv [v_1, \dots, v_k] \in \mathbb{R}^{n \times k}$ and

$$T_k \equiv \begin{bmatrix} \gamma_1 & \delta_2 & & & \\ \delta_2 & \gamma_2 & \delta_3 & & \\ & \delta_3 & \ddots & \ddots & \\ & & \ddots & \ddots & \delta_k \\ & & & \delta_k & \gamma_k \end{bmatrix} \in \mathbb{R}^{k \times k}, \quad T_{k+1,k} \equiv \begin{bmatrix} T_k \\ e_k^T \delta_{k+1} \end{bmatrix} \in \mathbb{R}^{(k+1) \times k},$$

we can write the matrix formulation of the Lanczos tridiagonalization as

$$AV_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T = V_{k+1} T_{k+1,k} \quad k = 1, \dots, n. \tag{3}$$

Based on [18], the Eq. (3) is in finite-precision replaced by

$$A\bar{V}_k = \bar{V}_k \bar{T}_k + \bar{\delta}_{k+1} \bar{v}_{k+1} e_k^T + \bar{F}_k = \bar{V}_{k+1} \bar{T}_{k+1,k} + \bar{F}_k, \quad k = 1, 2, \dots,$$

where \bar{F}_k is a small round-off term.

CGL and MINRES represent two methods based on the Lanczos tridiagonalization (2). At the k -th step, they search for the approximation of the solution in the subspace generated by the vectors v_1, \dots, v_k , i.e., $x_k = V_k y_k$ for some $y_k \in \mathbb{R}^k$. The corresponding residual has the form

$$r_k \equiv b - Ax_k = b - AV_k y_k = V_{k+1}(\delta_1 e_1 - T_{k+1,k} y_k).$$

The CGL method as a Galerkin method imposes the orthogonality of the residuals yielding

$$T_k y_k^L = \delta_1 e_1. \tag{4}$$

MINRES minimizes the norm of the residual r_k yielding

$$y_k^M = \operatorname{argmin}_{y \in \mathbb{R}^k} \|\delta_1 e_1 - T_{k+1,k} y\|. \tag{5}$$

Table 1. An overview of the decay properties of various entities in CGL and MINRES computations with respect to (1). For more details see [4].

Method/quantity	$\ x_k - x\ $	$\ x_k - x\ _A$	$\ r_k\ $	$\ r_k\ _A$
CGL	Monotone	Minimized	–	–
MINRES	Monotone	Monotone	Minimized	–

Decay properties of various entities in CGL and MINRES are summarized in Table 1. The residual vectors of the two methods can be related as

$$r_k^M = c_k^2 r_k^L + s_k^2 r_{k-1}^M \quad \text{with} \quad s_k^2 = \frac{\|r_k^M\|^2}{\|r_{k-1}^M\|^2}, \tag{6}$$

where s_k and c_k are the sine and cosine of the last Givens rotation used to eliminate the subdiagonal entry of the tridiagonal matrix $T_{k+1,k}$. The residual norms are related by the so-called *peak-plateau relation*

$$\|r_k^L\| = \frac{\|r_k^M\|}{\sqrt{1 - (\|r_k^M\|/\|r_{k-1}^M\|)^2}}, \tag{7}$$

or recursively, for $p = 0, 1, \dots$,

$$\frac{1}{\sqrt{\sum_{j=k}^{k+p} 1/\|r_j^L\|^2}} = \frac{\|r_{k+p}^M\|}{\sqrt{1 - (\|r_{k+p}^M\|/\|r_{k-1}^M\|)^2}}. \tag{8}$$

For more details see [2].

3 Comparison of Finite-Precision and Exact Entities

In this section, we show how the vectors r_l^L and v_l can be compared to their finite-precision counterparts based on the relations between CGL and MINRES residuals. We assume that we have a sequence $\{k_l\}_{l=1}^m$ satisfying

$$\|x_l^L - x\|_A \approx \|\bar{x}_{k_l}^L - x\|_A, \quad \|r_l^M\| \approx \|\bar{r}_{k_l}^M\|, \tag{9}$$

$$x_l^L \approx \bar{x}_{k_l}^L, \quad r_l^M \approx \bar{r}_{k_l}^M, \tag{10}$$

i.e., the quantities minimized in exact arithmetic and the corresponding vectors are comparable to their finite-precision counterparts, in the sense that the distance between them measured in a suitable norm is small relative to their size. The first of the assumptions is based on the observation made in [12, Sect. 5.9], where the question of the delay of CG convergence and the associated rank deficiency of the computed subspace has been addressed. Approaches to find such a sequence are discussed in Sect. 4. We consider problems where T_k is not too badly conditioned and thus the error due to the inexact solution of (4) and (5) is negligible, and

$$\bar{r}_k^{M,L} = \bar{V}_{k+1}(\|b\|e_1 - \bar{T}_{k+1,k}\bar{y}_k^{M,L}) - \bar{F}_k\bar{y}_k^{M,L}. \tag{11}$$

Sects. 3.1–3.2 consider the case when the exact MINRES does not stagnate, i.e., $\|r_l^M\|/\|r_{l-1}^M\| \not\approx 1$ and subsequently $\|\bar{r}_{k_l}^M\|/\|\bar{r}_{k_{l-1}}^M\| \not\approx 1$. The other case is discussed in Sect. 3.3.

3.1 CGL Residual Norms and Vectors

First, we relate the CGL *residual norms*. Provided that MINRES does not stagnate, we obtain from (9) that

$$\frac{\|r_l^M\|}{\sqrt{1 - (\|r_l^M\|/\|r_{l-1}^M\|)^2}} \approx \frac{\|\bar{r}_{k_l}^M\|}{\sqrt{1 - (\|\bar{r}_{k_l}^M\|/\|\bar{r}_{k_{l-1}}^M\|)^2}}, \quad (12)$$

where the error of approximation is determined by the error of approximation in (9). Since $\|\bar{r}_{k_l}^M\|/\|\bar{r}_{k_{l-1}}^M\| \not\approx 1$, applying the technique from [2, Theorem 4], we see that (8) is approximately valid also in finite-precision computation, i.e.,

$$\frac{1}{\sqrt{\sum_{j=k_{l-1}+1}^{k_l} 1/\|\bar{r}_j^L\|^2}} \approx \frac{\|\bar{r}_{k_l}^M\|}{\sqrt{1 - (\|\bar{r}_{k_l}^M\|/\|\bar{r}_{k_{l-1}}^M\|)^2}}, \quad (13)$$

where the error of approximation is determined by the round-off terms established in [2] not related to (9). Combining (12) and (13) with (7), we conclude that

$$\|r_l^L\| \approx \frac{1}{\sqrt{\sum_{j=k_{l-1}+1}^{k_l} 1/\|\bar{r}_j^L\|^2}}. \quad (14)$$

In words, the CGL residual norms cannot be compared directly, but finite-precision norms have to be *aggregated over the intermediate iterations*.

Now we turn to CGL *residual vectors*. Combining (6) and (7), we obtain

$$\frac{1}{\|r_l^L\|^2} r_l^L = \frac{1}{\|r_l^M\|^2} r_l^M - \frac{1}{\|r_{l-1}^M\|^2} r_{l-1}^M \quad (15)$$

for the exact arithmetic. Since we assume that (4) and (5) are solved with a negligible error, the first relation in (6) becomes in finite-precision computation

$$\bar{c}_k^2 \bar{r}_k^L = \bar{r}_k^M - \bar{s}_k^2 \bar{r}_{k-1}^M + \bar{F}_k \bar{y}_k^M. \quad (16)$$

Using [6], \bar{s}_k and \bar{c}_k can be expressed via the residual norms obtained from the exact computation with the extended matrix \hat{A} in the same way as in the second equation of (6). Due to [2, Lemmas 4 and 8], these norms are approximately equal to those obtained by finite-precision computation. Using the relation (16) recursively, applying the results by Cullum and Greenbaum, and omitting the round-off terms, we obtain

$$\sum_{j=k_{l-1}+1}^{k_l} \frac{1}{\|\bar{r}_j^L\|^2} \bar{r}_j^L \approx \frac{1}{\|\bar{r}_{k_l}^M\|^2} \bar{r}_{k_l}^M - \frac{1}{\|\bar{r}_{k_{l-1}}^M\|^2} \bar{r}_{k_{l-1}}^M, \quad (17)$$

with the error of approximation determined by the round-off terms in [2]. Combining (15) and (17) while taking into account assumptions (9) and (10) on MINRES gives

$$\frac{1}{\|r_l^L\|^2} r_l^L \approx \sum_{j=k_{l-1}+1}^{k_l} \frac{1}{\|\bar{r}_j^L\|^2} \bar{r}_j^L.$$

Using the relation (14) we finally get

$$r_l^L \approx \frac{1}{\sum_{j=k_{l-1}+1}^{k_l} \frac{1}{\|\bar{r}_j^L\|^2}} \cdot \sum_{j=k_{l-1}+1}^{k_l} \frac{1}{\|\bar{r}_j^L\|^2} \bar{r}_j^L. \quad (18)$$

Thus the residual vectors from finite-precision computation have to be aggregated over the same iterations as their norms.

3.2 Lanczos Vectors

Recall that in exact arithmetic, the residual of CGL is a multiple of the subsequent Lanczos vector, i.e., $r_l^L = (-1)^l \|r_l^L\| v_{l+1}$. In finite-precision computation, (11) gives

$$\bar{r}_k^L \approx (-1)^k \|\bar{r}_k^L\| \bar{v}_{k+1}.$$

This, together with (14) and (18) yields

$$v_{l+1} \approx \frac{(-1)^l}{\sqrt{\sum_{j=k_{l-1}+1}^{k_l} \frac{1}{\|\bar{r}_j^L\|^2}}} \cdot \sum_{j=k_{l-1}+1}^{k_l} \frac{(-1)^j}{\|\bar{r}_j^L\|} \bar{v}_{j+1}. \quad (19)$$

Thus if the exact MINRES does not stagnate, assuming (9) and (10) the exact Lanczos vectors can be *approximated by a linear combination* of several consecutive Lanczos vectors from finite-precision computation. The derivation above does not rely on the orthogonality among the vectors $\bar{v}_{k_{l-1}+2}, \dots, \bar{v}_{k_l+1}$.

3.3 Influence of Exact MINRES Stagnation

Now consider a plateau in exact MINRES convergence curve, i.e., $\|r_l^M\|/\|r_{l-1}^M\| \approx 1$ for some l . This can be caused (among others) by presence of a *tight cluster of eigenvalues* in the spectrum of A . Due to (7), we simultaneously observe a peak in the exact CGL residual norms. In this case, (12) and (13) may not hold and some of the CGL residual norms in exact arithmetic may not have a finite-precision counterpart of the form (14).

If the exact MINRES does not stagnate till the last iteration, we can proceed p_l iterations forward to achieve

$$\|r_{l+p_l}^M\|/\|r_{l-1}^M\| \ll 1. \quad (20)$$

Then, the approximations (12) and (13) become valid again for $\bar{r}_{k_{l-1}}^M$ and $\bar{r}_{k_{l+p_l}}^M$. Using (8), we conclude that the norms can be compared as

$$\begin{aligned} \frac{1}{\sqrt{\sum_{j=l}^{l+p_l} 1/\|r_j^L\|^2}} &= \frac{\|r_{l+p_l}^M\|}{\sqrt{1 - (\|r_{l+p_l}^M\|/\|r_{l-1}^M\|)^2}} \\ &\approx \frac{\|\bar{r}_{k_{l+p_l}}^M\|}{\sqrt{1 - (\|\bar{r}_{k_{l+p_l}}^M\|/\|\bar{r}_{k_{l-1}}^M\|)^2}} \approx \frac{1}{\sqrt{\sum_{j=k_{l-1}+1}^{k_{l+p_l}} 1/\|\bar{r}_j^L\|^2}}, \end{aligned} \quad (21)$$

i.e., both finite-precision and exact-arithmetic norms are aggregated over consecutive iterations. Other ways of comparison are also possible.

4 Construction of the Subsequence $\{k_l\}_{l=1}^m$

In this section, we aim at finding the subsequence of iterations $\{k_l\}_{l=1}^m$. We discuss several possible approaches, where the first one was in a similar form considered previously in [5, 12].

Numerical Rank of the Computed Subspace. Focusing on the rank-deficiency of the matrix \bar{V}_k of computed Lanczos vectors, the subsequence can be determined as

$$k_l^{\text{rank}} \equiv \max\{k \mid \text{num_rank}(\bar{V}_k) = l\}.$$

The definition of numerical rank is generally a subtle issue and the resulting subsequence is dependent on its choice. Denoting $\bar{\sigma}_i$ the singular values of \bar{V}_k , we use

$$\text{num_rank}(\bar{V}_k) \equiv \{\#\bar{\sigma}_i \mid \bar{\sigma}_i > \tau\}. \quad (22)$$

The choice of the truncation parameter τ should reflect the fact that the exact matrix V_j has orthonormal columns, i.e., its singular values equal 1. We set in our experiments $\tau = 0.1$. Alternatively, numerical rank could be based, e.g., on finding the maximum gap between the singular values of \bar{V}_k .

Explicit Fitting of the Convergence Curves. Focusing on the delay of convergence, the subsequence can be found by explicit fitting of the quantities minimized over the Krylov subspace. In this way we find optimal subsequence with respect to one of the two assumptions in (9).

Fitting the CGL Convergence Curves:

$$k_l^L = \underset{k}{\text{argmin}} \left| \|x_l^L - x\|_A - \|\bar{x}_k^L - x\|_A \right|. \quad (23)$$

Fitting the MINRES Convergence Curves:

$$k_l^M = \underset{k}{\text{argmin}} \left| \|r_l^M\| - \|\bar{r}_k^M\| \right|. \quad (24)$$

Note that the applicability of the approaches depends on the entities available. While (22) requires only the matrix \bar{V}_k , (24) requires r_l^M , i.e., exact computation has to be simulated. Approach (23) requires in addition the true solution x , which is not available in practical computations. Since CGL and MINRES are closely related, it is natural to expect that (23) and (24) provide similar subsequences. Fitting other entities, such as Ritz values, would theoretically also be possible. Note that (22) gives a strictly increasing subsequence $\{k_l\}_{l=1}^m$, which is not necessary the case for the other approaches and which becomes important especially for problems with stagnation in the exact convergence curves; see Sect. 5.

5 Numerical Experiments

Now we compare the approaches to construction of $\{k_l\}$, we discuss the assumptions (9) and (10), and illustrate the results obtained for the CGL residuals and Lanczos vectors. Exact arithmetic is simulated by incorporating double reorthogonalization of the computed Lanczos vectors into the Lanczos process. It was shown in [18] that such algorithm is backward stable, i.e., it represents an exact Lanczos process for a nearby problem. The projected problems (4) and (5) are solved by the MATLAB function `mldivide`. Computations are stopped before the maximum attainable accuracy is reached. Experiments are performed in MATLAB R2015b.

We consider several test matrices from the Harwell-Boeing Collection [3] and the test matrix `strakos` introduced in [23], with parameters $n = 100$, $\lambda_{\min} = 0.1$, $\lambda_{\max} = 1000$, and $\gamma = 0.7$. The properties of the matrices are summarized in Table 2. For all matrices we choose $b = [1, \dots, 1]^T$.

Fulfilling the Assumptions. In order to apply the results of Sect. 3, we first need the subsequence $\{k_l\}$ fulfilling (9) and (10). Figure 2 shows the subsequences constructed by approaches proposed in Sect. 4 together with the evolution of the singular values of the matrix \bar{V}_k for problems `strakos` and `bcsstk01`. All three subsequences follow the edge of nonzero singular values throughout the whole computation. From the differences between $\{k_l^L\}$ and $\{k_l^M\}$ optimal with respect to the two convergence curves, we conclude that there is no $\{k_l\}$ optimal in all considered aspects. The plots in Fig. 3 (left) show the match between the exact

Table 2. Properties of the test matrices.

Problem	n	$\text{nnz}(A)$	$\ A\ $	$\kappa(A)$
<code>strakos</code>	100	100	1×10^4	1×10^5
<code>bcsstk01</code>	48	400	3×10^9	1.6×10^6
<code>bcsstk04</code>	132	3648	9.6×10^6	5.6×10^6
<code>nos7</code>	729	4617	9.9×10^6	4.1×10^9

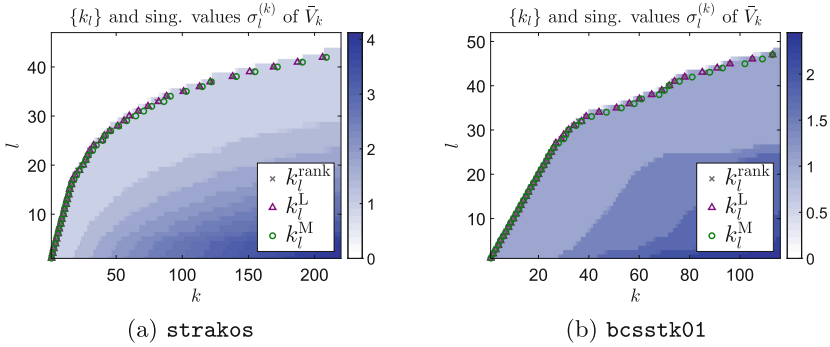


Fig. 2. The singular values of the computed matrix \bar{V}_k together with the subsequences $\{k_l\}$ constructed using the three approaches from Sect. 4.

and finite-precision convergence curves shifted using $\{k_l\}$. We see a nice overlap of the convergence curves. In the right plots, we observe that

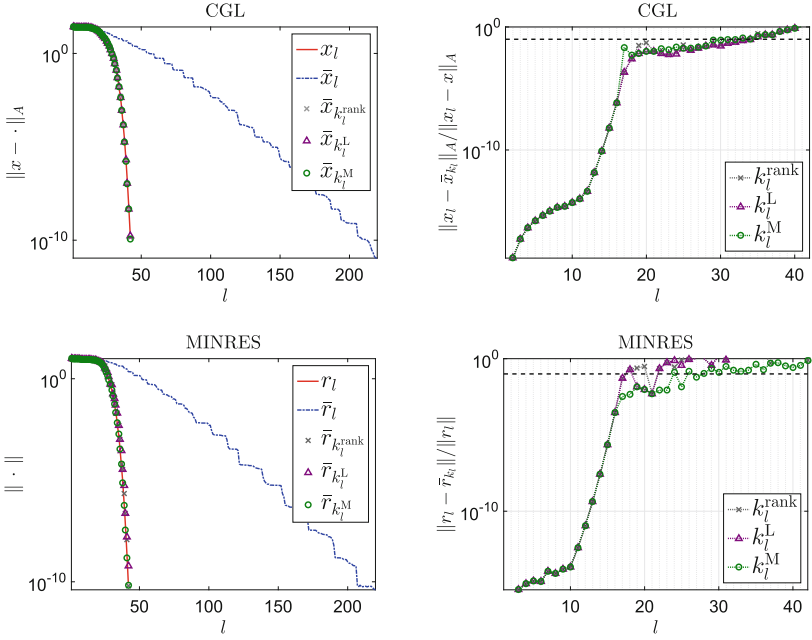
$$\|x_l^L - \bar{x}_{k_l}^L\|_A \ll \|x_l^L - x\|_A \quad \text{and} \quad \|r_l^M - \bar{r}_{k_l}^M\| \ll \|r_l^M\|$$

holds for most iterations, fulfilling sufficiently the assumption (10). From the experiments, the subsequence $\{k_l^M\}$ obtained by optimal fitting of the MINRES residual norms seems to provide the best results with respect to (9) and (10) and therefore it is used in the following experiments.

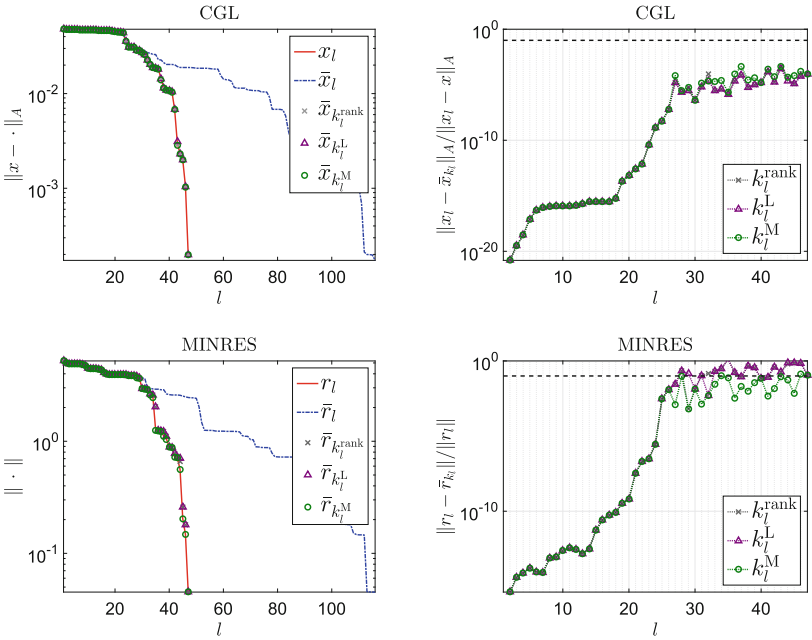
CGL Residuals and Lanczos Vectors. Now we verify (14), (18), and (19) derived in Sect. 3. Figure 4 (left) gives the comparison between the exact and finite-precision residual norms aggregated using (14) and shifted as in Fig. 3. In both cases, we observe very good match between the exact and aggregated finite-precision CGL convergence curve. For problem *strakos*, the approximation error of (18) for the residual vectors depicted in Fig. 4a (right) is essentially determined by the approximation error of the MINRES residual vectors, shown in Fig. 3a (right). For *bcsstk01*, the approximation is for the CGL residuals slightly worse than for the MINRES residuals, compare Figs. 3b and 4b. This is caused by the fact that in several iterations MINRES almost stagnates. A similar plot for the Lanczos vectors using (19) is provided in Fig. 5. Due to the relation (11), the approximation error is here similar to the approximation error of the CGL residuals.

Stagnation in MINRES Convergence. For real problems, the exact MINRES residual norm may not decrease sufficiently in each iteration, and severe oscillation may appear in the norm of the exact CGL residual.³ Figure 6 shows results for two test problems of such type. Although (9) is satisfied, (14) does

³ In such a case, approach (24) tends to construct subsequences for which $k_l = k_{l-1}$ may hold for some l .

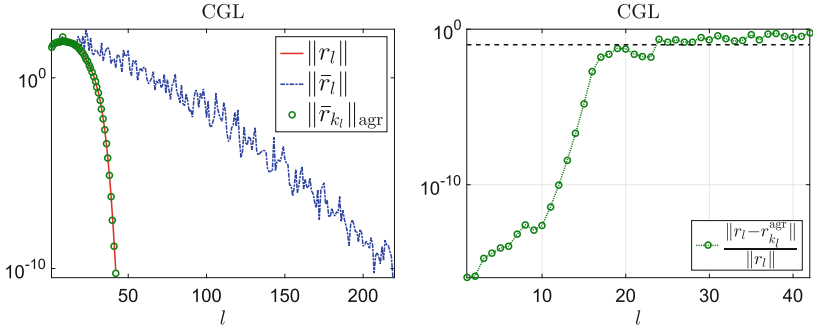


(a) strakos

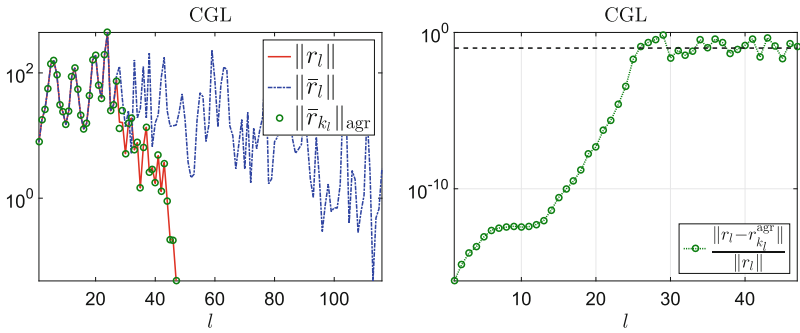


(b) bcsstk01

Fig. 3. Fulfillment of the assumptions (9) and (10) for two test problems. Left: The match between the exact convergence curve and the finite-precision convergence curve shifted using various subsequences $\{k_l\}$. Right: The match between the vectors themselves.

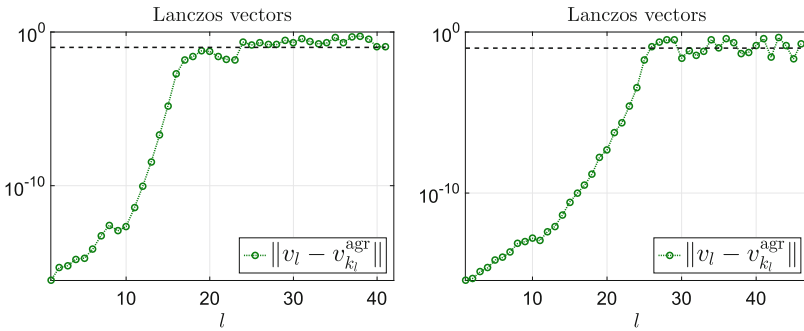


(a) strakos



(b) bcsstk01

Fig. 4. Left: the exact and finite-precision CGL residual norms; $\|r_{k_l}\|_{agr}$ denotes the right-hand side of (14). Right: relative difference between the exact and aggregated finite-precision residuals; $r_{k_l}^{agr}$ denotes the right-hand side of (18).



(a) strakos

(b) bcsstk01

Fig. 5. Difference between the exact and aggregated finite-precision Lanczos vectors; $v_{k_l}^{agr}$ denotes the right-hand side of (19).

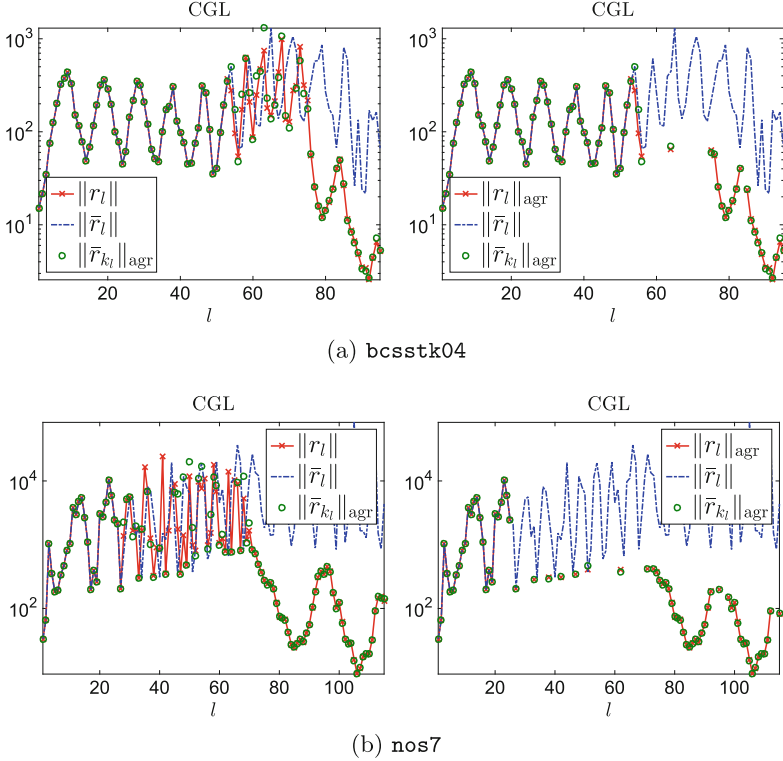


Fig. 6. CGL residuals compared as in (14) and (21) (plotted against $l+p_l$), respectively. Stagnation tolerance η is set to 0.01 (*bcsstk04*) and 0.002 (*nos7*).

not hold, see Fig. 6 (left). In order to apply the alternative formula derived in Sect. 3.3, we need to detect iterations with stagnation. Based on [2, Theorem 4], we suggest the criterion

$$1 - (\|r_l^M\| / \|r_{l-1}^M\|)^2 < \eta, \quad (25)$$

with η much smaller than the relative approximation error $|\|r_l^M\| - \|\bar{r}_{k_l}^M\|| / \|r_l^M\|$. We proceed as follows. If the stagnation criterion (25) is satisfied, we substitute r_l^M by r_{l+1}^M and continue until we get sufficient decrease (20). Both sides of (21) are then plotted against $l + p_l$. We use every residual norm only once, i.e., in the next step of comparison we start with $l_{new} \leftarrow l + p_l + 1$. The larger the value η , the more iterations are aggregated and the sparser plots we get. Results obtained using this aggregation scheme are shown in Fig. 6 (right).

6 Conclusion

We have demonstrated that in many cases quantities minimized in exact MINRES and CGL computation can be compared directly to their selected

finite-precision counterparts for the same linear algebraic problem. We have proposed three approaches for determination of the subsequence of relevant finite-precision iterations – based on the numerical rank of the computed Lanczos matrix or on optimal fitting of the convergence curves. We have shown that entities whose size does not decay monotonically can not be compared directly. However, we have derived formulas relating the exact CGL residuals (and their norms) and the exact Lanczos vectors to vectors obtained in the finite-precision aggregated over the intermediate iterations. We have explained limitations of this approach for problems, where the exact MINRES method (nearly) stagnates and proposed an alternative way of comparison based on more general aggregation scheme. The results have been supported by experiments on standard test problems.

Acknowledgment. Research supported by the Grant Agency of Charles University (GAUK 196216) and by the Grant Agency of the Czech Republic (17-04150J).

References

1. Carpraux, J.F., Godunov, S.K., Kuznetsov, S.V.: Condition number of the Krylov bases and subspaces. *Linear Algebra Appl.* **248**, 137–160 (1996)
2. Cullum, J., Greenbaum, A.: Relations between Galerkin and norm-minimizing iterative methods for solving linear systems. *SIAM J. Matrix Anal. Appl.* **17**(2), 223–247 (1996)
3. Duff, I.S., Grimes, R.G., Lewis, J.G.: Users’ guide for the Harwell-Boeing sparse matrix collection (1992)
4. Fong, D.C.L., Saunders, M.A.: CG versus MINRES: an empirical comparison. *SQU J. Sci.* **17**(1), 44–62 (2012)
5. Gergelits, T.: Analysis of Krylov subspace methods. Master’s thesis, Charles University in Prague (2013)
6. Greenbaum, A.: Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.* **113**, 7–63 (1989)
7. Greenbaum, A., Strakoš, Z.: Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.* **13**(1), 121–137 (1992)
8. Hansen, P.C.: Rank-Deficient and Discrete Ill-Posed Problems. Society for Industrial and Applied Mathematics (1998)
9. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.* **49**, 409–436 (1952)
10. Kuznetsov, S.V.: Perturbation bounds of the Krylov bases and associated Hessenberg forms. *Linear Algebra Appl.* **265**, 1–28 (1997)
11. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bureau Stand.* **45**, 255–282 (1950)
12. Liesen, J., Strakoš, Z.: Krylov Subspace Methods: Principles and Analysis. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford (2013)
13. Meurant, G., Strakoš, Z.: The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica* **15**, 471–542 (2006)

14. O'Leary, D.P., Strakoš, Z., Tichý, P.: On sensitivity of Gauss-Christoffel quadrature. *Numer. Math.* **107**(1), 147–174 (2007)
15. Paige, C.C.: The computation of eigenvalues and eigenvectors of very large sparse matrices. Ph.D. thesis, London University (1971)
16. Paige, C.C.: Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra Appl.* **34**, 235–258 (1980)
17. Paige, C.C., Saunders, M.A.: Solutions of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* **12**(4), 617–629 (1975)
18. Paige, C.C.: Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *IMA J. Appl. Math.* **18**(3), 341–349 (1976)
19. Paige, C.C.: An augmented stability result for the Lanczos Hermitian matrix tridiagonalization process. *SIAM J. Matrix Anal. Appl.* **31**(5), 2347–2359 (2010)
20. Paige, C.C., Van Dooren, P.: Sensitivity analysis of the Lanczos reduction. *Linear Algebra Appl.* **6**(1), 29–50 (1999)
21. Parlett, B.N., Scott, D.S.: The Lanczos algorithm with selective orthogonalization. *Math. Comput.* **33**(145), 217–238 (1979)
22. Simon, H.D.: Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra Appl.* **61**, 101–131 (1984)
23. Strakoš, Z.: On the real convergence rate of the conjugate gradient method. *Linear Algebra Appl.* **154–156**, 535–549 (1991)



Software Tool for Cranial Orthosis Design

Milan Jaros, Tomas Karasek, Petr Strakos, and Alena Vasatova^(✉)

IT4Innovations, VSB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic
{milan.jaros,tomas.karasek,petr.strakos,alena.vasatova}@vsb.cz

Abstract. Cranial orthoses are used to correct an abnormal children head shape, and such they have to be designed individually. Customization of those orthoses is currently fully manual task. A software tool should make this process semi-automatic with only small intervention from the user and speed up the whole process. In the future, this tool will be part of the process chain from 3D scanning of the patient head to the 3D printing of the final product. This will allow to produce the orthosis anywhere, without necessity to have expensive devices on one place. For high quality of 3D printing, 3D computer models with high-resolution meshes must be used. We have started development of our tool by rapid testing of methodology. For this purpose we used open source software Blender. Although Blender's functions we used are more robust, they are also unnecessary computationally more expensive. For this reason we have implemented the necessary transformation functions using radial basis functions (RBF) which can be easily modified to include rigid body movements.

Keywords: Cranial orthosis · Software tool · Blender
Radial basis function · Rigid parts

1 Introduction

Treatment of skull deformities of children by cranial orthosis, see Fig. 1, has been adopted by pediatricians and has been increasingly used since it was first documented in 1979, by [3]. Orthosis could be used to fix positional skull deformities such as plagiocephaly, brachycephaly and scaphocephaly, see Fig. 2. These asymmetric head shapes are caused by external forces applied to an infant's malleable skull during prenatal/postnatal development or at birth (e.g. multiple-birth infants, sleeping position or premature birth). Majority of skull deformities present at birth improve spontaneously approximately six weeks after the delivery. Therefore, the most important deformities are those that develop throughout the first months of child's life.

The number of cranial deformities has increased considerably since international efforts of pediatricians recommended the sleeping supine position as a strategy to reduce sudden death syndrome of the newborns. Keeping infant too long in one position (in car seats, baby carriers and other accessories) has also

helped to decrease the time children remain in the prone positions, but on the other hand increases the number of children who can develop cranial deformities. More information can be found in [9, 10].



Fig. 1. Example of the cranial orthosis.



Fig. 2. Examples of plagiocephaly, brachycephaly and scaphocephaly.

When manufactured, every orthosis has to be designed and customized individually, which is currently a fully manual task. Our software tool, which we have developed, requires only small intervention from a medical technician. This speeds up the orthosis production and enables individual parts of the process from 3D scanning to 3D printing to be held anywhere in the world without expensive devices on one place.

To manufacture cranial orthosis by the method of 3D printing, 3D computer models with high-resolution meshes must be used to achieve a product of high quality. To be able to modify the 3D models, large system of linear equations must be solved. This system grows enormously with the size of the used 3D models which requires the use of efficient and fast solvers.

For purpose of rapid prototyping and to test proposed methodology for modification of 3D models of cranial orthosis based on mesh morphing we have used open source 3D graphics and animation software Blender. Our first implementation of the tool was based on modification and enhancement of Blender's two specific modifiers, MeshDeform and ShrinkWrap. Existing methods and functions implemented in MeshDeform were parallelized to improve their performance. However, even parallel version of the MeshDeform did not provide satisfactory performance, and we decided to abandon the functionality of MeshDeform and decided to implement our own transformation procedure using Radial Basis Function (RBF).

RBFs have become a well-established tool to interpolate scattered data. They are used as a transformation functions to interpolate the displacements of the boundary nodes of the mesh to the inner domain. The method requires solving a relatively small system of equations. But the relativity is caused by comparing the number of boundary nodes to the whole mesh. With huge meshes even the boundary becomes large. The implementation of the method is quite simple, even for the 3D applications, because no grid-connectivity information is needed. Parallelization of the concept is also straightforward.

2 Cranial Orthosis Model

During the customization of the generic cranial orthosis, the model is modified to individual patient based on idealized 3D scan of the head, see Fig. 3. This idealized scan is based on the original deformed scan, which is transformed to be as close to ideal head shape as possible. The transformation has to meet certain rules. For example, the circumference of the head can change by maximally 2–3 cm (according the age), the width of the head can not change, the head has to be symmetric etc. This transformation is also done fully manually by technician in CAD like software and it is planned to be automatized. The scan of head is cropped afterwards by outlines specified by the medical technician.

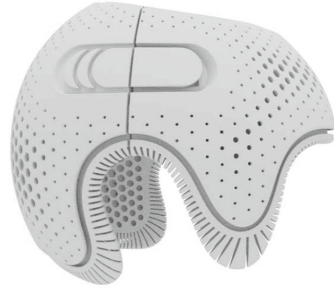
The model itself is composed of two parts: a helmet and a locking mechanism, see Fig. 4. Each part is represented by high-resolution mesh.



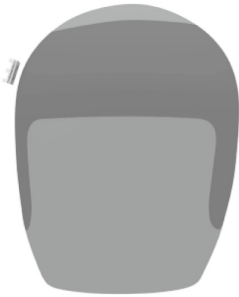
Fig. 3. Input data: original and cropped 3D scan of patient head.



(a) Front view of the model.



(b) Side view of the model



(c) Rigid part (the locking mechanism) from the front.



(d) Rigid part (the locking mechanism) from the side.

Fig. 4. Input data: cranial orthosis model.

Beside that, we also need auxiliary meshes, so-called cages, see Fig. 5 which purpose is to serve as control points for computing RBF (main cage) and to compute rigid body movement of the locking mechanism (the secondary cage).

The goal of the transformation is a non-rigid deformation of the orthosis body to fit the cropped scan and a rigid transformation of the locking mechanism to its specified position. The locking mechanism can only translate and rotate to preserve its functionality.

3 Morphing Algorithm

At first, we used modification and enhancement of Blender for rapid testing of the proposed methodology. We bound the model mesh with the control cage using MeshDeform modifier, then we shrank the control cage to the scans by Shrinkwrap modifier (the function is based on finding closest point on surface

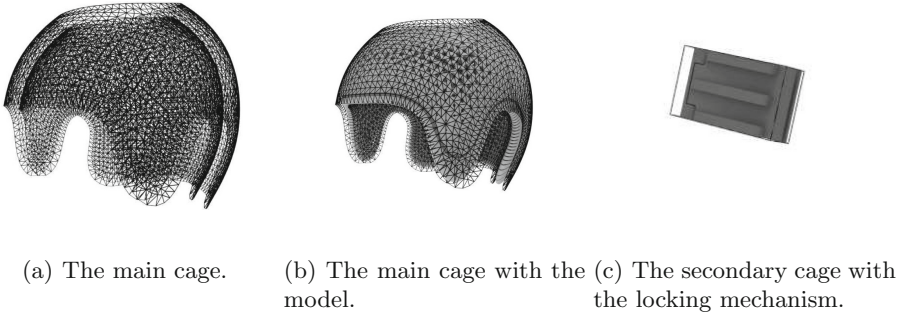


Fig. 5. Input data: cages.

with given offset). MeshDeform modifier automatically deforms the model mesh according to the cage deformation, based on [5]. Since this step is computationally extensive MeshDeform modifier was parallelized using MPI technology to improve its speed and to allow handling of large data sets.

Although, parallel version of the MeshDeform modifier gives good results, it is still computationally extensive and we have replaced it completely by transformation using radial basis function (RBF), which is computationally less expensive. Transformation by RBF also allows easy incorporation of the rigid parts, which MeshDeform modifier cannot implicitly do.

The final version of the algorithm can be described as follows:

1. Read input Data.
2. Shrink main cage on cropped scan of the head.
3. Project rigid cage to main cage, find principal axes.
4. Identify local rigid body movement from shrunk and original principal axes.
5. Compute main elastic transformation based on RBF with rigid body movement.
6. Transform given parts.
7. Save results.

Graphical representation of the algorithm in form of simplified example is depicted in Fig. 6.

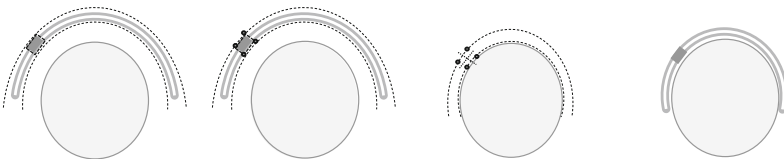


Fig. 6. Simplified example; from left to right - input data, projected rigid cage on the main cage before shrink, Shrunk cages on the scan of the head with principal axes of the locking mechanism, final result

4 Radial Basis Function

Basic principle of RBF could be described in such a way that we are looking for a function $\varphi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ that exactly interpolates the displacement $u_j \in \mathbb{R}^3$ of given control points $x_j \in \mathbb{R}^3$, $j = 1, \dots, m$ and smoothly interpolates this displacement into the mesh. RBF are well known to be suitable for solving this type of problem [2]. The displacement function is then represented as a linear combination of polynomial corrections p_i and a linear combination of radially symmetric kernels $\rho_{x_j}(x) = \rho(\|x - x_j\|_{\mathbb{R}^3})$ located at centers x_j , independent for each dimension $l = 1, 2, 3$:

$$\varphi_l(x) = \sum_{i=1}^{n_p} \beta_{l,i} p_i(x) + \sum_{j=1}^m \theta_{l,j} \rho_{x_j}(x),$$

where n_p is the dimension of used polynomials. We choose linear, thus $n_p = 4$.

The coefficients $\beta_l \in \mathbb{R}^{n_p}$ and $\theta_l \in \mathbb{R}^m$ are defined by

$$\begin{pmatrix} K & B^\top \\ B & 0 \end{pmatrix} \begin{pmatrix} \theta_l \\ \beta_l \end{pmatrix} = \begin{pmatrix} u_l \\ 0 \end{pmatrix}, \quad (1)$$

where

$$K := (\rho(\|x_j - x_k\|_{\mathbb{R}^3}))_{j,k=1,\dots,m} \in \mathbb{R}^{m \times m},$$

$$B := (p_i(x_k))_{\substack{i=1,\dots,n_p \\ k=1,\dots,m}} \in \mathbb{R}^{n_p \times m}$$

and

$$u_l := (u_{1,l}, \dots, u_{j,l}) \in \mathbb{R}^m.$$

The choice of the kernel function $\rho(r)$ has a significant influence on the result of the deformation. We use triharmonic [12]

$$\rho(r) = r^3$$

or thin plate spline (TPS) [7]

$$\rho(r) := \frac{\Gamma(3/2 - q)}{2^{2q} \pi^{3/2} (q-1)!} r^{2q-3}.$$

To be able to handle rigid transformation, we incorporated techniques from [6], where certain parts of image are handled as rigid.

Lets have n rigid objects with predefined linear transformations denoted by matrices $L_q \in \mathbb{R}^{3 \times n_p}$, $q = 1, \dots, n$. Instead of discrete distance transformations used in image processing we compute distance between given point and closest point of the mesh using octree-based spatial algorithm, which can search the mesh to quickly locate the point on the mesh face [11]. $\mathcal{D}_0(x)$ represents the distance from a point x to the closest object and $\mathcal{D}_q(x)$, $q = 1, \dots, n$ to q -th object. These functions ensure that the non-linear part of the transformation and the linear transformations L_q , $q \neq r$ tend to zero as we move towards the r -th rigid object.

The coefficients β_l for polynomial corrections are replaced by weighted sum of the individual object linear transformations

$$\mathcal{L}(x) = \sum_{q=1}^n w_q(x)L_q,$$

where

$$w_q(x) = \frac{v_q(x)}{\sum_{r=1}^n v_r(x)}, \quad v_q(x) = \frac{1}{\mathcal{D}_q(x)^\mu}$$

The kernels are also weighted

$$\tilde{\rho}_{x_j}(x) = |\mathcal{D}_0(x)| |\mathcal{D}_0(x_j)| \rho_{x_j}(x).$$

Thus we get transformation function

$$\varphi_l(x) = \sum_{i=1}^{n_p} \mathcal{L}(x)p_i(x) + \sum_{j=1}^m \theta_{l,j} \tilde{\rho}_{x_j}(x),$$

and Eq. (1) can be rewritten as

$$K\theta_l + T = u_l, \quad l = 1, 2, 3,$$

where

$$T = \begin{pmatrix} p(x_1)^T \mathcal{L}(x_1)^T \\ p(x_2)^T \mathcal{L}(x_2)^T \\ \vdots \\ p(x_m)^T \mathcal{L}(x_m)^T \end{pmatrix}.$$

Linear transformation matrix L_q consists of 12 coefficients, and we need 4 points in space to determine them. The easy way is to take these points from

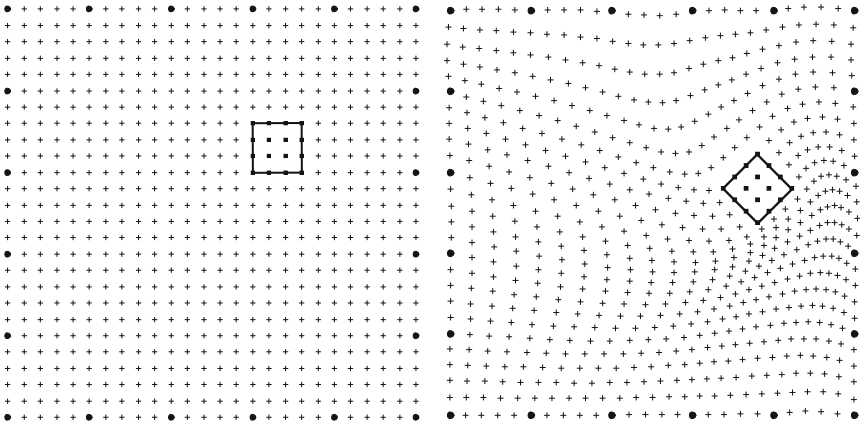


Fig. 7. Simplified 2D example.

principal axes of the rigid cage, Principal axes are obtained through the principal component analysis (PCA). PCA is a statistical method used to estimate the necessary information from the measured data [4,8]. We are able to determine the axes through the use of eigenvalues and eigenvectors of the covariance matrix consisting of a small group of neighboring points.

In Fig. 7 we show the simplified 2D example of rigid object transformation as a part of a more complex mesh. The cage points are marked by circles and the rigid object is marked by square.

5 Implementation

For our implementation we use the VTK library to work with 3D geometry and mesh models. To solve large systems of linear equations we use the Intel MKL library. This library offers direct solvers and it is optimized for high performance.

Beside the time necessary to obtain the solution of the unknowns from the linear system of equations in (1), another time demanding operation in the designing process is the transformation of the orthosis model to the final shape. The model consists of millions of polygons, therefore to spatially transform each of them we have parallelized the operation by utilizing OpenMP technology.

In the following we provide the pseudo code of the whole designing process. It goes as such:

1. Read mesh files (Wavefront .obj) - see Figs. 4 and 5.
 - (a) the outer part of the main cage for helmet.
 - (b) the inner part of the main cage for helmet.
 - (c) the secondary cage for the rigid part.
 - (d) the model of the helmet.
 - (e) the model of the rigid part.
 - (f) the model of the patient's head.
2. Shrink cages.
 - (a) shrink the outer part of the main cage (1a) to the head (1f) and offset it.
 - (b) shrink the inner part of the main cage (1b) to the head (1f).
 - (c) shrink the cage of the rigid part (1c) to the inner part of the original main cage (1b).
 - (d) shrink the cage of the rigid part (1c) to the shrunk inner part of the main cage (2b).
- 3 Merge cages.
 - (a) merge inner (1b) and outer (1a) part of the original main cage.
 - (b) merge inner (2b) and outer (2a) part of the shrunk main cage.
4. Estimate the linear transformation of the rigid part between (2c) and (2d).
 - (a) compute the coordination system of the (2c) using PCA.
 - (b) compute the coordination system of the (2d) using PCA.
 - (c) estimate the linear transformation from the solution of the problem $Ax = b$ which is based on the results of step (4a) and (4b).
5. Compute main elastic transformation based on RBF with rigid body movement.

- (a) utilize the above results to assemble another $Ax = b$ problem, detailed description is given in Sect. 4.
 - (b) obtain solution by applying direct solver for dense matrices (Intel MKL).
6. Transform model of the orthosis (1d) and (1e).
- (a) utilize solution x from (5b), detailed description is given in Sect. 4.
7. Save results to the OBJ files.

6 Results

To establish performance of our implementation we have performed measurements focusing on algorithm speed and its possible speed-up by utilizing OpenMP framework on multiple cores. We have also measured computational demands of the algorithm based on the model size.

For all the tests, configuration of the RBF and the solver was as follows:

- Thin plate spline (TPS) has been used as a kernel function $\rho(r)$, see Sect. 4.
- Bunch-Kaufman factorization of a symmetric matrix using packed storage has been used as a solver.

6.1 Algorithm Speed and Possible Speed-Up by OpenMP on Multiple CPU Cores

In Table 1 we provide time measurements of different parts of the implemented algorithm. As can be seen, the most time consuming part is the transformation of the model to the desired shape and size. Fortunately, this step can be easily parallelized and more computational resources can be used. Effect of such parallelization is documented in Fig. 8. The problem description that specifies this particular measurement is following:

- Number of vertices from step (3a) of pseudocode: 5452
- Number of vertices from step (3b) of pseudocode: 5452
- Number of vertices from step (1d) of pseudocode: 612546
- Number of vertices from step (1e) of pseudocode: 4818
- System size (size of A from step (5)): 5452×5452

Table 1. Computation times within different phases of the algorithm while utilizing OpenMP on 1 to 24 CPU cores

CPU cores [-]	1	2	4	8	16	24
Shrink [s] (2)	0.80	0.75	0.73	0.72	0.73	0.72
Prepare [s] (3)–(4)	85.36	46.44	23.32	12.30	7.96	5.63
Solve [s] (5)	3.94	2.65	2.18	1.96	1.88	1.88
Transform [s] (6)	248.07	124.19	62.29	32.56	15.75	10.87
Total [s] (2)–(6)	338.16	174.03	88.51	47.54	26.33	19.09

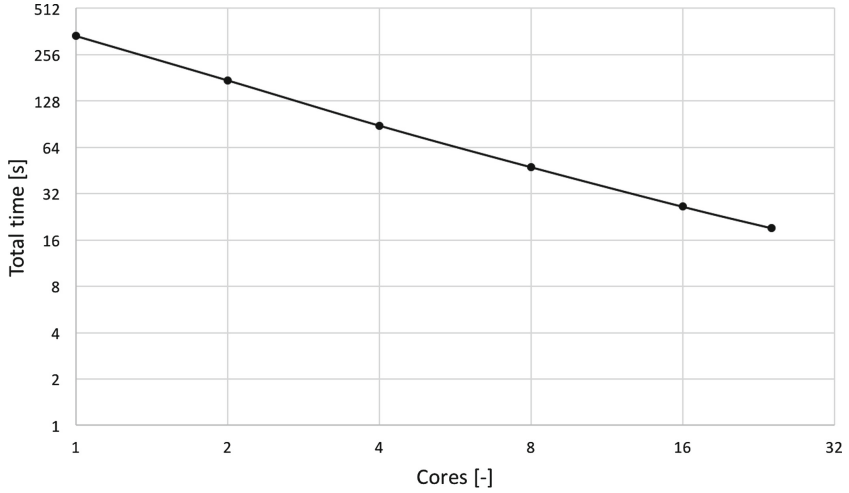


Fig. 8. Total time of the algorithm while utilizing increasing number of CPU cores

Table 2. Comparison of computation times for different sizes of the model

Model size [vertices] (1d)	243897	494826	1033902
Shrink [s] (2)	0.24	0.72	2.61
Prepare [s] (3)–(4)	1.65	4.83	18.54
Solve [s] (5)	0.44	1.89	10.19
Transform [s] (6)	2.76	8.79	38.91
Total [s] (2)–(6)	5.09	16.23	70.25

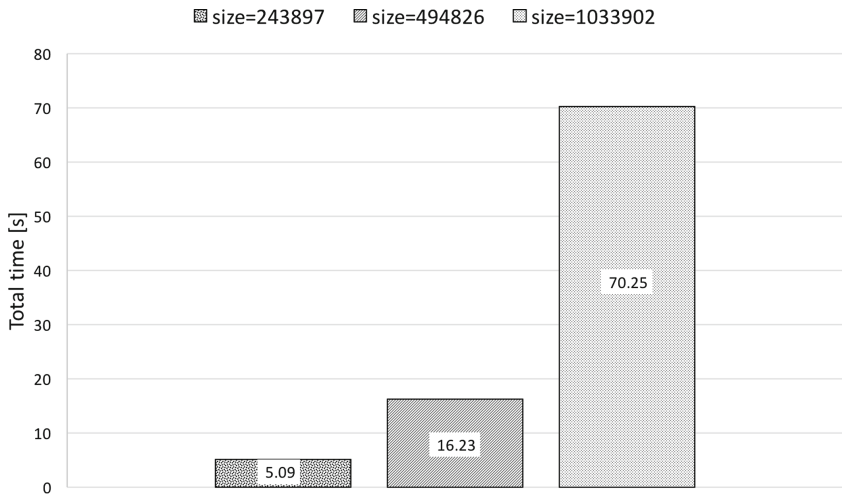


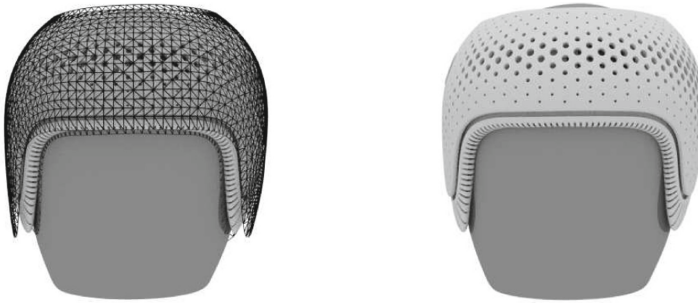
Fig. 9. Total computation time of the algorithm for different sizes of the model

6.2 Computation Times for Different Sizes of the Model

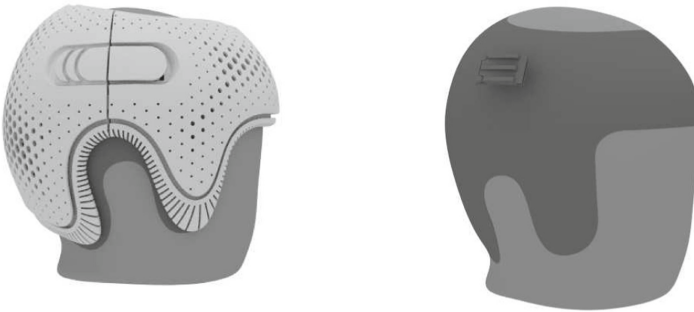
To evaluate algorithm behavior on different sizes of the problem we have tested it on three different models, see Table 2. Within the test, all available resources at computing nodes have been used.

For tests we have always used one node of Salomon supercomputer [1] where each node is equipped with powerful x86-64 computer consisting of 24 cores (2 x Intel Xeon E5-2680v3, 2.5 GHz, 12 cores) and 128 GB RAM.

Total computation times of the algorithm obtained on the three different sizes of the model are well depicted in Fig. 9. The largest model fitting to the desired size and shape of the head is shown in Fig. 10. This particular model also fulfills the required quality demands for the 3D print, which is about 1 000 000 vertices.



(a) Transformed model with the cage and (b) Transformed model with the scan from the front.



(c) Transformed model with the scan from the side. (d) Transformed locking mechanism with the scan from the side.

Fig. 10. Transformed model of the cranial orthosis to specific size and shape of the head

7 Conclusion

We have provided suitable method for morphing of a complex 3D model of a cranial orthosis. Method is based on Radial Basis Functions (RBFs) and it can adjust the size and shape of the orthosis from the general one to the customized one based on the scan of the patient's head. The method can perform non-rigid transformation of the orthosis together with rigid transformation of its specific part that has to preserve the size and the shape for its proper functionality. Our method can therefore provide virtual designing process of the orthosis to specific patient's needs without the necessity to create any physical prototypes that would need to be manually adjusted. Our method has been also speeded up by OpenMP framework and it can efficiently utilize available resources of a computer.

Acknowledgement. This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center - LM2015070". This work is the main objective of contractual research conducted in collaboration with ING corporation spol. s.r.o.

References

1. Hardware Overview - IT4Innovations Documentation, August 2017. <https://docs.it4i.cz/salomon/hardware-overview/>
2. de Boer, A., van der Schoot, M., Bijl, H.: Mesh deformation based on radial basis function interpolation. *Comput. Struct.* **85**(11), 784–795 (2007). <https://doi.org/10.1016/j.compstruc.2007.01.013>. <http://www.sciencedirect.com/science/article/pii/S0045794907000223>. Fourth MIT Conference on Computational Fluid and Solid Mechanics
3. Clarren, S.K.: Plagiocephaly and torticollis: etiology, natural history, and helmet treatment. *J. Pediatr.* **98**(1), 92–95 (1981). <http://www.sciencedirect.com/science/article/pii/S0022347681805495>
4. Hotelling, H.: Analysis of a Complex of Statistical Variables Into Principal Components. Warwick & York (1933). <https://books.google.cz/books?id=qJfXAAAAMAAJ>
5. Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T.: Harmonic coordinates for character articulation. *ACM Trans. Graph.* **26**(3) (2007). <https://doi.org/10.1145/1276377.1276466>
6. Little, J., Hill, D., Hawkes, D.: Deformations incorporating rigid structures. *Comput. Vis. Image Underst.* **66**(2), 223–232 (1997). <https://doi.org/10.1006/cviu.1997.0608>. <http://www.sciencedirect.com/science/article/pii/S1077314297906081>
7. Modersitzki, J.: Numerical Methods for Image Registration. Oxford University Press, Oxford (2004)
8. Pearson, K.: On lines and planes of closest fit to systems of points in space. *Philos. Mag.* **2**(11), 559–572 (1901). <https://doi.org/10.1080/14786440109462720>
9. Persing, J., James, H., Swanson, J., Kattwinkel, J.: Prevention and management of positional skull deformities in infants. *Pediatrics* **112**(1), 199–202 (2003). <http://pediatrics.aappublications.org/content/112/1/199>

10. Schreen, G., Matarazzo, C.G.: Plagiocephaly and brachycephaly treatment with cranial orthosis: a case report. *Einstein* **11**, 114–118 (2013). http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1679-45082013000100021&nrm=iso
11. Schroeder, W., Martin, K.M., Lorensen, W.E.: *The Visualization Toolkit: An Object-oriented Approach to 3D Graphics*, 2nd edn. Prentice-Hall Inc, Upper Saddle River (1998)
12. Sieger, D., Menzel, S., Botsch, M.: High quality mesh morphing using triharmonic radial basis functions. In: Jiao, X., Weill, J.C. (eds.) *Proceedings of the 21st International Meshing Roundtable*, pp. 1–15. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-33573-0_1



Implementation of BM3D Filter on Intel Xeon Phi for Rendering in Blender Cycles

Milan Jaros, Petr Strakos^(✉), and Tomas Karasek

IT4Innovations, VSB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic
{milan.jaros, petr.strakos, tomas.karasek}@vsb.cz

Abstract. In this paper parallel implementation of Sparse 3D Transform-Domain Collaborative filter (BM3D) on the Intel Xeon Phi architecture is presented. Efficiency of the implementation in terms of speedup compared to serial implementation of the filter is demonstrated on denoising of rendered images. We also provide comparison with another parallel CPU version and show that ours performs better.

Using the state-of-the-art image filters such as BM3D offers powerful denoising capability in the area of image filtering. To achieve the highest possible quality of the result, the filter has to perform multiple demanding tasks over a single image. Effective implementation of the filter is therefore very important. This is also the case, when filtering is used for image rendering. Rendering times can be significantly decreased by application of powerful time efficient denoising filters. Unfortunately the existing serial implementation of the BM3D filter is time consuming. In this paper we provide efficient parallel implementation of the BM3D filter, and we apply it as a noise reduction technique to the rendered images that reduces the rendering times. We also provide an optimized version of the filter for the Intel Xeon Phi and Intel Xeon architecture.

Keywords: Image denoising · Intel Xeon Phi · Blender cycles
Rendering · Collaborative filtering · High performance computing

1 Introduction

There are plenty of areas where advanced image filtering methods can be employed. They are extensively used in medical imaging, computer vision, or they can be effectively applied in the area of image rendering. Here, very often, path tracing algorithms utilizing the Monte Carlo (MC) method are used [9]. Based on the statistical approach, such rendering systems trace each ray of the light in the scene and calculate its effect on an individual object based on the environmental and material parameters. The results of such renderers bring high level of realism. This is unfortunately paid back by long rendering times. On the other hand, using only a small amount of rendered samples per pixel can reduce the time but also incorporates a high level of noise in the image. To reduce the

rendering times while keeping the photo realistic quality, utilization of the image denoising methods has been studied [1, 5].

The computationally extensive tasks are usually solved on powerful workstations, or they can use the computational power of supercomputers. Image rendering is one of the areas that generates high computational load. Utilization of supercomputers therefore becomes an interesting idea.

In our contribution, we concentrate on the state-of-the-art image denoising BM3D method and customize it for HPC or multi-core environment. The method has been published in [3] and is one of the best in the area of image denoising [7]. We use the color variant of the method, which has been presented in [2]. Since the method is computationally demanding, its practical potential has been lower. For this reason, we provide its effective parallel implementation. We present implementation that suits two different computer architectures, Intel Xeon and Intel Xeon Phi. Our main focus is put on Intel Xeon Phi with their many integrated core (MIC) architecture. It is nowadays extensively used in supercomputing centres worldwide, apart from the typical CPU architecture. We provide comparison of our parallel version of the filter running on the two architectures with the sequential version of the filter [2]. We also compare our implementation with different parallel implementation from Lebrun [8] that utilizes OpenMP and runs on CPU. We present results in terms of total rendering and filtering times, and we also provide results showing increasing improvement by different optimizations. We also show the positive effect of the filter on the reduction of the rendering time while conserving the final image quality. It is important to mention that we have elaborated the method within our own developed rendering concept called CyclesPhi that can utilize the power of supercomputers, and we make it available within the open-source 3D creation Blender suite.

2 Previous Work

Large amount of algorithms has been studied by researchers in terms of noise reduction in rendering methods based on Monte Carlo. It is possible to divide the algorithms into those trying to modify sampling of the renderer and those using filtering techniques to decrease the residual noise of the renderer.

The recent filtering techniques within the Monte Carlo based rendering systems propose, for example, the iterative approach as in [10]. Here, the authors use a two-step iterative process. First, for the initially rendered image with low samples they use a set of filters in every pixel and select the filter that minimizes the pixel error in terms of mean square error (MSE). Second, for the filter selection they additionally increase the pixel samples based on the filter selection and re-render the image and proceed again through first step. The authors are using discrete set of simple Gaussian filters, which need appropriate amount of samples/pixel (32 samples/pixel used) to work effectively and to obtain satisfactory results.

Another filtering approach presented in [6] considers utilization of the advanced filtering methods in the concept that is primarily intended as a post-processing filtering step. It utilizes the BM3D method as a filter. The authors provide a multilevel denoising algorithm, which first estimates the noise level locally from the close neighbourhood around each image pixel. For such a noise map they reconstruct a histogram of noise levels. Histogram is then divided to several levels defined by the user. To do this, cumulative distribution function (CDF) of the noise map is used. Each discrete level is characterized with the value of standard deviation. For every value of standard deviation filtering is provided. Resulting images are then combined to compute the final image. The provided concept does not depend on used filter, but it performs best with the BM3D method. The authors use the original Matlab C/C++ MEX implementation of BM3D method provided by Dabov et al. in [2,3]. This version does not leverage any parallel programming. Although in context of total rendering time of a single image its runtime is fast, in context of image filtering of larger image series it is slow. This version also can not utilize a specific hardware such as GPU or MIC. In our contribution we provide an optimized parallel version of the color BM3D filter for CPU and MIC architecture.

Concerning the BM3D filter and its available parallel implementations that can serve for general purpose, meaning also for rendering, we have found just two of them [8,11]. The one from Lebrun utilizes multi-core CPUs while the Sarjanoja et al. tackle GPUs. As for the implementation of Sarjanoja et al., they perform better than original Dabov's single threaded version only if they use the so called modified profile. This modified profile unfortunately brings lower filtering quality than the original parameters [4] of the method. Sometimes the difference is negligible sometimes it is not. Within the original profile, beside the lower speed they also run out the memory while filtering UHD (3840×2160) images. An odd thing, which we find in their contribution, is that their testing of Lebrun's parallel version performs worse than the original single threaded version of Dabov's. This is completely opposite to the findings one would expect and it is also opposite to our findings we provide in our contribution within Sect. 7.

3 BM3D Filtering Method

Block-matching and the 3D collaborative filtering method operate over the image trying to minimize the amount of noise based on sparsity of similar image blocks. The filtering method is general with respect to the type of attenuated noise [3], but for ease of explanation the noise is assumed as Additive White Gaussian Noise (AWGN). This can be formulated by the following equation

$$z(x) = y(x) + \eta(x), \quad (1)$$

where z stands for the evaluated image, y is the noiseless image and η is the additive zero-mean Gaussian noise. Variable x stands for the pixel coordinate within the image.

The detailed description of the BM3D method is covered in [3]. The color version of the method is presented in [2]. Here we summarize just the main aspects of the method and accentuate the most computationally extensive parts of the algorithm.

BM3D is a two-step method. In the first step, image is divided into several overlapping areas where similar smaller parts of the image called patches are searched for. Searching for the similar patches is done in a sparse domain provided by wavelet transform of each patch. The found similar patches are stacked in the 3D array and the whole stack is transformed from image to sparse representation by 3D transform (specific combination of transform matrices providing the 3D transform is stated in [3]). Here the filtering operation in the form of the hard thresholding is performed. After this the stack is transformed back to the image domain. This can be symbolically written as

$$\mathbf{Y} = T_{3D}^{-1}(\mathcal{Y}(T_{3D}\mathbf{Z})), \quad (2)$$

where \mathbf{Y} is the stack of filtered patches, T_{3D} is specific 3D transform, \mathcal{Y} represent the filtering operation and \mathbf{Z} is the stack of noisy image patches. This is the core of the method and due to the searching of the similar patches in sparse domain the filtering can be very effective. Each patch from the filtered stack is redistributed back to its position within the image and all overlapping pixels are aggregated and averaged out by weighted average. This can be symbolically written as

$$\hat{y}(x) = \frac{\sum \sum w \cdot Y(x)}{\sum \sum w \cdot \chi(x)}, \quad (3)$$

where \hat{y} is the filtered image, w is appropriate weight, Y is a patch from the considered stack of patches, and χ is the patch support. Summation goes through all the patches within one stack and through all the stacks within one image.

In the similar manner, the second step of the method is performed. It uses the results of the preceding step to filter out the noise even further from the noisy input image. Equations 2 and 3 are also used here, differences are only in the way how the searching of similar patches is provided, and how the filtering inside the sparse domain represented by \mathcal{Y} is provided. Both filtering steps of the method are graphically summarized in Fig. 1.

3.1 BM3D - Computationally Extensive Parts

The BM3D method operates in a sparse domain, where all the filtering is performed either by hard-thresholding in Step 1 or by Wiener filtering in Step 2. Conversion to sparse domain is done by matrix multiplication of each selected image patch with transformation matrices. The number of patches that are transformed and then further processed is high. Based on the recommended setting of the method, as elaborated in [4, 8], it is around 1300 patches for every single reference patch. The number of the reference patches depends on the image resolution, but generally it is about 1/16th of the number of image pixels. In the case of rendering tasks, image resolution is typically HD (1920 × 1080 pixels) or

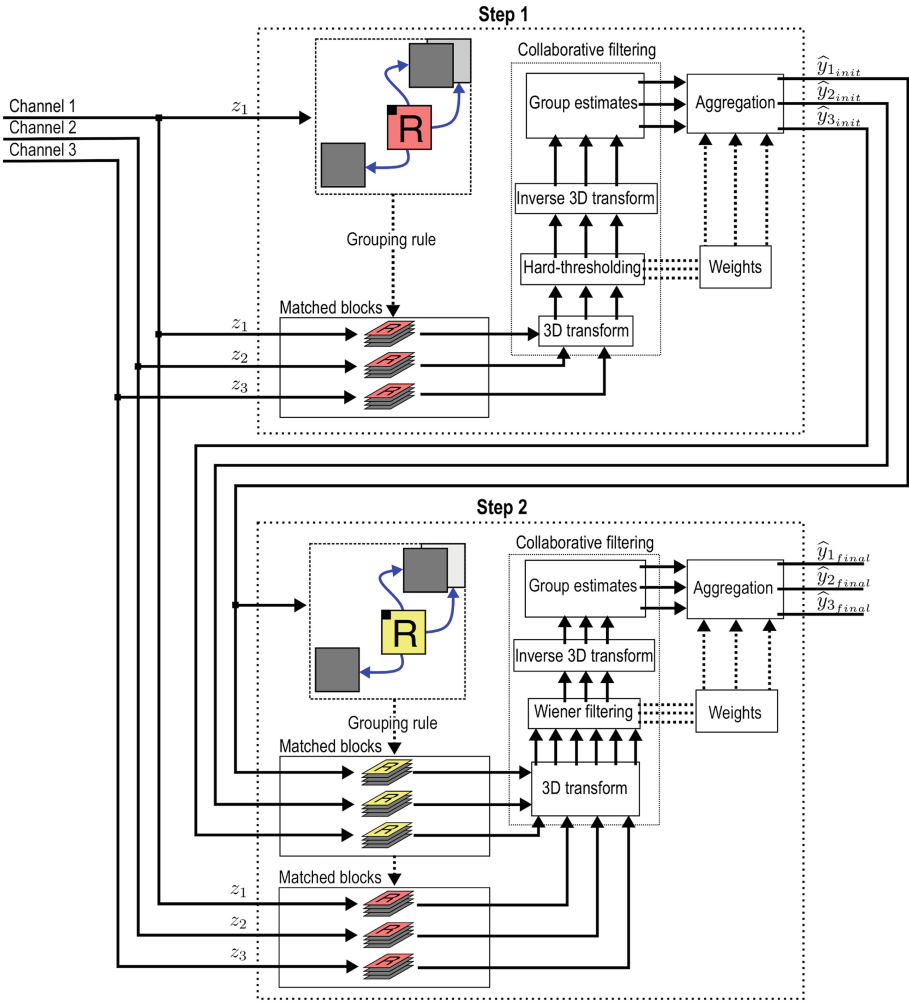


Fig. 1. Workflow of the collaborative filtering method

higher, which gives approximately 130 000 of reference patches at least. Summing it all up, it gives around 340 000 000 of image patches that are being processed just in Step 1. Similar counting holds also for the Step 2. Luckily computations can be parallelized around each reference patch. Limitations are set if one tries for concurrency between Step 1 and Step 2. It is not possible since the results of Step 1 are used right at the start of Step 2.

Another issue arises with pixel aggregation denoted by Eq. 3, where assignment to individual pixel positions is performed. This appears in both of the filtering steps. If one uses parallelization concept where area around each reference patch is solved by individual thread of multiple threads, then one needs

to assign to each pixel position sequentially, because the areas are overlapping and therefore multiple access to same memory location can easily occur if not carefully treated. This is an huge bottleneck and speed barrier if solved trivially by sequential code. If we want to parallelize the aggregation process we have to ensure that at one time instant memory area at specific address is accessed just by one thread and meanwhile non-blocking any other communication from the remaining threads.

Our solution to the mentioned issues using parallelization is described in Sect. 6.

4 Denoising Capabilities of the Collaborative Filtering Method on Rendered Images

As it is experimentally tested in [7], BM3D as a state-of-the-art filtering method outperforms many of the actual image denoising methods. We show its application on two distinct rendered scenes. They differ in the depth of field (DOF). First scene of Tatra car has large DOF, while scene with the worm has small DOF to highlight the worm body from the background. This is to accentuate the typical filtering problem if one wants to preserve the edges and also blur out the noise in the background. Each of the scenes has been rendered for an increasing set of samples per pixel (spp) as shown in Fig. 2, for Tatra and Fig. 3, for the worm respectively. The number of rendered samples/pixel is proportional to the level of noise that remains in the image after the rendering. The higher the number of samples the lower the amount of the residual noise. Filtered images up to 512 samples/pixel are shown in Figs. 4 and 5. For higher sampling filtering loses its effect because the level of noise after rendering is low. A complete list of the computed results is presented in Tables 1 and 2. In case of Tatra scene, it makes sense to use filtering up to 512 samples/pixel and up to 2048 samples/pixel in case of the worm. It is possible to use 2 or 4 times lower number of samples/pixel if we use filtering to obtain the same visual quality of the rendered scene.

5 Sequential Implementation of BM3D Filter

The sequential version of the algorithm is summarized in the following pseudo-code. As stated in Sect. 3.1, computationally extensive parts, printed in bold, are concentrated in 3(b)i-3(b)iii and in both aggregation parts 4 and 6.

1. Load image
2. Set parameters and create transformation matrices for Step 1, Step 2 (see [3, 8])
3. Step 1 - Calculate the group estimates
 - (a) Set positions of reference blocks
 - (b) **for** $i = 1 \div R_1$ number of reference blocks in the image
 - i. **Do grouping by matching on Channel 1**
 - ii. Use created matching rule on Channel 2, 3

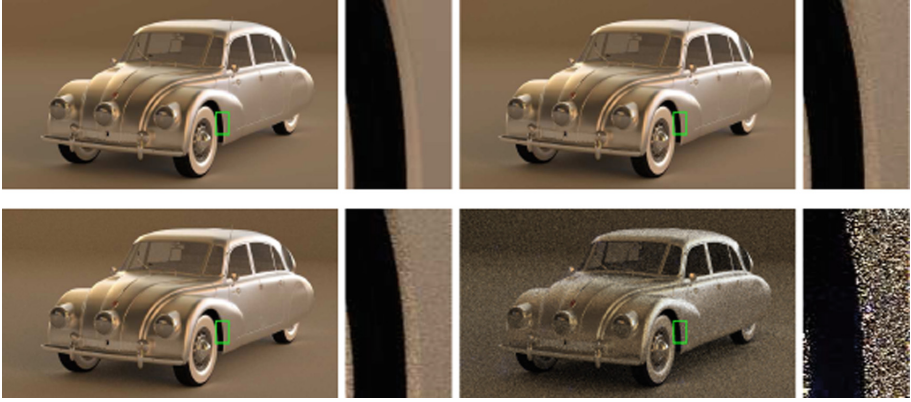


Fig. 2. Rendered scene of Tatra car **without** filtering; from left to right, up to bottom - 8196 samples/pixel, 256 samples/pixel, 64 samples/pixel, 1 sample/pixel



Fig. 3. Rendered scene of the worm **without** filtering; from left to right, up to bottom - 8196 samples/pixel, 256 samples/pixel, 64 samples/pixel, 1 sample/pixel

- iii. **Perform collaborative filtering**
- iv. Compute the weight value for the composed group of blocks
- (c) **end**
4. Step 1 - **Aggregate the group estimates, weights and compute the initial image estimate**
5. Step 2 - Calculate the group estimates
 - (a) Set positions of reference blocks
 - (b) **for** $i = 1 \div R_2$ number of reference blocks in the image
 - i. **Do grouping by matching on Channel 1 of the initial image estimate**
 - ii. Use created matching rule on Channel 2, 3 of the initial image estimate



Fig. 4. Rendered scene of Tatra car **with** filtering; from left to right, up to bottom - 512 samples/pixel, 256 samples/pixel, 64 samples/pixel, 1 sample/pixel



Fig. 5. Rendered scene of the worm **with** filtering; from left to right, up to bottom - 512 samples/pixel, 256 samples/pixel, 64 samples/pixel, 1 sample/pixel

- iii. Collect image blocks from noisy channels using the matching rule
 - iv. Compute the Wiener coefficients
 - v. Compute the weight value corresponding to the filtered group of blocks
 - vi. **Perform the collaborative Wiener filtering**
- (c) **end**
6. Step 2 - **Aggregate the group estimates, weights and compute the final image**

Table 1. Computed values for Tatra scene; 8-bit/channel range; FHD (1920×1080) resolution; MSE - Mean Square Error, SSIM - Structural Similarity Index, PSNR - Peak Signal-to-Noise Ratio

Tatra	Raw			Filtered			
Samples/pixel	MSE	SSIM	PSNR	MSE	SSIM	PSNR	σ
1	4581.158	0.065	11.521	1358.037	0.647	16.802	59.525
2	2524.610	0.095	14.109	509.139	0.754	21.062	46.469
4	1436.970	0.136	16.556	200.482	0.878	25.110	36.123
8	772.817	0.193	19.250	83.286	0.900	28.925	26.842
64	94.142	0.541	28.393	12.235	0.960	37.255	9.509
128	35.356	0.735	32.646	8.703	0.965	38.734	5.851
256	19.637	0.818	35.200	5.740	0.974	40.542	4.354
512	10.262	0.888	38.019	4.285	0.980	41.812	3.146
1024	3.389	0.959	42.831	3.514	0.983	42.673	1.819
8192	0.000	1.000	-	0.000	1.000	-	0.000

Table 2. Computed values for the worm scene; 8-bit/channel range; FHD (1920×1080) resolution

Worm	Raw			Filtered			
Samples/pixel	MSE	SSIM	PSNR	MSE	SSIM	PSNR	σ
1	6496.412	0.053	10.004	3410.300	0.496	12.803	65.007
2	4880.197	0.071	11.246	1938.514	0.588	15.256	60.648
4	3346.443	0.097	12.885	958.296	0.679	18.316	52.962
8	2015.827	0.131	15.086	408.117	0.763	22.023	42.771
64	352.891	0.350	22.654	47.618	0.836	31.353	18.600
128	173.270	0.477	25.744	22.810	0.894	34.550	13.088
256	89.716	0.599	28.602	13.377	0.928	36.867	9.438
512	40.439	0.743	32.063	8.617	0.951	38.777	6.346
1024	18.643	0.849	35.426	5.661	0.963	40.602	4.313
2048	8.049	0.923	39.073	5.965	0.969	40.374	2.834
4096	2.697	0.971	43.822	6.337	0.973	40.112	1.641
8192	0.000	1.000	-	0.000	1.000	-	0.000

6 Parallel Implementation of BM3D Filter

To achieve the best possible implementation in terms of the algorithm speed, we have implemented it in C++ (Intel Compiler 2017.1) and integrated it under our CyclesPhi rendering engine. We have used OpenMP standard with its `#pragma` directives for parallel programming and `SIMD` directives for vectorization.

We use our own vectorized code for operations with matrices (summation, subtraction, multiplication, L_2 norm) because it performs better than Eigen or MKL libraries on small matrices that are used within the filter implementation. We work mainly with matrices of size 8×8 . Due to such small size of matrices filter has to perform hundreds of millions of operations. We solve this issue by `#pragma omp simd` vectorization of the aforementioned operations. By using in-lined functions for the operations we further shorten the computation time. Advantage of using `#pragma omp simd` is that it accommodates to the architecture that is being used (AVX2 and KNC in our case).

Although this implementation is already quite efficient, large bottleneck in matrix multiplications still persisted. This was solved by direct assembler implementation, which helps especially on MIC architecture, as can be seen in results. The assembler code was generated by library for small matrix multiplication (LIBXSMM v1.8). We have generated the assembler code for the commonly used sizes of multiplied matrices.

Another issue to deal with was the memory allocation for the small matrices. This problem is more concerning MIC than CPU. While using Eigen library the time necessary for memory allocation and de-allocation was too high. This was another reason for moving to our own solution. Beside own code for operations with matrices we have defined own classes for the matrices and handle the memory allocation within them. At the beginning, we allocate all the memory necessary for the computations and during the computations we dynamically assign the memory. This way we significantly accelerate the computation and also the initialization of variables that are needed for the task.

For parallelization of serial code the obvious step is to parallelize the operations around each reference patch using `#pragma omp parallel`. It means the for loop 3(b)–3(c), (5(b)–5(c)) in pseudo-code of Sect. 5 was parallelized this way. Another computationally extensive operation in serial code is aggregation of the group estimates (4 and 6. in pseudo-code). Here the parallelization has to be done more carefully, because we are aggregating the results from different memory locations to shared memory area for all threads. Multiple writes to the same part of the memory can occur. To prevent this, while retaining parallelization with its speedup, we efficiently re-ordered the vector of indexes that localize reference patches within the image. Re-ordering prevents patch overlapping between concurrently solved tasks around reference patches. The situation is documented in Fig. 6.

We have implemented the code with focus on MIC architecture, but all the enhancements of the code are fully compatible with CPU architecture. Gradual improvements on both CPU and MIC architecture are documented in the results section.

We have also performed a comparison between our parallel implementation of the BM3D and the one provided by Lebrun [8]. Lebrun implements CPU version of the BM3D filter that can leverage parallelization using OpenMP. We have compared our MIC and CPU versions with Lebrun’s in terms of speed and memory demands. For the reference also Dabov’s single threaded version is stated [2]. More details can be found in Sect. 7.

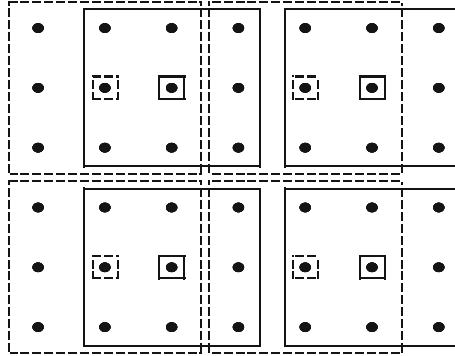


Fig. 6. Re-ordering of reference patches for preventing the area overlapping. Areas outlined by same line type (continuous, dashed) are solved concurrently in parallel. First all areas outlined by dashed line are solved by available thread, before areas outlined by continuous lines are being processed. In this way multiple writes are eliminated

7 Results

Tatra and worm scenes were used as examples for the tests of filter processing time. For a reference we also show the rendering times for different samples per pixel, see Table 3. We have tested the filter for the set of sampling used in Sect. 4. Specific sampling does not influence the filter runtime directly, but it affect the value of standard deviation of noise σ . There is a change in parameters of the filtering method (size of the patch, step between reference patches, etc.) based on the level of noise represented by the σ as recommended in [4, 8]. Parameters are different if $\sigma > 40$ or $\sigma \leq 40$. In case of Tatra, sampling from 1 to 2 samples/pixel has $\sigma > 40$, other sampling has lower σ , see Table 1. In case of the worm, up to 8 samples/pixel it is $\sigma > 40$, rest of the sampling has lower σ , see Table 2. This distinguishes the measured filtering runtimes. We have thus computed their mean values from all the measurements in each scene based on the value of σ . Results can be seen in Tables 4 and 5. There are also shown measurements on different architectures (CPU, MIC) with specific optimizations (AVX2, AVX2+SIMD, AVX2+xsmm+SIMD, KNC-offload, KNC-offload+SIMD, KNC-offload+xsmm+SIMD). In our tests KNC-offload stands for Xeon Phi (brand name Knights Corner) with offload programming model. All of these results are compared with the results obtained by running the filter implementation of Dabov et al. [2].

Our speed improvements by different optimization of the filtering algorithm are in graphical form represented in Fig. 7.

Our parallel implementation of the filter was further compared with parallel version by Lebrun. Comparison was made in terms of speed and memory utilization while filtering Tatra images of increasing resolution. For reference we have also compared it with single threaded Dabov's version. Results can be seen in Table 6. Filters are using the recommended parameter setting as stated in [4] under the normal profile. Our solution is much more efficient in terms of memory

Table 3. Rendering times for Tatra and the worm scene for specific samples per pixel; FHD resolution

Rendering time	Time [h:m:s.ms]					
	Samples/pixel	Tatra	Worm	Samples/pixel	Tatra	Worm
1		00:04.34	00:32.27	128	00:56.01	07:41.19
2		00:04.68	00:35.73	256	01:48.05	14:57.93
4		00:05.50	00:42.56	512	03:31.49	29:12.96
8		00:07.16	00:55.90	1024	06:58.27	58:13.70
16		00:10.31	01:22.07	2048	14:03.66	01:56:14.17
32		00:16.75	02:16.01	4096	28:02.70	03:52:25.31
64		00:29.68	04:05.13	8192	55:16.09	07:44:07.64

Table 4. Tatra - BM3D runtime for different optimizations and architectures compared with original version of the filter by Dabov et al.; FHD resolution

Filter runtime	Ours [s]		Dabov et al. [s]	
	$\sigma > 40$	$\sigma \leq 40$	$\sigma > 40$	$\sigma \leq 40$
Arch., Inst. set				
CPU, AVX2	76.9	139.5	107.4	66.4
CPU, AVX2+SIMD	23.0	24.7		
CPU, AVX2+xsmm+SIMD	12.7	15.0		
MIC, KNC-offload	369.6	598.8		
MIC, KNC-offload+SIMD	81.6	142.3		
MIC, KNC-offload+xsmm+SIMD	19.8	38.1		

Table 5. Worm - BM3D runtime for different optimizations and architectures compared with original version of the filter by Dabov et al.; FHD resolution

Filter runtime	Ours [s]		Dabov et al. [s]	
	$\sigma > 40$	$\sigma \leq 40$	$\sigma > 40$	$\sigma \leq 40$
Arch., Inst. set				
CPU, AVX2	77.9	141.4	109.0	62.9
CPU, AVX2+SIMD	23.3	25.2		
CPU, AVX2+xsmm+SIMD	12.8	15.1		
MIC, KNC-offload	370.2	597.9		
MIC, KNC-offload+SIMD	81.6	142.5		
MIC, KNC+xsmm-offload+SIMD	19.9	38.5		

utilization especially if compared with Lebrun, which utilizes incredible 100 GB of memory compared to ours 2 GB for image in FUHD resolution. In terms of speed our solution performs better or is the same up to FHD resolution of the image. On higher resolutions our implementation starts to lag behind Lebrun. Single threaded version performs always worse than others.

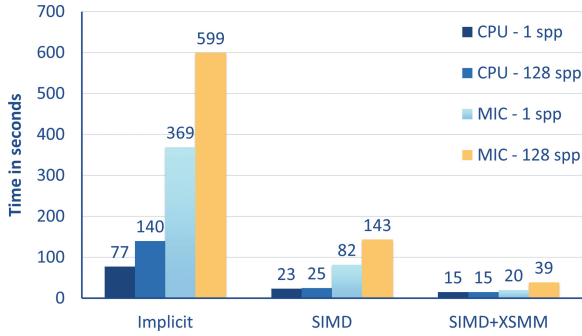


Fig. 7. BM3D runtime for different optimizations and architectures

Table 6. Tatra - our BM3D implementation compared to Lebrun’s and Dabov’s; images with 128 samples per pixel; $\sigma = 5$ as a filtering parameter; Normal profile from [4] used for BM3D setting

Resolution	CPU		MIC		Dabov		Lebrun	
	[s]	[GB]	[s]	[GB]	[s]	[GB]	[s]	[GB]
768×576	3.1	0.14	7.7	1.51	14.9	0.68	4.4	2.60
1280×720	6.7	0.17	16.2	1.55	31.9	0.76	7.7	4.50
1920×1080	15.0	0.24	38.1	1.65	73.4	0.92	14.9	8.60
2560×1440	30.9	0.33	67.7	1.76	131.3	1.11	24.9	13.50
3840×2160	62.2	0.60	151.6	2.15	278.9	1.63	58.6	28.00
7680×4320	295.9	2.10	623.8	3.84	955.7	4.20	204.5	101.00

All the tests were performed on one computing node of Salomon supercomputer. Specifically on 2x Intel Xeon E5-2680v3, 2.5 GHz for CPU tests and Dabov et al. tests and 1x Intel Xeon Phi 7120P, 61cores for MIC tests.

From the results it can be seen, how our parallelization concept can help in speeding up the algorithm runtime. Although the speedup is not ideally proportional in terms of utilized cores, we can bring up to 8x faster implementation on CPU (24 cores, 24 threads) and up to 5x faster implementation on MIC (61 cores, 244 threads) compared to one core solution of Dabov’s.

8 Conclusion

In our contribution we have presented optimized parallel version of the state-of-the-art filtering technique BM3D. Final version of the algorithm can run on two different architectures (CPU, MIC) and it can be efficiently used on supercomputers. Since compute nodes are often equipped with CPU+MIC our implementation could completely utilize those computing nodes and thus it can be extremely suitable in computationally extensive areas such as image rendering.

Provided implementation is faster than originally presented algorithm. The highest speed-up reaches up to $8\times$ in case of CPU and up to $5\times$ in case of MIC. We perform better also against different parallel implementation of the filter from Lebrun. Our solution is faster on smaller resolutions and utilizes memory much more efficiently. Although Lebrun's version is faster on high resolutions it becomes impractical due to extreme memory demands.

Acknowledgements. This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center - LM2015070".

References

1. Bauszat, P., Eisemann, M., Magnor, M.: Guided image filtering for interactive high-quality global illumination. *Comput. Graph. Forum* **30**(4), 1361–1368 (2011)
2. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space. In: *Proceedings - International Conference on Image Processing, ICIP*, vol. 1, pp. I313–I316 (2006)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising with block-matching and 3D filtering. In: *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 6064 (2006)
4. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process.* **16**(8), 2080–2095 (2007)
5. Dammertz, H., Sewtz, D., Hanika, J., Lensch, H.P.A.: Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In: Doggett, M., Laine, S., Hunt, W. (eds.) *High Performance Graphics. The Eurographics Association* (2010)
6. Kalantari, N.K., Sen, P.: Removing the noise in Monte Carlo rendering with general image denoising algorithms. *Comput. Graph. Forum* **32**(2 Part 1), 93–102 (2013)
7. Katkovnik, V., Foi, A., Egiazarian, K., Astola, J.: From local kernel to nonlocal multiple-model image denoising. *Int. J. Comput. Vis.* **86**(1), 1–32 (2010)
8. Lebrun, M.: An analysis and implementation of the BM3D image denoising method. *Image Process. On Line* **2**, 175–213 (2012)
9. Pharr, M., Jakob, W., Humphreys, G.: *Physically Based Rendering: From Theory to Implementation*, 3rd edn, pp. 1–1233 (2016)
10. Rousselle, F., Knaus, C., Zwicker, M.: Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.* **30**(6), 159:1–159:12 (2011)
11. Sarjanoja, S., Boutellier, J., Hannuksela, J.: BM3D image denoising using heterogeneous computing platforms. In: *Conference on Design and Architectures for Signal and Image Processing, DASIP*, vol. 2015, December 2015



Investigating Convergence of Linear SVM Implemented in PermonSVM Employing MPRGP Algorithm

Jakub Kružík^{1,2(✉)}, Marek Pecha^{1,2,3}, Václav Hapla^{3,4}, David Horák^{1,2,3},
and Martin Čermák^{1,2,3,5}

¹ IT4Innovations National Supercomputing Center,
VŠB - Technical University of Ostrava, Ostrava, Czech Republic
jakub.kruzik@vsb.cz

² Institute of Geonics CAS, Ostrava, Czech Republic

³ Department of Applied Mathematics, VŠB - Technical University of Ostrava,
Ostrava, Czech Republic

⁴ Department of Earth Sciences, ETH Zurich, Zürich, Switzerland

⁵ ENET Centre, VŠB - Technical University of Ostrava, Ostrava, Czech Republic

Abstract. This paper deals with the novel PermonSVM machine learning tool. PermonSVM is a part of our PERMON toolbox. It implements the linear two-class Support Vector Machines. PermonSVM is built on top of PermonQP (PERMON module for quadratic programming) which in turn uses PETSc. The main advantage of PermonSVM is that it is parallel. The parallelism comes from a distribution of matrices and vectors. The MPRGP algorithm, implemented in PermonQP, is used as a solver of the quadratic programming problem arising from the dual SVM formulation. The scalability of MPRGP was proven in problems of mechanics with more than billion of unknowns solved on tens of thousands of cores. Apart from the scalability of our approach, we also investigate the relations between training rate, hyperplane margin, the value of the dual functional, and the norm of the projected gradient.

Keywords: Support Vector Machines · SVM · PERMON
PermonSVM · PermonQP · MPRGP · Quadratic programming · QP

1 Introduction

In the last two decades, the Support Vector Machines (SVMs) [7], due to their accuracy and obliviousness to dimensionality [18], have become a popular machine learning technique with applications including genetics [5], image processing [9], and weather forecasting [16]. In this paper, we are only interested in SVMs for classification. SVMs belong to supervised learning algorithms, i.e. algorithms developing a decision model from labelled training samples (training dataset). The SVM decision model is represented by the maximal-margin hyperplane, i.e. the hyperplane that separates the training dataset into two classes

with the maximal possible gap between the hyperplane and both classes. Development of the SVM decision model leads to solving a quadratic programming (QP) problem. A brief description of SVMs is given in Sect. 2.

In Sect. 3, PermonSVM [10] is introduced. PermonSVM represents one of a few open-source SVM implementations for distributed environment (it is parallelized using MPI). It focuses on solving large SVM problems on supercomputers. PermonSVM is build on top of PETSc [4] and PermonQP [11]. PermonQP is a PETSc based package for the solution of large scale QP problems. It includes implementations of several QP solvers and it can also use any of KSP [17] and TAO [14] solvers.

Section 4 describes the MPRGP [8] algorithm used for the solution of the QP arising from the SVM formulation. MPRGP is implemented in PermonQP.

Finally, numerical results are presented in Sect. 5. We investigate the convergence of SVM by looking, in each MPRGP iteration, at the training rate (percentage of correctly classified samples in the training dataset), the hyperplane margin, the value of the QP cost function, and the norm of the projected gradient (used in the stopping criterion of MPRGP). The scalability of our approach is also demonstrated.

2 Support Vector Machines for Classifications

SVM is a supervised binary classifier, i.e. a classifier that decides whether a sample falls into either Class A (label 1) or Class B (label -1) by means of a model. The model is determined from the already categorised training samples in the training phase of the classifier. Unless otherwise stated, let us assume that the training samples are linearly separable, i.e. it is possible to separate the Class A samples and the Class B samples using a hyperplane. The essential idea of the SVM classifier training is to find the *maximal-margin hyperplane* that divides the Class A from the Class B samples by the widest possible empty strip, which is called the functional margin. The samples contributing to the definition of such hyperplane are called the *support vectors* – see the circled samples lying on the dashed hyperplanes depicted in Fig. 1.

Let us denote the training samples as a set of ordered pairs such that

$$T := \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\},$$

where m is the number of samples, $\mathbf{x}_i \in \mathbb{R}^n$ ($n \in \mathbb{N}$ represents a number of attributes) is the i -th sample and $y_i \in \{-1, 1\}$ denotes the label of the i -th sample, $i \in \{1, 2, \dots, m\}$. Let H be the *maximal-margin hyperplane* $\mathbf{w}^T \mathbf{x} - b = 0$, where \mathbf{w} is a normal vector; $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane H from the origin along its normal vector \mathbf{w} . The problem of finding the hyperplane H can be formulated as a constrained optimization problem in the following hard-margin primal SVM formulation:

$$\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w} \quad \text{s.t.} \quad y_i (\mathbf{w}^T \mathbf{x}_i - b) \geq 1. \quad (1)$$

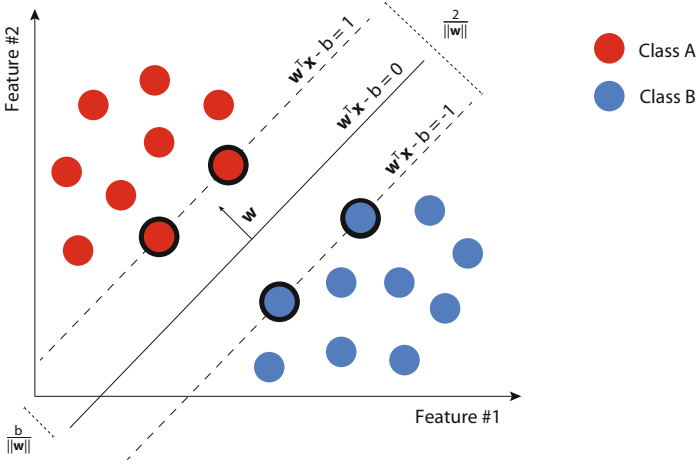


Fig. 1. An example of a two-class classification problem solved by the linear hard-margin SVM.

For the case of the non-perfectly linearly separable training samples, the soft-margin SVM was designed. To handle the sensitivity of the SVM classifier to possible outliers, we introduce slack variables $\xi_1, \xi_2, \dots, \xi_m$, and modify the hard-margin primal SVM formulation (1) into the soft-margin primal SVM formulation

$$\min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^m \xi_i \quad \text{s.t.} \quad \begin{cases} y_i (\mathbf{w}^T \mathbf{x}_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, \end{cases} \quad (2)$$

where C is a user-specified penalty¹. Higher value of C increases the importance of minimising $\|\mathbf{w}\|$ (equivalent to maximising the margin) at the expense of satisfying the margin constraint for fewer samples. Let us refer here to several monographs mentioning the significance of C :

- “In the support-vector networks algorithm one can control the trade-off between complexity of decision rule and frequency of error by changing the parameter C, \dots ” [7]
- “The parameter C controls the trade off between errors of the SVM on training data and margin maximization ($C = \infty$ leads to hard-margin SVM)” [15, p. 82].
- “... the coefficient C affects the trade-off between complexity and proportion of nonseparable samples and must be selected by the user” [6, p. 366].

We can observe that if $0 \leq \xi_i \leq 1$, then the i -th sample lies somewhere between the margin and their respective hyperplane (illustrated in Fig. 2); if $\xi_i > 1$, the i -th sample is misclassified (illustrated in Fig. 3).

¹ The penalty C is often called a regularization parameter in ML communities.

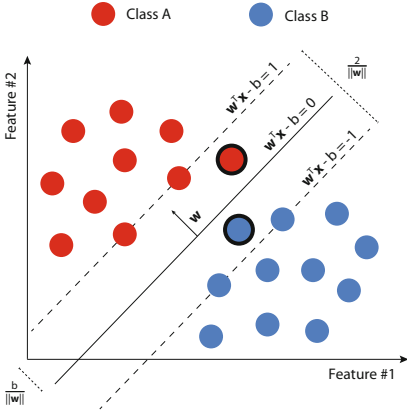


Fig. 2. Soft-margin SVM example: the encircled samples are correctly classified, but are on the wrong side of their respective hyperplane

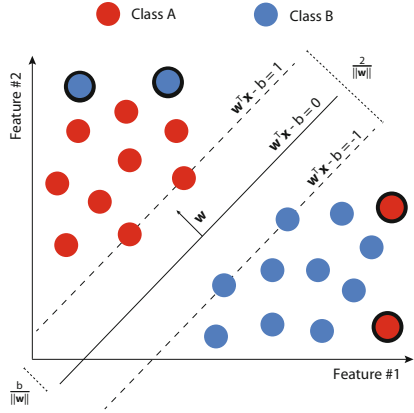


Fig. 3. Soft-margin SVM example: the encircled samples are misclassified.

The primal formulation of the soft-margin SVM (2) can be simplified by exploiting the Lagrange duality with the Lagrange multipliers $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_m]^T$, $\beta = [\beta_1, \beta_2, \dots, \beta_m]^T$. Evaluating the Karush-Kuhn-Tucker conditions, eliminating β and using other modifications, the problem results into the dual formulation with an inequality (box) constraint [7].

$$\min_{\alpha} \frac{1}{2} \alpha^T Y^T K Y \alpha - \alpha^T e \text{ s.t. } \mathbf{o} \leq \alpha \leq C e, \quad (3)$$

where $e = [1, 1, \dots, 1]^T$, $\mathbf{o} = [0, 0, \dots, 0]^T$, $X = [x_1, x_2, \dots, x_m]$, $y = [y_1, y_2, \dots, y_m]^T$, $Y = \text{diag}(y)$, and $K \in \mathbb{R}^{n \times n}$ is symmetric positive semi-definite (SPS) matrix such that $K := X^T X$. In the machine learning communities, K is called the Gram matrix, the kernel matrix, or in the QP terminology, the Hessian.

Further, we introduce dual to primal reconstruction formulas for the normal vector

$$w = X Y \alpha, \quad (4)$$

and the bias

$$b = \frac{1}{|I^{SV}|} \sum_{i \in I^{SV}} (x_i^T w - y_i), \quad (5)$$

where I^{SV} denotes the support vector index set, i.e. $I^{SV} := \{i \mid \alpha_i > 0, i = 1, 2, \dots, m\}$, and $|I^{SV}|$ is the cardinality of I^{SV} . From the normal vector w and bias b , we can easily set up the decision rule

$$\text{If } w^T x + b \geq 0, \text{ then } x \text{ belongs to Class A, else } x \text{ belongs to Class B.} \quad (6)$$

The decision rule (6) with concrete w and b is also called the SVM model for the linearly separable problems.

3 PermonSVM: SVM Implementation Based on PETSc

PermonSVM is a new SVM tool designed to run mainly in parallel even on large supercomputers. It is written on top of PETSc [4] and PermonQP [11]. Distribution of matrices using MPI through the PETSc framework provides the parallelism.

PermonSVM provides an implementation of the two-class classification via soft-margin SVM. It implements a scalable training procedure based on a linear kernel. In the training procedure, PermonSVM takes advantage of the scalable matrix-vector product of PETSc matrices and vectors and an implicit representation of the Gram matrix (i.e. the matrix product $\mathbf{X}^T \mathbf{X}$ is not formed), which saves memory and CPU time.

The resulting QP problem with an implicit Hessian matrix is solved by the scalable QP solvers implemented in the PermonQP package.

Additional features include fast, load-balanced cross-validation and grid search for parameter tuning, L1 and L2 loss-functions, and LIBSVM data parser. PermonSVM provides an executable for SVM classification as well as C API designed to be PETSc-like. Its typical usage is presented in Code 1.

```

MPI_comm comm = PETSC_COMM_WORLD;

PermonSVM svm;

Mat Xt, Xt_test; Vec y, y_test;

PetscInt n_examples = PETSC_DEFAULT, n_attributes = PETSC_DECIDE;
PetscInt n_test_examples = PETSC_DEFAULT, n_test_attributes = PETSC_DECIDE;
PetscInt numbering_base = 1;

/*PETSC_DEFAULT means find best C by means grid-search and cross-validation*/
PetscReal C = PETSC_DEFAULT;

char filename[PETSC_MAX_PATH_LEN] = "examples/heart_scale";
char filename_test[PETSC_MAX_PATH_LEN] = "examples/heart_scale.t";

PetscInt N_all, N_eq;

svm_file_load(filename, n_examples, n_attributes, numbering_base, Xt, y);
svm_file_load(filename_test, n_test_examples, n_test_attributes, numbering_base, Xt_test,
              y_test);

PermonSVMCreate(comm, &svm);

PermonSVMSetC(svm, C);
PermonSVMSetTrainingSamples(svm, Xt, y);

PermonSVMTrain(svm);
PermonSVMTest(svm, Xt_test, y_test, &N_all, &N_eq);

PermonSVMGetC(svm, &C);

```

Code 1: Calling PermonSVM API.

4 MPRGP Algorithm

MPRGP (Modified Proportioning and Reduced Gradient Projection) [8] represents an efficient algorithm for the solution of convex QP with box constraints, i.e. for

$$\min \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{x}^T \mathbf{b} \quad \text{s.t.} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \quad (7)$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is SPS, \mathbf{x} is the solution, \mathbf{b} is the right hand side, \mathbf{l} and \mathbf{u} is the lower respectively upper bound. The basic version can be considered as a modification of the Polyak algorithm. MPRGP combines the proportioning algorithm with the gradient projections.

Let $\mathbf{g} = \mathbf{Ax} - \mathbf{b}$ be the gradient. Than we can define component-wise (for $j \in \{1, 2, \dots, n\}$) gradient splitting which is computed after each gradient evaluation. The free gradient is defined as

$$g_j^f = \begin{cases} 0 & \text{if } x_j = l_j \quad \text{or} \quad x_j = u_j, \\ g_j & \text{otherwise.} \end{cases}$$

The reduced free gradient is

$$g_j^r = \begin{cases} 0 & \text{if } x_j = l_j \quad \text{or} \quad x_j = u_j, \\ \min\left(\frac{x_j - l_j}{\bar{\alpha}}, g_j\right) & \text{if } l_j < x_j < u_j \quad \text{and} \quad g_j > 0, \\ \max\left(\frac{x_j - u_j}{\bar{\alpha}}, g_j\right) & \text{if } l_j < x_j < u_j \quad \text{and} \quad g_j \leq 0, \end{cases}$$

where $\bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}]$ is used as a step length in the expansion step. The definition of the chopped gradient is

$$g_j^c = \begin{cases} 0 & \text{if } l_j < x_j < u_j, \\ \min(g_j, 0) & \text{if } x_j = l_j, \\ \max(g_j, 0) & \text{if } x_j = u_j. \end{cases}$$

Finally, the projected gradient is defined as $\mathbf{g}^P = \mathbf{g}^f + \mathbf{g}^c$. Its norm decrease is the natural stopping criterion of the algorithm.

Let the projection onto the feasible set $\Omega = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$ be defined as

$$P_\Omega(\mathbf{x})_j = \min(u_j, \max(l_j, x_j)).$$

Now we have all the necessary ingredients to summarise MPRGP in Algorithm 1.

5 Numerical Experiments

In this section, we show the scalability of our approach as well as what the relations between the training rate (percentage of correctly classified samples), the hyperplane margin, the value of the dual functional, and the norm of the projected gradient are. The hyperplane (given by \mathbf{w} and b) is computed in each iteration of MPRGP. Using the computed hyperplane, we can evaluate the training rate and the margin ($2/\|\mathbf{w}\|$). The value of the dual functional is trivially computed from the gradient which is available in every MPRGP iteration. The computation of these metrics is relatively expensive. Therefore, it is by default disabled, but it can be toggled by a command line switch. The decrease of the

Algorithm 1. MPRGP

Input: \mathbf{A} , $\mathbf{x}^0 \in \Omega$, \mathbf{b} , $\Gamma > 0$, $\bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}]$

- 1 $\mathbf{g} = \mathbf{A}\mathbf{x}_0 - \mathbf{b}$, $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^0)$, $k = 0$
- 2 **while** $\|\mathbf{g}^P(\mathbf{x}^k)\|$ is not small:
- 3 **if** $\|\mathbf{g}^c(\mathbf{x}^k)\|^2 \leq \Gamma^2 \mathbf{g}^r(\mathbf{x}^k)^T \mathbf{g}^f(\mathbf{x}^k)$:
- 4 $\alpha_f = \max\{\alpha_{cg} : \mathbf{x}^k - \alpha_{cg}\mathbf{p}\}$
- 5 $\alpha_{cg} = \mathbf{g}^T \mathbf{p} / \mathbf{p}^T \mathbf{A} \mathbf{p}$
- 6 **if** $\alpha_{cg} < \alpha_f$:
- 7 // CG step
- 8 $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_{cg}\mathbf{p}$
- 9 $\mathbf{g} = \mathbf{g} - \alpha_{cg}\mathbf{A}\mathbf{p}$
- 10 $\beta = \mathbf{g}^f(\mathbf{x}^{k+1})^T \mathbf{A}\mathbf{p} / \mathbf{p}^T \mathbf{A} \mathbf{p}$
- 11 $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^{k+1}) - \beta\mathbf{p}$
- 12 **else:**
- 13 // Expansion step
- 14 $\mathbf{x}^{k+\frac{1}{2}} = \mathbf{x}^{k+1} - \alpha_f\mathbf{p}$
- 15 $\mathbf{g} = \mathbf{g} - \alpha_f\mathbf{p}$
- 16 $\mathbf{x}^{k+1} = P_\Omega(\mathbf{x}^{k+\frac{1}{2}} - \bar{\alpha}\mathbf{g}^f(\mathbf{x}^{k+\frac{1}{2}}))$
- 17 $\mathbf{g} = \mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}$
- 18 $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^{k+1})$
- 19 **else:**
- 20 // Proportioning step
- 21 $\alpha_{cg} = \mathbf{g}^T \mathbf{g}^c(\mathbf{x}^k) / \mathbf{g}^c(\mathbf{x}^k)^T \mathbf{A} \mathbf{g}^c(\mathbf{x}^k)$
- 22 $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_{cg}\mathbf{g}^c(\mathbf{x}^k)$
- 23 $\mathbf{g} = \mathbf{g} - \alpha_{cg}\mathbf{A}\mathbf{g}^c(\mathbf{x}^k)$
- 24 $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^{k+1})$
- 25 $k = k + 1$

Output: \mathbf{x}^k

projected gradient norm is natural as well as the default stopping criterion of MPRGP.

Relations mentioned above are demonstrated on a dataset from the ExCAPE project [1] and also on the URL [13] dataset. The ExCAPE project aim is to predict compound bioactivity for the pharmaceutical industry. The tested dataset is related to Pfam protein database. It contains 226.1 thousand samples with 2048 attributes. The URL dataset relates to the detecting of malicious websites involved in criminal scams. It contains 2.4 million samples with 3.23 million attributes. The dataset is publicly available on LIBSVM datasets websites [2] in the LIBSVM format.

The experiments were run on the Salomon supercomputer [3] at IT4Innovations. Salomon consists of 1008 compute nodes. Each compute node contains two 2.5 GHz, 12-core Intel Xeon E5-2680v3 (Haswell) processors and 128 GB of memory. Compute nodes are interconnected by InfiniBand FDR56. Salomon has the peak performance of 2 petaFLOPS.

The initial guess was set to the zero vector. The relative norm of projected gradient (i.e. the ratio of the projected gradient norm and the right-hand side norm) being smaller than $1e-1$ was used as the stopping criterion in all numerical experiments. From our experience, while the tolerance is exceptionally high, it is more than adequate to find a good solution. This is illustrated by the following results.

In Tables 1 and 2 and in accompanying Figs. 4 and 5 the impact of the parameter C is shown. We report the maximal achieved training rate (Max rate) and the training rate upon the solver convergence (Converged rate) as well as the number of iterations needed to reach these rates.

Looking at the results of the ExCAPE dataset (Table 1 and Fig. 4), except for $C = 1e-5$, the difference between the maximal rate and converged rate ranges from 0.4 to 0.63%. Also note, that the best rate is achieved after relatively few iterations. To actually satisfy the convergence criterion it is necessary to do between 2.6 and 8 times as many iterations needed to get the maximum rate.

Table 1. ExCAPE dataset: comparison of the maximal achieved training rate (and iteration it occurred) with training rate obtained after solver converged (and again iteration this occurred).

C	1e-5	1e-4	1e-3	1e-2	1e-1
Max rate	76.8712	82.4114	84.3426	84.5243	84.6941
Converged rate	73.1472	81.8946	83.9301	84.1291	84.0641
Max rate iter	4	58	756	8415	9976
Converged rate iter	24	277	2868	22240	80306

The differences are much smaller for the URL dataset (Table 2 and Fig. 5). Again, we ignore in the following discussion the results for the smallest parameter C , because the solution is not good enough. The rate attained after the convergence is between 0.04 and 0.13% lower than the maximum rate. However, to reach the best rate it is necessary to do only from 50 to 70% of the number iterations needed to achieve convergence.

Further, we analyse the training rate, margin, value of dual functional, and norm of the projected gradient on the per iteration basis. The results are shown for the ExCAPE dataset in Figs. 6 and 7 for $C = 1e-3$, and for the URL dataset in Figs. 8 and 9 for $C = 1e-5$.

The MPRGP algorithm guarantees the decrease of the functional value in every iteration. In these examples, the norm of the projected gradient decreases monotonously as well. However, this is not guaranteed, and in fact, we observed high fluctuations for the ExCAPE dataset with larger values of the C parameter.

More interestingly, the training rate peaks after a relatively small number of iterations. The training rate also oscillates. It is barely noticeable in these examples. However, we observed very severe oscillation for the ExCAPE dataset

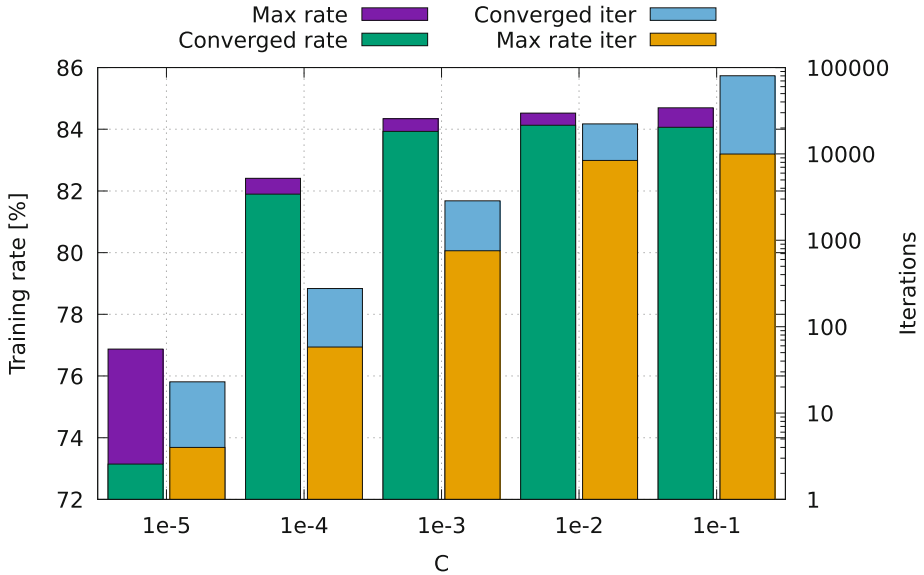


Fig. 4. ExCAPE dataset: comparison of the maximal achieved training rate (and iteration it occurred) with training rate obtained after solver converged (and again iteration this occurred).

Table 2. URL dataset: comparison of the maximal achieved training rate (and iteration it occurred) with training rate obtained after solver converged (and again iteration this occurred).

C	1e-7	1e-6	1e-5	1e-4	1e-3
Max rate	79.2815	96.0976	96.3368	97.2961	98.2226
Converged rate	78.3386	96.0374	96.2101	97.2186	98.1828
Max rate iter	5	60	571	5469	47329
Converged rate iter	10	120	1161	9525	68548

with larger values of the parameter C . The rate difference between consecutive iterations was sometimes over 17%. Also, notice that the hyperplane margin starts to decrease after relatively few iterations.

The decreasing value of the dual functional and that it is negative means, thanks to positive semi-definiteness of the Hessian and the positiveness of α , that the dual solution α , on the whole, increases. Meaning, that the satisfaction of the first constraint in (2) improves. The margin generally has a decreasing tendency, i.e. the norm of \mathbf{w} increases, suggesting (from (2)) that the sum of distances of samples from their respective hyperplanes decreases as well. Note, that this does not tell us anything about the training rate. In fact, we can see that improving this sum can lead to decrease in the training rate.

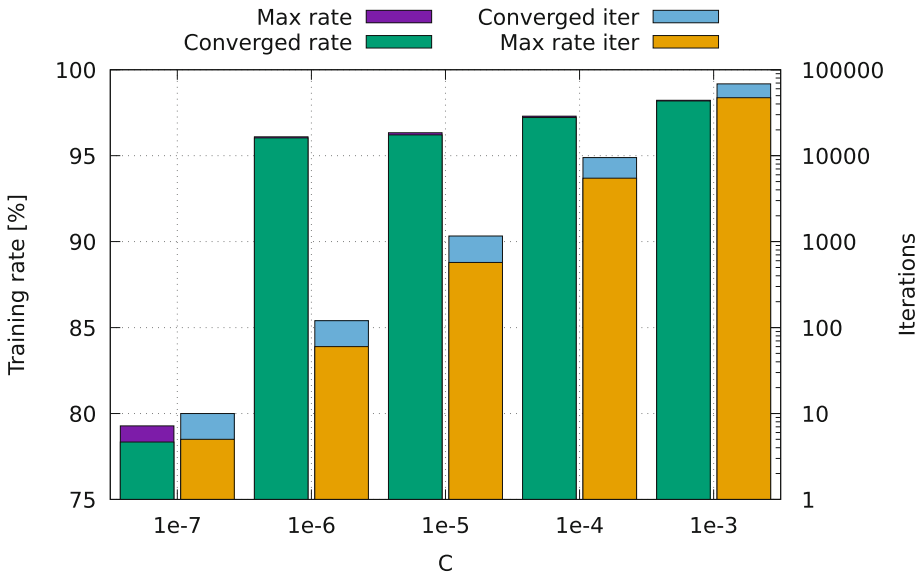


Fig. 5. URL dataset: comparison of the maximal achieved training rate (and iteration it occurred) with training rate obtained after solver converged (and again iteration this occurred).

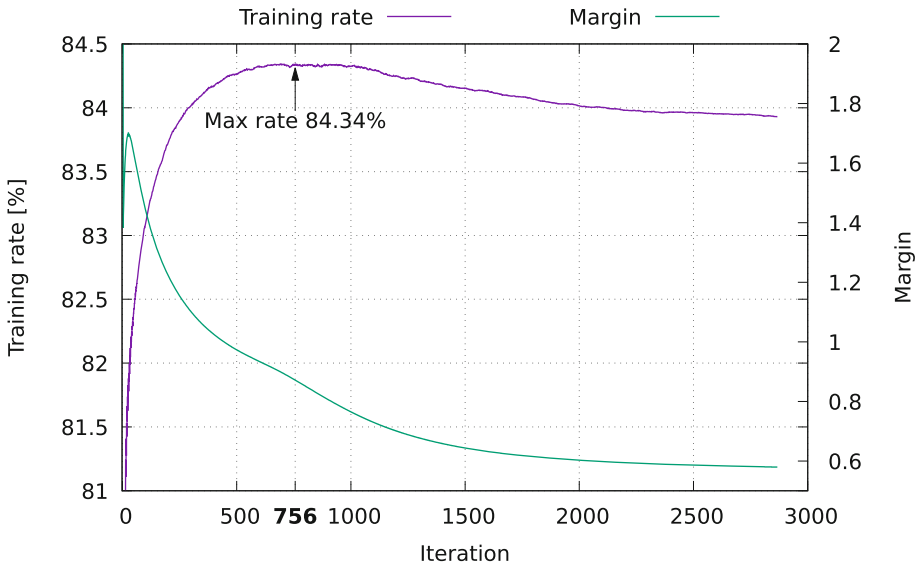


Fig. 6. ExCAPE dataset, $C = 1e-3$: the relation of the training rate and margin on the iteration number. The iteration number given in bold is where the maximum training rate was reached.

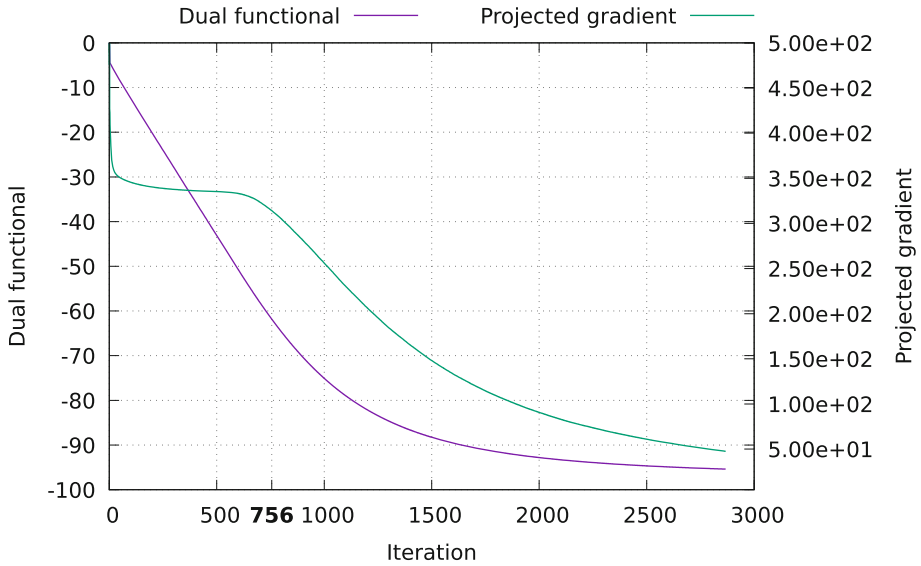


Fig. 7. ExCAPE dataset, $C = 1e-3$: the relation of the value of dual functional and the norm of the projected gradient on the iteration number. The iteration number given in bold is where the maximum training rate was reached.

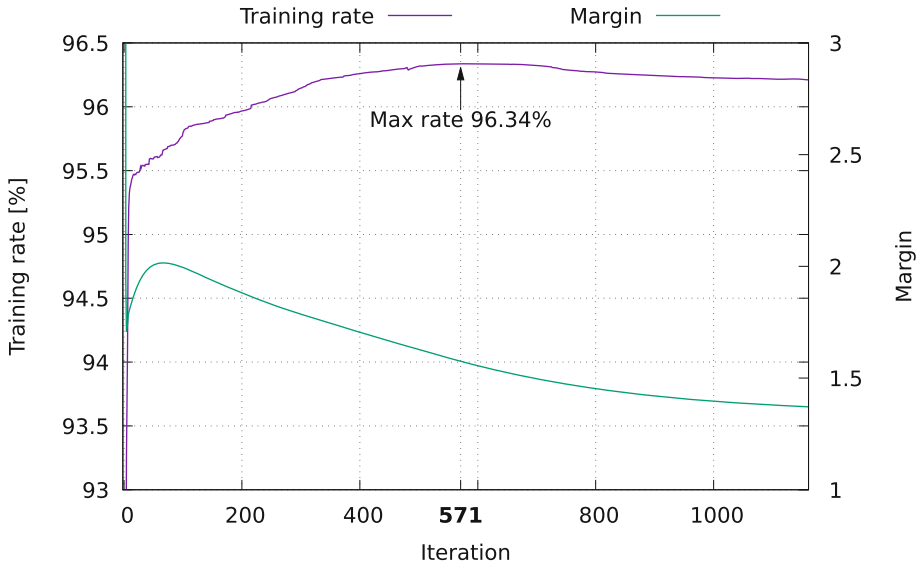


Fig. 8. URL dataset, $C = 1e-5$: the relation of the training rate and margin on the iteration number. The iteration number given in bold is where the maximum training rate was reached

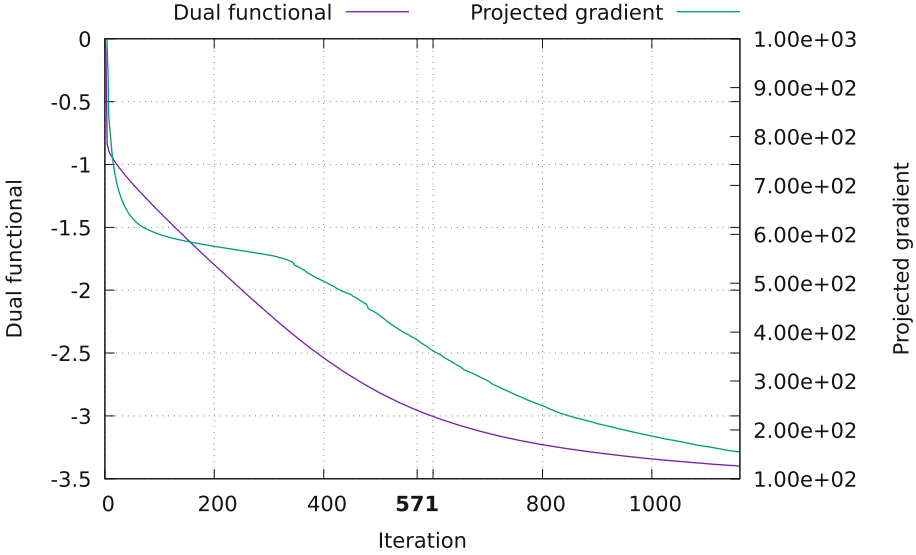


Fig. 9. URL dataset, $C = 1e-5$: the relation of the value of dual functional and the norm of the projected gradient on the iteration number. The iteration number given in bold is where the maximum training rate was reached.

Table 3. ExCAPE dataset, $C = 1e-3$: MPRGP strong parallel scalability

Number of cores	1	2	4	8	16	24	48
Time	289.36	158.87	81.75	43.86	30.29	28.95	19.63

The default stopping criterion of MPRGP based on the norm of the projected gradient seems ill-suited for SVM. It appears, despite the large tolerance, that the problems are solved unnecessary accurately. However, it is relatively easy to implement and use stopping criteria commonly used in SVM solvers. Looking only at the training rate, it seems that MPRGP can obtain a reasonable solution very quickly in few iterations.

Finally, we demonstrate the strong scalability of our solver. The big advantage of PermonSVM is that it can run in a distributed environment. Moreover, the MPRGP algorithm was proven to be scalable; for example, it can solve problems of mechanics with more than billion of unknowns on tens of thousands of cores [12]. The scalability results for the ExCAPE dataset are summarized in Table 3 and Fig. 10. The results for the URL dataset are presented in Table 4 and Fig. 11. The scalability is essentially the same as the scalability of the sparse matrix-vector product. This operation is memory bounded as illustrated by “Time on half nodes” results on the URL dataset. In this case, only half of the cores on each node are used (6 cores on each socket). This MPI rank placement significantly increases memory throughput. Thanks to this, the scaling is almost perfect up to 48 cores, after which the size of the distributed dataset starts to be too small to utilise the cores fully.

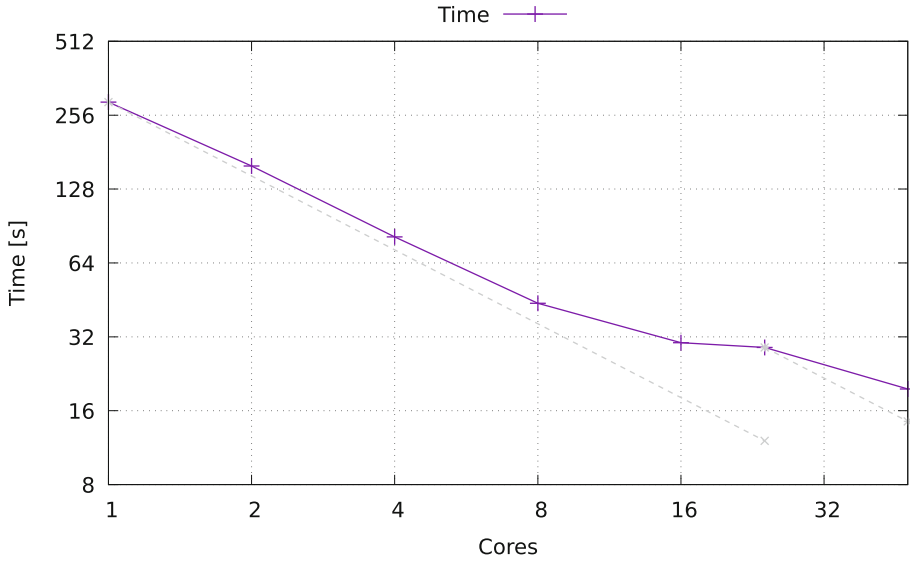


Fig. 10. ExCAPE dataset, $C = 1e-3$: MPRGP strong parallel scalability

Table 4. URL dataset, $C = 1e-5$: MPRGP strong parallel scalability

Number of cores	1	2	4	8	16	24	48	72	96	120	144	168
Time	4025.91	1996.99	1030.58	540.32	308.59	242.10	121.80	94.74	86.36	75.67	72.70	72.98
Time on half nodes	-	-	-	-	-	185.74	89.74	71.87	65.60	61.39	60.55	60.22

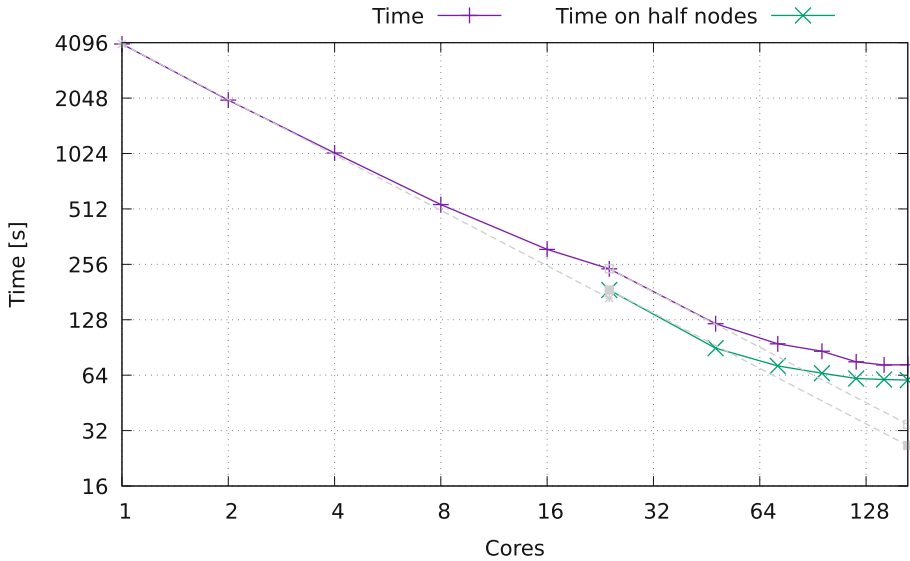


Fig. 11. URL dataset, $C = 1e-5$: MPRGP strong parallel scalability

6 Conclusion

We have introduced a novel, open-source PermonSVM machine learning tool employing scalable quadratic programming algorithms implemented in the PermonQP module. PermonSVM provides an implementation of the two-class classification via soft-margin SVM. Currently, it supports only a linear kernel. As a default, it uses the MPRGP algorithm for the solution of QP obtained from the dual SVM formulation.

We demonstrated the behaviour of the MPRGP algorithm on a dataset from the ExCAPE project as well as on the URL dataset. We analysed the relations between the training rate, the hyperplane margin, the value of the dual functional and the norm of the projected gradient on the per iteration basis. We note that the algorithm achieves a good training rate after relatively few iterations. The scalability of our approach was also demonstrated.

Further work will include implementation of a better stopping criterion and nonlinear kernels.

Acknowledgments. This work was supported by the Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science (LQ1602), and from the Large Infrastructures for Research, Experimental Development and Innovations project IT4Innovations National Supercomputing Center (LM2015070); by the internal student grant competition project SGS No. SP2018/165; by projects LO1404: Sustainable development of CENET, and CZ.1.05/2.1.00/19.0389: Research Infrastructure Development of the CENET; and by the Czech Science Foundation (GACR) projects no. 15-18274S and 17-22615S. We would also like to acknowledge partners in the ExCAPE project for providing us with training datasets related to the Pfam protein database.

References

1. ExCAPE: exascale compound activity prediction. <http://www.excape-h2020.eu>
2. LIBSVM data: classification, regression, and multi-label. <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>
3. IT4Innovations: Salomon cluster documentation - hardware overview. National Supercomputing Center, VSB-Technical University of Ostrava (2017). <https://docs.it4i.cz/salomon-cluster-documentation/hardware-overview>
4. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zhang, H.: PETSc - Portable, Extensible Toolkit for Scientific Computation. <http://www.mcs.anl.gov/petsc>
5. Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares Jr., M., Haussler, D.: Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Nat. Acad. Sci. U.S.A.* **97**(1), 262–267 (2000)
6. Cherkassky, V., Mulier, F.M.: *Learning from Data: Concepts, Theory, and Methods*. Wiley-IEEE Press, Hoboken (2007)
7. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)

8. Dostál, Z.: Optimal Quadratic Programming Algorithms, with Applications to Variational Inequalities. SOIA, vol. 23. Springer, New York (2009). <https://doi.org/10.1007/b138610>
9. Foody, G.M., Mathur, A.: The use of small training sets containing mixed pixels for accurate hard image classification: training on mixed spectral responses for classification by a SVM. *Remote Sens. Environ.* **103**(2), 179–189 (2006)
10. Hapla, V., Horák, D., Pecha, M.: PermonSVM (2017). <http://permon.it4i.cz/permonsvm.htm>
11. Hapla, V., Horák, D., Čermák, M., Kružík, J., Pospíšil, L., Sojka, R.: PermonQP (2015). <http://permon.it4i.cz/qp/>
12. Horak, D., Dostal, Z., Hapla, V., Kruzik, J., Sojka, R., Cermak, M.: Projector-less TFETI for contact problems: preliminary results. In: *Civil-Comp Proceedings*, vol. 111 (2017)
13. Ma, J., Saul, L., Savage, S., Voelker, G.: Identifying suspicious URLs: an application of large-scale online learning, pp. 681–688 (2009). Cited By 173
14. Munson, T., Sarich, J., Wild, S., Benson, S., McInnes, L.C.: TAO users manual. Technical report ANL/MCS-TM-322. Argonne National Laboratory (2015). <http://tinyurl.com/tao-man>
15. Rychetsky, M.: Algorithms and Architectures for Machine Learning Based on Regularized Neural Networks and Support Vector Approaches (*Berichte Aus Der Informatik*). Shaker Verlag GmbH, Herzogenrath (2001)
16. Shi, J., Lee, W.J., Liu, Y., Yang, Y., Wang, P.: Forecasting power output of photovoltaic systems based on weather classification and support vector machines. *IEEE Trans. Ind. Appl.* **48**(3), 1064–1069 (2012)
17. Smith, B.F., et al.: PETSc users manual. Technical report ANL-95/11 - Revision 3.5. Argonne National Laboratory (2016). <http://tinyurl.com/petsec-man>
18. Vishnu, A., Narasimhan, J., Holder, L., Kerbyson, D., Hoisie, A.: Fast and accurate support vector machines on large scale systems. In: *2015 IEEE International Conference on Cluster Computing*, pp. 110–119, September 2015



Using ESPRESO as Linear Solver Library for Third Party FEM Tools for Solving Large Scale Problems

Ondřej Meca^(✉), Lubomír Říha, Alexandros Markopoulos,
Tomáš Brzobohatý, and Tomáš Kozubek

IT4Innovations, VŠB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic
{ondrej.meca,lubomir.riha,alexandros.markopoulos,tomas.brzobohaty,
tomas.kozubek}@vsb.cz

Abstract. ESPRESO is a FEM package that includes a Hybrid Total FETI (HTFETI) linear solver targeted at solving large scale engineering problems. The scalability of the solver was tested on several of the world's largest supercomputers. To provide our scalable implementation of HTFETI algorithms to all potential users, a simple C API was developed and is presented. The paper describes API methods, compilation and linking process.

As a proof of concept we interfaced ESPRESO with the CSC ELMER solver and compared its performance with the ELMER FETI solver. HTFETI performs two level decomposition, which significantly improves both memory utilization and solver performance. To select optimal second level decomposition we have developed a performance model that controls decomposition automatically. This is a major simplification for all users that ensures optimal solver settings.

We show that the ESPRESO HTFETI solver is up to 3.7 times faster than the ELMER FETI solver when running on 13 500 MPI processes (the 614 compute nodes of the Salomon supercomputer) and solving 1.5 billion unknown problems of 3D linear elasticity.

Keywords: Total FETI · Hybrid Total FETI · ESPRESO · ELMER Automatic tuning model · Multi-level decomposition

1 Introduction

The ESPRESO library is an open-source highly parallel library for solving large scale engineering problems (e.g. heat transfer, structural analysis). Our main focus is to create a highly efficient parallel solver based on several FETI (Finite Element Tearing and Interconnecting) algorithms which are suitable for parallel machines with tens or hundreds of thousands of cores. ESPRESO itself is able to load problems described in several formats and produce output in the VTK format.

To make the library available to other researchers, a simple plain C array based API has been developed. Through the API, potential users have all the benefits of the fast and scalable FETI solver. This paper contains (i) a theoretical description of FETI methods used, (ii) a practical example of connecting the library with the widely used open-source tool CSC Elmer, (iii) a performance model to setup optimal decomposition and (iv) a performance evaluation on a 3D linear elasticity benchmark.

2 Finite Element Tearing and Interconnecting Methods

The history of the FETI (Finite Element Tearing and Interconnecting) method is longer than twenty years [1]. It is one of the most efficient domain decomposition techniques for parallel solving of boundary value problems described by partial differential equations (PDEs). The main idea consists of the PDE domain splitting into several non-overlapping subdomains, in which the PDEs are discretized separately and the interconnectivity of the primal PDE solution is enforced by the dual unknowns that are the Lagrange multipliers (LM). The classical FETI algorithm is based on eliminating the primal unknowns so that the resulting linear system, in terms of the LM, can be solved iteratively with the projected conjugate gradient method [2]. Over the years, numerous variants of the basic FETI concept have been developed. One reason for this was the effort to overcome difficulties with computing the action of generalized inverses of the stiffness matrices and identifying its kernel spaces. The FETI-DP variant [3, 4] modifies the original FETI method so that the stiffness matrix is non-singular, the kernel space is trivial, and the inverse to the stiffness matrix exists. It is due to the connectivity of the subdomains in the so-called *corner nodes*. The drawback of FETI-DP appears in real world problems with a complicated geometry, where it is not immediately clear what the corner nodes should be. The opposite strategy gave rise to the T(otal)FETI method [5], in which the kernel space is as large as possible. In this variant, the Dirichlet boundary conditions of the PDE problem are enforced by the LM so that the stiffness matrix is block diagonal and all diagonal blocks are subdomain stiffness matrices with the same kernel spaces.

Another important motivation for development of new FETI variants came with implementation on more sophisticated computer architectures, where multiple CPU cores are grouped into CPUs/sockets and multiple sockets are present in a single compute node. From the minimal communications point of view, it is reasonable to translate the computer architecture hierarchy into the FETI method. This leads to a new group of hybrid (two-level) FETI methods. The FETI-FETI-DP method proposed in [6, 7] combines the classical FETI method used on the global level with the FETI-DP method used on clusters. In this paper we deal with the TFETI-TFETI method that uses the TFETI method on both levels [8]. It will be denoted by the H(ybrid)TFETI method. The basic idea of the two-level FETI method is graphically explained in the following benchmark, in which we also introduce respective notation.

The Cube Benchmark Problem

For explanation and better demonstration of some aspects, techniques, manipulation with the algorithms, and methods in this paper, we use the cube benchmark with hierarchical decomposition and discretization depicted in Fig. 1.

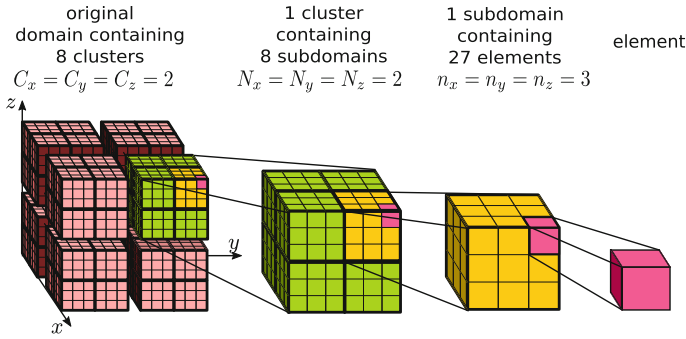


Fig. 1. Two levels of decomposition: 2 clusters ($C = 2$), 2 subdomains ($N = 2$), 3 elements ($n = 3$) in each space dimension

This hierarchical decomposition and discretization consists of three levels:

- Level 1 - decomposition into clusters is controlled by parameters C_x , C_y , and C_z (numbers of clusters in x , y , and z direction). Each cluster occupies one computational node.
- Level 2 - each cluster is decomposed into the subdomains controlled by parameters N_x , N_y , and N_z (numbers of subdomains in x , y , and z direction).
- Level 3 - each subdomain is discretized uniformly by hexahedral finite elements handled by parameters n_x , n_y , n_z (numbers of elements in x , y , and z direction).

If, for example, the number of clusters in all directions is the same $C_x = C_y = C_z = 2$, the description in the text is simplified to $C = 2$. The simplification is applied also to subdomains N and elements n .

The implementation of the Hybrid Total FETI method (HTFETI) does not differ significantly from the original approach (TFETI), and having both algorithms in one library requires few additions.

We will briefly introduce our HTFETI method for the 3-dimensional problem given by a metallic cube decomposed into two clusters and each cluster into two subdomains (see Fig. 2). After the FEM discretization, domain decomposition, and linear algebra object assembly, the linear system reads as follows:

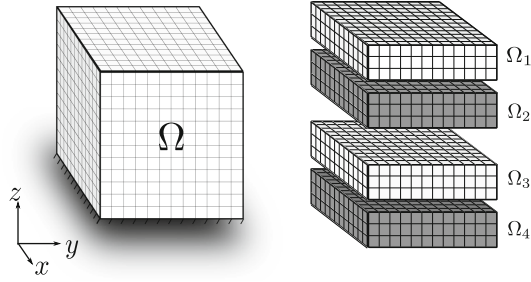


Fig. 2. Cube: $C_x = C_y = 1$, $C_z = 2$, $N_x = N_y = 1$, $N_z = 2$, $n_x = n_y = 12$, $n_z = 3$

$$\begin{pmatrix} \mathbf{K}_1 & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{B}_{c,1}^\top & \mathbf{O} & \mathbf{B}_1^\top \\ \mathbf{O} & \mathbf{K}_2 & \mathbf{O} & \mathbf{O} & \mathbf{B}_{c,2}^\top & \mathbf{O} & \mathbf{B}_2^\top \\ \mathbf{O} & \mathbf{O} & \mathbf{K}_3 & \mathbf{O} & \mathbf{O} & \mathbf{B}_{c,3}^\top & \mathbf{B}_3^\top \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{K}_4 & \mathbf{O} & \mathbf{B}_{c,4}^\top & \mathbf{B}_4^\top \\ \hline \mathbf{B}_{c,1} & \mathbf{B}_{c,2} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{B}_{c,3} & \mathbf{B}_{c,4} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \hline \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{B}_3 & \mathbf{B}_4 & \mathbf{O} & \mathbf{O} & \mathbf{O} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \\ \lambda_{c,1} \\ \lambda_{c,2} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \\ \mathbf{f}_4 \\ \mathbf{o} \\ \mathbf{o} \\ \mathbf{c} \end{pmatrix}, \quad (1)$$

where \mathbf{K}_i is the local stiffness matrix of the i -th subdomain, \mathbf{f}_i is the corresponding load vector, \mathbf{u}_i is the corresponding vector of displacements, $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4)$ are the standard FETI constraints matrices, \mathbf{c} is its corresponding right-hand side vector, λ is the vector of Lagrange multipliers enforcing the above constraints,

$$\mathbf{B}_c = \begin{pmatrix} \mathbf{B}_{c,1} & \mathbf{B}_{c,2} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{B}_{c,3} & \mathbf{B}_{c,4} \end{pmatrix}$$

is the constraints matrix with redundant constraints with respect to \mathbf{B} gluing subdomains inside clusters, $\lambda_c = (\lambda_{c,1}^\top, \lambda_{c,2}^\top)^\top$ is the corresponding vector of Lagrange multipliers.

To simplify the description of the algorithm, system (1) is permuted in the following way:

$$\begin{pmatrix} \mathbf{K}_1 & \mathbf{O} & \mathbf{B}_{c,1}^\top & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{B}_1^\top \\ \mathbf{O} & \mathbf{K}_2 & \mathbf{B}_{c,2}^\top & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{B}_2^\top \\ \mathbf{B}_{c,1} & \mathbf{B}_{c,2} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \hline \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{K}_3 & \mathbf{O} & \mathbf{B}_{c,3}^\top & \mathbf{B}_3^\top \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{K}_4 & \mathbf{B}_{c,4}^\top & \mathbf{B}_4^\top \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{B}_{c,3} & \mathbf{B}_{c,4} & \mathbf{O} & \mathbf{O} \\ \hline \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{O} & \mathbf{B}_3 & \mathbf{B}_4 & \mathbf{O} & \mathbf{O} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \lambda_{c,1} \\ \mathbf{u}_3 \\ \mathbf{u}_4 \\ \lambda_{c,2} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{o} \\ \mathbf{f}_3 \\ \mathbf{f}_4 \\ \mathbf{o} \\ \mathbf{c} \end{pmatrix}. \quad (2)$$

Let us introduce a new notation consistent with the line partitioning in (2)

$$\left(\begin{array}{c|c|c} \tilde{\mathbf{K}}_1 & \mathbf{O} & \tilde{\mathbf{B}}_1^\top \\ \hline \mathbf{O} & \tilde{\mathbf{K}}_2 & \tilde{\mathbf{B}}_2^\top \\ \hline \tilde{\mathbf{B}}_1 & \tilde{\mathbf{B}}_2 & \mathbf{O} \end{array} \right) \begin{pmatrix} \tilde{\mathbf{u}}_1 \\ \tilde{\mathbf{u}}_2 \\ \tilde{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{f}}_1 \\ \tilde{\mathbf{f}}_2 \\ \tilde{\mathbf{c}} \end{pmatrix}, \quad (3)$$

where $\boldsymbol{\lambda} = \tilde{\boldsymbol{\lambda}}$ and $\mathbf{c} = \tilde{\mathbf{c}}$. Eliminating $\tilde{\mathbf{u}}_I$, $I = 1, 2$, we also eliminate the subset of dual variables $\boldsymbol{\lambda}_{c,I}$, $I = 1, 2$ related to matrix $\tilde{\mathbf{B}}_c$. These LM are expressed exactly, and therefore the structure behaves like a problem decomposed into two subdomains instead of four. The group of subdomains glued together is denoted as a cluster. Here, the first cluster consists of subdomains 1 and 2, and the second cluster of subdomains 3 and 4.

$$\tilde{\mathbf{K}} = \text{diag}(\tilde{\mathbf{K}}_1, \tilde{\mathbf{K}}_2), \quad \tilde{\mathbf{B}} = (\tilde{\mathbf{B}}_1, \tilde{\mathbf{B}}_2), \quad \tilde{\mathbf{f}} = (\tilde{\mathbf{f}}_1^\top, \tilde{\mathbf{f}}_2^\top)^\top, \quad \tilde{\mathbf{R}} = \text{diag}(\tilde{\mathbf{R}}_1, \tilde{\mathbf{R}}_2), \quad (4)$$

where $\tilde{\mathbf{R}}_I$ is a kernel of the stiffness matrix $\tilde{\mathbf{K}}_I$. Similar to standard FETI notation we obtain

$$\begin{aligned} \tilde{\mathbf{F}} &= \tilde{\mathbf{B}}\tilde{\mathbf{K}} + \tilde{\mathbf{B}}^\top, & \tilde{\mathbf{G}} &= -\tilde{\mathbf{B}}\tilde{\mathbf{R}}, \\ \tilde{\mathbf{d}} &= \tilde{\mathbf{B}}\tilde{\mathbf{K}}\tilde{\mathbf{f}} - \tilde{\mathbf{c}}, & \tilde{\mathbf{e}} &= -\tilde{\mathbf{R}}^\top\tilde{\mathbf{f}}. \end{aligned} \quad (5)$$

By elimination of $\tilde{\mathbf{u}}_I$, $I = 1, 2$ in (3) we get the Schur complement system

$$\begin{pmatrix} \tilde{\mathbf{F}} & \tilde{\mathbf{G}} \\ \tilde{\mathbf{G}}^\top & \mathbf{O} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{\lambda}} \\ \tilde{\boldsymbol{\alpha}} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{d}} \\ \tilde{\mathbf{e}} \end{pmatrix} \quad (6)$$

that can be solved by the homogenization of the second equation $\tilde{\mathbf{G}}^\top\tilde{\boldsymbol{\lambda}} = \tilde{\mathbf{e}}$ and application of the same projected iterative methods as in the classical FETI approach to the homogenized system. The projections onto the kernel of $\tilde{\mathbf{G}}^\top$ are computed by the orthogonal projector

$$\tilde{\mathbf{P}} = \mathbf{I} - \tilde{\mathbf{G}} \left(\tilde{\mathbf{G}}^\top \tilde{\mathbf{G}} \right)^{-1} \tilde{\mathbf{G}}^\top. \quad (7)$$

For 3D linear elasticity problems, the dimension of the Coarse Problem (CP) matrix $\tilde{\mathbf{G}}^\top\tilde{\mathbf{G}}$ for 4 subdomains in 2 clusters is adequately smaller (the dimension of the CP is $2 \times 6 = 12$, compared to the TFETI case where the dimension is $4 \times 6 = 24$ for the same subdomain decomposition).

3 Implementation

The library is based on C++ and uses a hybrid parallelization in a form of MPI + OpenMP and vectorization. The communication between clusters (first level decomposition) is done using message passing (MPI). Subdomains inside a cluster (second level of decomposition) reside in a shared memory, therefore their processing is parallelized using OpenMP. The processing of a single subdomain is further accelerated by the vectorization.

The library is able to utilize a few external tools for linear algebra operations. The most time consuming part is using a sparse linear solver for pseudo inverse of local stiffness matrices \mathbf{K}_i . In the default settings ESPRESO uses sparse and dense BLAS operations and the PARDISO solver from the Intel MKL library [9]. Optionally, PARDISO from the Intel MKL library can be replaced by the original version of PARDISO [10,11], or the Dissection solver [12]. For the subdomain decomposition of a problem, ESPRESO uses Metis [13].

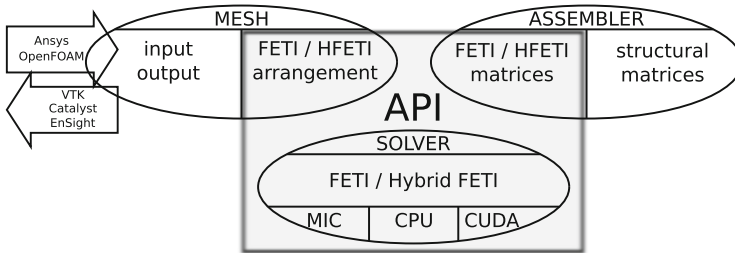


Fig. 3. A block diagram of the ESPRESO library.

The internal structure of ESPRESO is depicted in Fig. 3. The library can be divided into three main parts: *mesh*, *assembler*, and *solver*.

Mesh serves as an interface for unification of various input and output formats. It contains a description of the whole geometry decomposed into clusters. Based on the assembler settings, mesh decomposes clusters into subdomains using Metis and prepares structures for assembling TFETI/HTFETI matrices. Then, the assembler builds the structural matrices (K, f) and TFETI/HTFETI specific matrices (B_1, B_0) that are passed to the (H)TFETI solver. In the current version the assembler fully supports heat transfer (including transient or non-linear problems) and has limited support for linear elasticity. The solver has a general matrix based interface allowing it to solve an arbitrary type of a physical problem.

To make the ESPRESO library available to other researchers, a simple plain C array based API has been developed. This approach was chosen to provide a simple interface usable with programming languages other than C and C++ (e.g. Fortran). Through the API, potential users obtain all benefits of the ESPRESO library (a fast FETI solver scalable on several of the world's largest supercomputers [14]).

4 Using ESPRESO as Linear Solver in Third-Party Tools

This section describes the basic usage of the ESPRESO solver in other tools. At first the compilation and linking issues are described. Then, all provided functions are listed and explained. The last section describes parameters for controlling the solver to achieve optimal performance.

4.1 Compilation and Linking

ESPRESO is distributed under the BSD license. The source files can be obtained from a public repository [15]. The documentation [16] contains installation instructions for the latest version. Following the instructions, besides the main executable files, the `feti4i` library is also compiled. The header file with Fortran module files are in the directory `ESPRESO_ROOT/include`. After successful installation the library `libfeti4i.so` (`libfeti4i.a` in the case of static linking) is placed into the directory `ESPRESO_ROOT/lib`.

4.2 Description of API Methods

Methods in API are designed to be as simple as possible to provide easy usage together with sufficient variability allowing optimal settings of the FETI solver. Data are assumed to be decomposed by MPI processes. Each process passes only its local data together with information about the neighboring processes. The FETI solver works in two phases:

- **Preprocessing** - in this phase the problem is decomposed into subdomains, the mesh prepares required TFETI/HTFETI data, and the assembler generates the matrices. The matrices are passed to the solver, which performs the rest of the preprocessing (calculate CP, prepare HTFETI objects, prepare communication buffers, etc.)
- **Solve** - the second phase runs the iterative solver.

According to the phases above, the API provides methods `FETI4ICreateInstance` and `FETI4ISolve`. Both methods have to be called on all MPI processes as they use MPI collective operations. The `FETI4ICreateInstance` method creates an instance with preprocessed data. The method accepts the following parameters:

- **matrix** - the stiffness matrix created by `FETI4ICreateStiffnessMatrix` and composed from finite elements using method `FETI4IAddElement`. The matrix is composed from finite elements matrices which allows the mesh later decomposition and preparation of objects for the HTFETI.
- **rhs** - the right-hand side is passed as a single vector. The size has to correspond with the size of the stiffness matrix.
- **l2g** - local to global mapping of the Degrees Of Freedom (DOF) is used for identification of interfaces between clusters. The size has to correspond with the size of the stiffness matrix. Global indices have no restriction except that the same DOF on different processes have to get the same number.
- **neighbors** - the list of neighboring processes.
- **dirichlet_indices**, **dirichlet_values** - the Dirichlet boundary condition. Indices are in local addressing. Its values should be consistent across neighboring MPI processes - shared DOF should have the same value on all neighboring processes.

- **options** - vectors of integer and real values. Best practice is to manually update only required values, after their setting to default values using methods `FETI4ISetDefaultIntegerOptions` and `FETI4ISetDefaultRealOptions`.

The created instance holds the data created during the preprocessing. Hence, changes in the input data have no effect on the instance, and input vectors can be safely removed. In order to change input data, API provides update methods that correctly updates the instance (e.g. for solving transient problems). The instance can be solved by `FETI4ISolve`.

Before the creation of an instance, a stiffness matrix has to be created. The matrix contains information about its type (symmetric positive definite, symmetric indefinite, unsymmetric) and indexing (usually 1 for Fortran and 0 for C/C++). The initially empty stiffness matrix is filled by element data. In order to allow later decomposition each element is passed with its type, node indices, DOF indices, and element stiffness matrix. The type is equal to element dimension (volume = 3, plane = 2, line = 1) and the size of the element stiffness matrix has to be `dofsSize x dofsSize`. Addition of elements is not thread safe.

When the solver in the ESPRESO library is finished the data should be destroyed by `FETI4IDestroy`. A simple example of usage of the library can be found in the ESPRESO main repository in the file `src/app/apitest.cpp`. This example is the best source of instructions for how to use the latest release of the library.

4.3 Solver Configuration Parameters

The performance of the FETI solver depends on various parameters. Parameters are passed to the library during the creation of the instance. To avoid setting all solver parameters manually, ESPRESO provides methods that set all parameters to their default values. Default settings are given by the library, however, they can be changed in the ESPRESO Configuration File (`ecf`). Whenever ESPRESO is started, it tries to read the `espresso.ecf` file. If the file exists, the default settings are changed according to the settings in the file (default `espresso.ecf` file. An example is shown in Listing 1.1).

Listing 1.1. Default `espresso.ecf` file

```

FETI4ILIBRARY {
  DOMAINS 0; // Automatic decomposition
  SOLVER {
    // TOTAL_FETI, HYBRID_FETI
    METHOD HYBRID_FETI;
    // PCG, PIPEPCG, ORTHOGONALPCG, GMRES, BICGSTAB
    ITERATIVE_SOLVER PCG;
    // NONE, LUMPED, WEIGHT_FUNCTION, DIRICHLET
    PRECONDITIONER DIRICHLET;
    PRECISION 1E-05;
    MAX_ITERATIONS 200;
  }
}

```

5 Case Study with CSC ELMER

In this section the usage of the ESPRESO library is demonstrated with the open-source multiphysical simulation software Elmer [17] developed at CSC - IT Center for Science in Finland. It contains interfaces to various direct and iterative linear solvers (MUMPS, HYPRE, PARDISO and Trilinos). ELMER also contains its own implementation of the FETI solver.

However, the Elmer FETI solver does not support multilevel decomposition. This limits its optimal hardware utilization. In the current version, an interface for the ESPRESO library was added. Results in Sect. 6 show that the ESPRESO HTFETI solver is faster than the original FETI implementation.

Elmer is available on GitHub [18]. For compilation with the ESPRESO library one has to set the environment variable FETI4I_ROOT to the directory with the ESPRESO installation and build Elmer with the flag WITH_FETI4I set to TRUE. The build directory BUILD_DIR/fem/tests/WinkelBmNavierFETI4I contains a use case to test the ESPRESO library which is used in this paper (the directory is created during installation). The running of Elmer is controlled by *.sif files. To turn the ESPRESO solver on, the following two parameters have to be set in the sif file:

- Linear System Solver = Direct
- Linear System Direct Method = FETI4I.

5.1 Automatic Multilevel Decomposition

As mentioned previously ELMER contains an implementation of the FETI solver. It is used as reference solver for performance comparison. Based on our measurements for the 3D linear elasticity benchmark (the Winkel benchmark from Elmer examples), this is the most scalable solver as is also shown in Fig. 4. This has also been confirmed with ELMER developers.

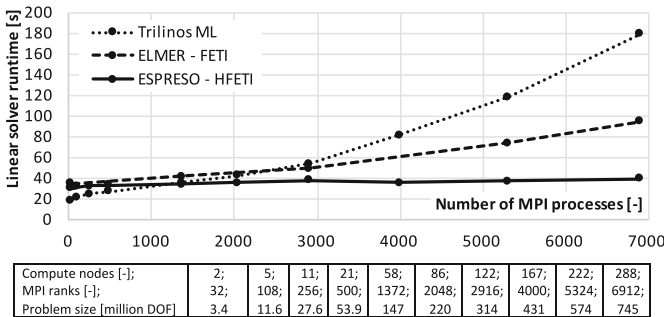


Fig. 4. Comparison of Elmer FETI, ESPRESO FETI, and Trilinos ML solvers.

The main bottleneck of the ELMER FETI solver is that it can only process one subdomain per MPI process. Due to the fact that the FETI solver needs to

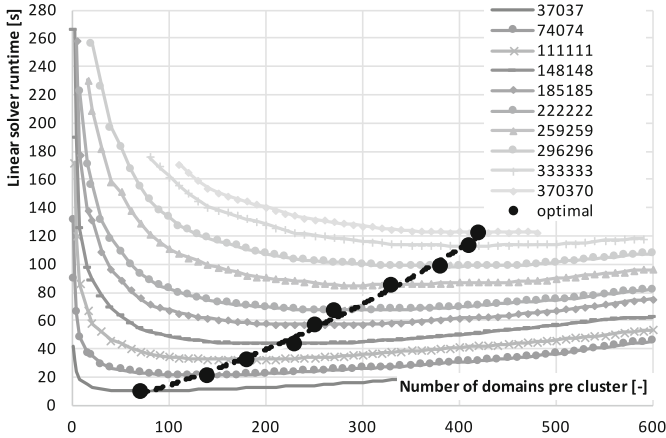


Fig. 5. Effect of the second level decomposition for 10 different cluster sizes in the range 37037–370370 DOF on HTFETI solver total runtime. To suppress the effect of the OpenMP parallelization, the benchmark ran with one thread per MPI.

do factorization of the subdomain stiffness matrices $\tilde{\mathbf{K}}_i$ the size of subdomain is limited. For instance a compute node of the IT4I Salomon supercomputer with 128 GB RAM and 2x12 CPU cores running 24 MPI processes per node can solve approximately 100,000 DOF per MPI rank and 2,400,000 DOF per node for 3D linear elasticity. However using such large subdomains uses an unnecessarily high amount of RAM as well as compute time. To tackle this problem ESPRESO uses multilevel decomposition and a Hybrid Total FETI solver in the following manner. ELMER defines the first level decomposition into clusters - one cluster per MPI process. ESPRESO then internally decomposes each cluster into subdomains, this is the second level decomposition.

The second level decomposition is a critical parameter that has the most significant effect on the HTFETI solver performance as presented in Fig. 5. The figure shows solver runtime for 10 different cluster sizes going from 37037 DOF to 370370 DOF, and second level decomposition going from 1 to 512 subdomains per cluster. The speedup that can be achieved is **7.6** for 37037 DOF cluster and **6.3** for 74074 DOF cluster. Larger clusters cannot even be solved without the second level decomposition.

To make the usage of the HTFETI solver more transparent to the user we have implemented a simple model that estimates optimal subdomain size based on the size of the cluster. This model is created based on the measurements that were used to generate Fig. 5, and its validation is shown in Fig. 6. We have created two different models for 3D linear elasticity physics.

- Preprocessing model - contains HTFETI solver preprocessing and Dirichlet preconditioner calculation,
- Single iteration model - contains single iteration of the PCG solver with Dirichlet preconditioner.

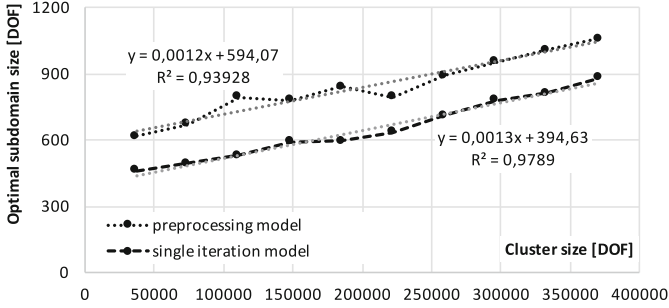


Fig. 6. Optimal subdomain size with various cluster sizes.

Table 1. The effect of using the preprocessing model for single iteration runtime.

DOF per cluster	37k	74k	111k	148k	185k	222k	259k	296k	333k	370k
Opt. for iterations [s]	0.058	0.124	0.189	0.255	0.335	0.392	0.496	0.573	0.659	0.713
Opt. for preprocessing [s]	0.059	0.127	0.196	0.259	0.343	0.402	0.496	0.576	0.663	0.717
Runtime variation	2.0%	2.1%	3.3%	1.5%	2.3%	2.5%	0.0%	0.5%	0.5%	0.7%

Table 2. The effect of using the single iteration model for preprocessing.

DOF per cluster	37k	74k	111k	148k	185k	222k	259k	296k	333k	370k
Opt. for preprocessing [s]	4.02	8.56	13.22	17.71	23.34	27.78	35.19	40.99	46.82	50.68
Opt. for iterations [s]	4.17	8.91	13.79	18.56	23.68	28.00	35.19	41.13	48.19	52.25
Runtime variation	3.6%	3.9%	4.1%	4.6%	1.5%	0.8%	0.0%	0.3%	2.9%	3.0%

From Fig. 6 one can see that both models can be described by simple linear interpolation and that for one cluster size they point to a different optimal subdomain size. Please note, that one cannot create a single model because different problems of an identical size require a different number of iterations to find the solution.

However, we have evaluated how the optimal second level decomposition for preprocessing affects the runtime of the single iteration, see Table 1, and vice versa, see Table 2. The key information lies in the last row of both tables which shows that the maximum error is less than 5% if either of these models is used. As we use the average value from these two models, prediction lies always within this 5% error.

6 Results

This section compares results of the FETI method implemented in Elmer with the HTFETI from ESPRESO. Performance was measured on the Salomon supercomputer at IT4Innovations National Supercomputing Center. Salomon consists of 1008 compute nodes (2 x Intel Xeon E5-2680v3, 2.5 GHz, 12 cores; 128 GB

DDR4@2133 MHz) interconnected by InfiniBand FDR56/7D hypercube. A 3D linear elasticity `winkel` benchmark from the Elmer repository was used for weak scalability evaluation. Tests compare both implementations on 4 - 13500 MPI processes/1 - 614 compute nodes with a constant problem size per MPI process. Only 22 cores from each node were used and one core per socket is left for MPI and system processes. Even though ESPRESO supports thread parallelization through OpenMP, all measurements were performed using only one thread per core. This ensured consistent conditions for both implementations.

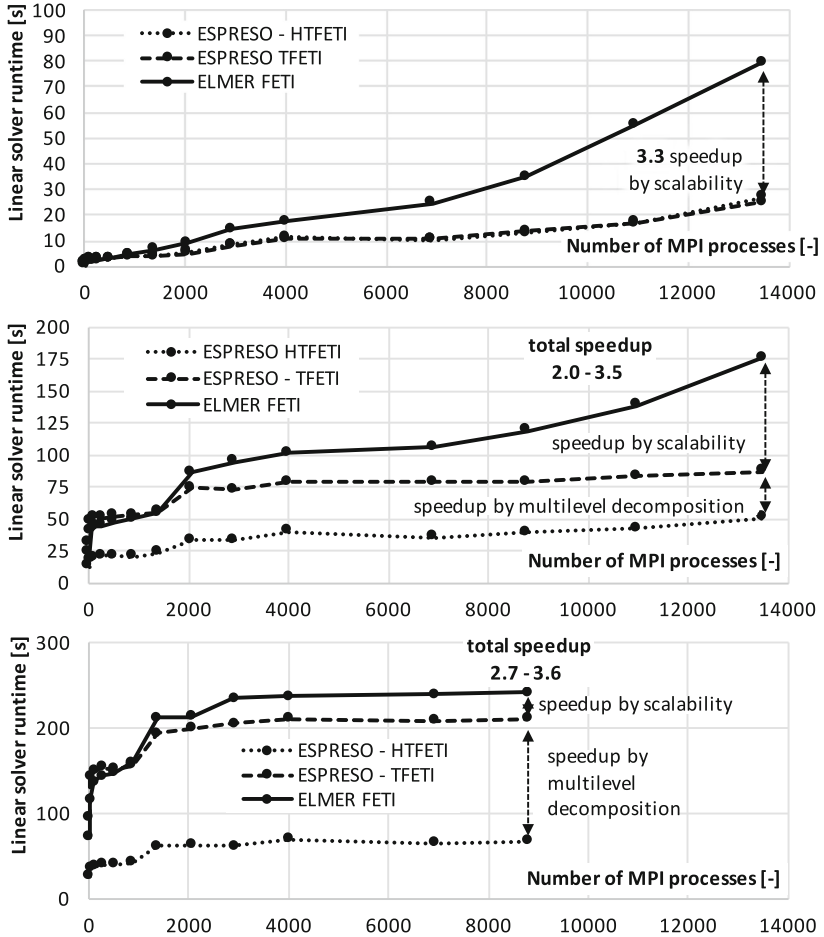


Fig. 7. Weak scaling of small subdomains with 9 000 DOF per MPI process (top), larger subdomains with 110 000 DOF per MPI process (middle) and large subdomains with 206 000 DOF per MPI process (bottom).

Figure 7 (top) shows solver performance when processing small subdomains (9 000 DOF per MPI process). In this case TFETI and HTFETI from ESPRESO

have the same performance because second level decomposition is not necessary. However one can observe that ESPRESO solvers scale better than the ELMER FETI solver, and can reach an up to 3.3 speedup - we call this scalability speedup, and it is defined as the difference between ELMER FETI and ESPRESO TFETI as they both process a single domain per MPI process.

Figure 7 (middle) shows the performance for subdomains of 110 000 DOF per MPI process. In this case one can observe two causes of performance improvements. The speedup by scalability is still present, but additionally the effect of the second level decomposition becomes equally important, as demonstrated by the difference between ESPRESO TFETI and ESPRESO HTFETI.

Figure 7 (bottom) shows the case of very large subdomains - 206 000 DOF per MPI process. Here the scalability speedup is even less important and the overall speedup is mostly gained by the second level decomposition. For the largest problems (8 788 MPI processes and 1.82 billion DOF) ESPRESO is up to 3.6 times faster. We were not able to run larger tests on more than 8788 MPI processes because Elmer does not support 64 bit integers.

7 Conclusion

Our previous work [14] has shown that the HTFETI solver scales on several of the world's largest supercomputers. To provide our scalable implementation to all potential users a simple C API was developed and presented. As proof of concept we interfaced ESPRESO with the CSC ELMER solver, and compared the HTFETI with the ELMER FETI solver. HTFETI performs two level decomposition, which significantly improves both memory utilization and solver performance.

As the optimal setup of the second level decomposition is key to achieving the fastest time to solution, we developed a performance model that controls decomposition automatically. This model is designed for applications that run one MPI process per CPU core, as ELMER does. This is a major simplification for all ELMER users, who do not have to be aware of this additional complexity and yet have the full benefit of it.

In Sect. 6 we show that the ESPRESO HTFETI solver is up to 3.7 times faster than the ELMER FETI solver when running on 13 500 MPI processes (the 614 compute nodes of the Salomon supercomputer) and solving a 1.5 billion DOF problem of 3D linear elasticity. This is caused by both lower parallel overhead (see Fig. 7 - "speedup by scalability") and optimal second level decomposition (see "speedup by multilevel decomposition" in the same figure).

Acknowledgement. This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070".

References

1. Farhat, C., Roux, F.X., Oden, J.T.: *Implicit Parallel Processing in Structural Mechanics*. Elsevier Science, Amsterdam (1994)

2. Farhat, C., Mandel, J., Roux, F.X.: Optimal convergence properties of the FETI domain decomposition method. *Comput. Method Appl. Mech. Eng.* **115**, 365–385 (1994)
3. Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., Rixen, D.: FETI-DP: a dual-primal unified FETI method, part I: a faster alternative to the two-level FETI method. *Int. J. Numer. Meth. Eng.* **50**(7), 1523–1544 (2001)
4. Klawonn, A., Widlund, O.B., Dryja, M.: Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM J. Numer. Anal.* **40**, 159–179 (2002)
5. Dostál, Z., Horák, D., Kučera, R.: Total FETI - an easier implementable variant of the FETI method for numerical solution of elliptic PDE. *Commun. Numer. Methods Eng.* **196**, 1155–1162 (2006)
6. Klawonn, A., Rheinbach, R.: Highly scalable parallel domain decomposition methods with an application to biomechanics. *ZAMM* **1**, 5–32 (2010)
7. Klawonn, A., Lanser, M., Rheinbach, O.: Toward extremely scalable nonlinear domain decomposition methods for elliptic partial differential equations. *SIAM J. Sci. Comput.* **37**(6), C667–C696 (2015)
8. Brzobohatý, T., Jarošová, M., Kozubek, T., Menšík, M., Markopoulos, A.: The hybrid total FETI method. In: *Proceedings of the Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*. Civil-Comp, Ltd. (2011)
9. Intel: Math kernel library. <https://software.intel.com/en-us/mkl>
10. Petra, C.G., Schenk, O., Lubin, M., Gärtner, K.: An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization. *SIAM J. Sci. Comput.* **36**(2), C139–C162 (2014)
11. Petra, C.G., Schenk, O., Anitescu, M.: Real-time stochastic optimization of complex energy systems on high-performance computers. *IEEE Comput. Sci. Eng.* **16**(5), 32–42 (2014)
12. Suzuki, A., Roux, F.X.: A dissection solver with kernel detection for symmetric finite element matrices on shared memory computers. *Int. J. Numer. Methods Eng.* **100**(2), 136–164 (2014)
13. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* **20**(1), 359–392 (1998)
14. Říha, L., Brzobohatý, T., Markopoulos, A., Meca, O., Kozubek, T.: Massively parallel hybrid total FETI (HTFETI) solver. In: *Proceedings of the Platform for Advanced Scientific Computing Conference, PASC 2016*, pp. 7:1–7:11. ACM, New York (2016)
15. ESPRESO: Public repository. <https://github.com/It4innovations/espreso>
16. ESPRESO: Documentation. <http://espreso.it4i.cz/>
17. Elmer: CSC - IT Center for Science. <https://www.csc.fi/web/elmer>
18. Elmer: Public repository. <https://github.com/elmercsc/elmerfem>



MERIC and RADAR Generator: Tools for Energy Evaluation and Runtime Tuning of HPC Applications

Ondrej Vysocky¹(✉), Martin Beseda¹, Lubomír Říha¹, Jan Zapletal¹,
Michael Lysaght², and Venkatesh Kannan²

¹ IT4Innovations National Supercomputing Center,
VŠB-Technical University of Ostrava, Ostrava, Czech Republic
{ondrej.vysocky,martin.beseda,lubomir.riha,jan.zapletal}@vsb.cz

² Irish Centre for High End Computing, Dublin, Ireland
{michael.lysaght,venkatesh.kannan}@ichec.ie
<http://www.it4i.cz>, <https://www.ichec.ie>

Abstract. This paper introduces two tools for manual energy evaluation and runtime tuning developed at IT4Innovations in the READEX project. The MERIC library can be used for manual instrumentation and analysis of any application from the energy and time consumption point of view. Besides tracing, MERIC can also change environment and hardware parameters during the application runtime, which leads to energy savings.

MERIC stores large amounts of data, which are difficult to read by a human. The RADAR generator analyses the MERIC output files to find the best settings of evaluated parameters for each instrumented region. It generates a \LaTeX report and a MERIC configuration file for application production runs.

Keywords: READEX · MERIC · RADAR
Energy efficient computing · HDEEM · RAPL

1 Introduction

The Horizon 2020 project READEX (Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing) [18] deals with manual and also automatic tools that analyze High Performance Computing (HPC) applications, and searches for the best combination of tuned parameter settings to use them optimally for application needs. This paper presents tools developed in the READEX project for manual evaluation of the dynamic behavior of the HPC applications - the MERIC and RADAR generator.

The MERIC library evaluates application behavior in terms of resource consumption, and controls hardware and runtime parameters such as the Dynamic Voltage and Frequency Scaling (DVFS), Uncore Frequency Scaling (UFS), and

number of OpenMP threads through external libraries. User applications can be instrumented using the MERIC manual instrumentation to analyze each part of the code separately. The energy measurements are provided by the High Definition Energy Efficiency Monitoring (HDEEM) system [8], or by Running Average Power Limit (RAPL) counters [10].

The MERIC measurement outputs are analyzed using the RADAR generator, which produces detailed reports, and also a MERIC configuration file, which can be used to set the best parameter values for all evaluated regions in the application.

There are several research activities in HPC application energy saving due to applying power capping [6, 11] to the whole application run instead of parsing the application into regions and applying dynamic tuning. Other research is dealing with scheduling system using dynamic power capping with negligible time penalty based on previous application runs [16]. Dynamic application tuning is the goal of the READEX project, which should deliver a tool-suite for fully automatic application instrumentation, dynamism detection and analysis. The analysis should find the configuration that provide the maximum energy savings and can be used for the future production runs. The READEX tools are very complex and may not be easy to apply. Our tools present the same approach with focus on usage friendliness, albeit providing manual tuning only. Furthermore, the READEX tools are focused on x86 platforms only, which is not the case for MERIC.

2 Applications Dynamism

The READEX project expects that HPC applications have different needs in separate parts of the code. To find these parts inside a user application, three dynamism metrics are presently measured and used in the READEX project. They include:

1. Execution time
2. Energy consumed
3. Computational intensity

Among these three metrics, the semantics of execution time and energy consumed are straightforward. Variation in the execution time and energy consumed by regions in an application during its execution is an indication of different resource requirements. The computational intensity is a metric that is used to model the behaviour of an application based on the workload imposed by it on the CPU and the memory. Presently, computational intensity is calculated using the following formula 1 and is analogous to the operational intensity used in the roofline model [22].

$$\text{Computational intensity} = \frac{\text{Total number of instructions executed}}{\text{Total number of L3 cache misses}} \quad (1)$$

Selected regions in the user application are called significant. To detect the significant regions manually, profiling tools such as Allinea MAP [1] are used.

The dynamism observed in an application can be due to variation of the following factors:

- Floating point computations (for example, this may occur due to variation in the density of matrices in dense linear algebra).
- Memory read/write access patterns (for example, this may occur due to variation in the sparsity of matrices in sparse linear algebra).
- Inter-process communication patterns (for example, this may occur due to irregularity in a data structure leading to irregular exchange of messages for operations such as global reductions).
- I/O operations performed during the application’s execution.
- Different inputs to regions in the application.

To address these factors, a set of tuning parameters has been identified in the READEX project to gain possible savings due to static and dynamic tuning. The list of the parameters contains the following:

- hardware parameters of the CPU
 - Core Frequency (CF)
 - Uncore frequency (UCF)¹
- system software parameters
 - number of OpenMP threads, thread placement
- application-level parameters
 - depends on the specific application

All parameters can be set before an application is executed (this is called static tuning), in addition some of them can be tuned dynamically during the application runtime. For instance core and uncore frequencies can be switched without additional overhead, but switching the number of threads can affect performance due to NUMA effects and data placement and must be handled carefully. Static and dynamic tuning leads to static and dynamic savings, respectively.

Presently the MERIC tool (Sect. 3) is being developed and used in the READEX project to measure the above-mentioned dynamism metrics and evaluate applications. When using MERIC it is possible to dynamically switch CPU core and uncore frequencies and the number of used OpenMP threads. The measurements collected by these tools for an application are logged into a READEX Application Dynamism Analysis Report (RADAR) as described in Sect. 4.

3 Manual Dynamism Evaluation with MERIC

MERIC² is a C++ dynamic library (with an interface for Fortran applications) that measures energy consumption and runtime of annotated regions inside

¹ Uncore frequency refers to frequency of subsystems in the physical processor package that are shared by multiple processor cores, e.g., L3 cache and on-chip ring interconnect.

² MERIC repository: <https://code.it4i.cz/vys0053/meric>.

a user application. By running the code with different settings of the tuning parameters, we analyze possibilities for energy savings. Subsequently, the optimal configurations are applied by changing the tuning parameters (list of parameters mentioned in the previous Sect. 2) during the application runtime, which can be also done by using MERIC. MERIC wraps a list of libraries that provide access to different hardware knobs and registers, operating system and runtime system variables, i.e. tuning parameters, in order to read or modify their values. The main motivation for the development of this tool was to simplify the evaluation of various applications dynamic behavior from the energy consumption point of view, which includes a large number of measurements.

The library is easy to use. After inserting the MERIC initialization function, it is possible to instrument the application through the so-called probes, which wrap potentially significant regions of the analysed code. Besides storing the measurement results, the user should not notice any changes in the behavior of the application.

3.1 MERIC Features

MERIC has minimal influence on the application's runtime despite providing several analysis and tuning features. Its overhead depends on the energy measurement mode as described in this section, the amount of hardware performance counters read, as well as the number of instrumented regions.

Environment Settings

During the MERIC initialization and at each region start and end, the CPU frequency, uncore frequency and number of OpenMP threads are set. To do so, MERIC uses the OpenMP runtime API and the `cpufreq` [3] and `x86_adapt` [17] libraries.

Energy Measurement

The key MERIC feature is energy measurement using the High Definition Energy Efficiency Monitoring (HDEEM) system located directly on computational nodes that records 100 power samples per second of the CPUs and memories, and 1000 samples of the node itself via the BMC (Baseboard Management Controller) and an FPGA (Field Programmable Gate Array). Figure 1 shows the system diagram and a picture a node with the HDEEM.

HDEEM provides energy consumption measurement in two different ways, and in MERIC it is possible to choose which one the user wants to use by setting the `MERIC_CONTINUAL` parameter.

In one mode, the energy consumed from the point that HDEEM was initialized is taken from the HDEEM Stats structure (a data structure used by the HDEEM library to provide measurement information to the user application). In this mode we read the structure at each region start and end. This solution is straightforward, however, there is a delay of approximately 4 ms associated with every read from the HDEEM API. To avoid the delay, we take advantage of the fact that during measurement HDEEM stores power samples in its internal memory. In the second mode MERIC only needs to record timestamps at the

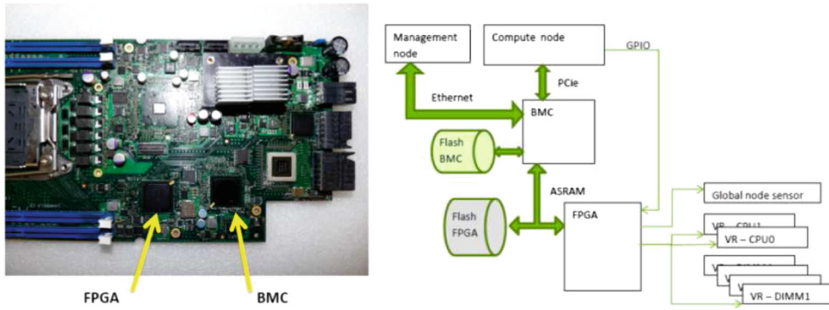


Fig. 1. A HDEEM system located on a node and the system diagram [2].

beginning and the end of each region instead of calling the HDEEM API. This results in a very small overhead for MERIC instrumentation during the application runtime because all samples are transferred from the HDEEM memory at the end of the application runtime. The energy consumption is subsequently calculated from the power samples based on the recorded timestamps.

Contemporary Intel processors support energy consumption measurements via the Running Average Power Limit (RAPL) interface. MERIC uses the RAPL counters with 1 kHz sampling frequency to allow energy measurements on machines without the HDEEM infrastructure as well as to compare them with the HDEEM measurements.

The main disadvantage of using RAPL is that it measures CPUs and memories power consumption only, without providing information about the power consumption of the blade itself. In the case of nodes with two Intel(R) Xeon(R) CPU E5-E5-2680 v3 (2×12 cores) processors the power baseline is approximately 70 W. To overcome this handicap we statically add this 70 W to our measurements when using RAPL counters. MERIC uses the `x86_adapt` library to read the RAPL counters.

The minimum runtime of each evaluated region has been set in the READDEX project to 100 ms when using HDEEM or RAPL, to have enough samples per region to evaluate the region optimum configuration correctly.

Hardware Performance Counters

To provide more information about the instrumented regions of the application, we use the `perf_event` and `PAPI` libraries, which provide access to hardware performance counters. Values from the counters are transferred into cache-miss rates, FLOPs/s³ and also the computational intensity that is a key metric for dynamism detection as described in Sect. 2.

³ The Intel Haswell processors do not support floating-point instructions counters. MERIC approximates FLOPs/s based on the counter of Advanced Vector Extensions (AVX) calculation operations. For more information visit <https://github.com/RRZE-HPC/likwid/wiki/FlopsHaswell>.

Shared Interface for Score-P

The Score-P software system, as well as the MERIC library, allows users to manually (and also automatically) instrument an application for tracing analysis. Score-P instrumentation is also used in the READEX tool suite [13].

A user that has already instrumented an application using Score-P instrumentation or would want to use it in the future may use the `readex.h` header file that is provided in the MERIC repository. This allows the user to only insert the user instrumentation once, but for both MERIC and Score-P simultaneously. When a user application is compiled, one has to define the preprocessor variables `USE_MERIC`, `USE_SCOREP` (Score-P phase region only) or alternatively `USE_SCOREP_MANUAL` to select which instrumentation should be used.

Table 1 shows the list of functions defined in the header file, with their MERIC and Score-P equivalents. Brief description of the mentioned MERIC functions is provided in Sect. 3.2, description of the Score-P functions can be found in its user manual [20].

Table 1. Function names defined in the `readex.h` header file, that can be used for MERIC and Score-P instrumentation.

Shared interface	MERIC function	Score-P function
<code>READEX_INIT</code>	<code>MERIC_INIT</code>	–
<code>READEX_CLOSE</code>	<code>MERIC_CLOSE</code>	–
<code>READEX_REGION_DEFINE</code>	–	<code>SCOREP_USER_REGION_DEFINE</code>
<code>READEX_REGION_START</code>	<code>MERIC_MeasureStart</code>	<code>SCOREP_USER_REGION_BEGIN</code>
<code>READEX_REGION_STOP</code>	<code>MERIC_MeasureStop</code>	<code>SCOREP_USER_REGION_END</code>
<code>READEX_PHASE_START</code>	<code>MERIC_MeasureStart</code>	<code>SCOREP_USER_OA_PHASE_BEGIN</code>
<code>READEX_PHASE_STOP</code>	<code>MERIC_MeasureStop</code>	<code>SCOREP_USER_OA_PHASE_END</code>

MERIC Requirements

MERIC currently adds synchronization MPI and OpenMP barriers into the application code to ensure that all processes/threads under one node are synchronized in a single region when measuring consumed resources or changing hardware or runtime parameters. We realize that this approach inserts extra overhead into application runtime and may discriminate a group of asynchronous applications. In future the library will allow the user to turn these barriers off.

Beyond the inserted synchronization the MERIC library requires several libraries to provide all previously mentioned features:

- Machine with HDEEM or `x86_adapt` library for accessing RAPL counters
- `Cpufreq` or `x86_adapt` library to change CPU frequencies
- PAPI and `perf_event` for accessing hardware counters

ARM Jetson TX1

The MERIC library was originally developed to support resource consumption measurement and DVFS on Intel Haswell processors [9], however it has

been extended to also provide support for the Jetson/TX1 ARM system [12] located at the Barcelona Supercomputing Center [14] (ARM Cortex-A57, 4 cores, 1.3 GHz) which supports energy measurements.

ARM systems are an interesting platform because they allow the setting of much lower frequencies [7] and save energy accordingly. In the case that system CPU uncore frequency is not possible to set, however, one can change the frequency of the RAM. Minimum CPU core frequency is 0.5 GHz and the maximum is 1.3 GHz. The minimum and maximum RAM frequency is 40 MHz and 1.6 GHz, respectively. To change frequencies on Jetson, no third-party libraries are necessary.

To gather power data, the Texas Instrument INA3221 chip is featured on the board [4]. It measures the per-node energy consumption and stores samples values in a file. It is possible to gather hundreds of samples per second, however the measurement effects the CPU. The following Table 2 shows the impact of sampling frequency on the CPU workload evaluated using htop⁴.

Table 2. The Jetson/TX1 energy measurement interface and its effect on the CPU workload when reading 10 up to 1000 power samples per second. The load was evaluated using htop when running the power sampling only.

Sampling frequency [Hz]	CPU workload
10	2%
50	4%
100	8%
200	14%
500	23%
1000	30%

3.2 Workflow

First, the user has to analyze their application using a profiler tool (such as Allinea MAP) and find the significant regions in order to cover the most consuming functions in terms of time, MPI communication, and I/O, and insert MERIC instrumentation into code to wrap the selected sections of the code. A region start function takes a parameter with the name of the region, but the stop function does not have any input parameters, because it ends the region that has been started most recently (last in, first out).

The instrumented application should be run as usual. To control MERIC behaviour it is possible to export appropriate environment variables or define a MERIC configuration file that allows the user to specify the settings not only for the whole application run (as in the case of environment variables), but also control the behavior for separate regions, computation nodes, or their sockets.

⁴ htop repository: <https://github.com/hishamhm/htop>.

The user can define hardware and runtime settings (CPU frequencies and number of threads) as well as select energy measurement mode, hardware counters to read and more.

4 RADAR: Measurement Data Analysis

RADAR presents a brief summary of the measurement results obtained with MERIC. This is a merged form of automatically generated dynamism report by both the RADAR generator (by IT4Innovations), described in detail in Sect. 4.1 and the `readex-dyn-detect` (by the Technical University of Munich), described in [19]. The report depicts diagrams of energy consumption with respect to a set of tuning parameters. It also contains different sets of graphical comparisons of static and dynamic significant energy savings across phases for different hardware tuning parameter configurations. In each perspective, the measured dynamism metrics are presented for the default configurations that are used for the tuning parameters.

4.1 The RADAR Generator

The RADAR generator⁵ allows users to evaluate the data measured by the MERIC tool automatically, and to get an uncluttered summary of the results in the form of a \LaTeX file. Moreover, it is possible to include the report generated by the `readex-dyn-detect` tool, as mentioned above.

Table 3. Heat map generated by the RADAR generator comparing impact of using different CPU core and uncore frequencies at application runtime in seconds.

$\frac{\text{Uncore freq [GHz (uncore)]}}{\text{Frequency [GHz (core)]}}$	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	12.256	11.071	10.633	10.084	9.407	8.937	9.284	8.581	8.513	8.296
1.4	11.829	10.57	10.152	9.178	8.682	8.684	8.094	8.192	7.966	7.666
1.6	11.723	10.178	9.438	8.706	8.373	8.008	7.821	7.471	7.552	7.212
1.8	10.996	9.969	8.952	8.57	7.929	7.779	7.477	7.138	7.085	6.93
2	10.607	9.516	8.925	8.203	7.79	7.356	7.096	6.908	6.802	6.744
2.2	10.23	9.734	9.02	7.977	7.5	7.23	7.129	6.778	6.827	6.361
2.4	10.775	9.438	8.416	7.919	7.367	7.208	6.772	6.577	6.436	6.356
2.5	10.798	9.086	8.366	7.856	7.555	7.072	6.66	6.605	6.257	6.107

The report itself contains information about both static and dynamic savings, represented not only by tables, but also plots and heat-maps. Examples can be seen in Fig. 3 and Table 3.

The generator is able to evaluate all chosen quantities at once, i.e. users do not have to generate reports for energy consumption, and compute intensity and execution time separately, because they can be contained in one report together.

⁵ RADAR generator repository: <https://code.it4i.cz/bes0030/readex-radar>.

This provides the advantage of direct visual comparison of all optimal settings, so users can achieve a greater understanding of the application behavior quickly. The execution time change for energy-optimal settings is also included in the report, as can be seen in Table 4.

Table 4. Summary table generated by the RADAR generator presenting possible energy or runtime saving that can be reached if the best static and also best dynamic settings for each region would be set.

Overall application evaluation					
	Default settings	Default values	Best static config.	Static savings	Dynamic savings
Energy consumption [J] (Samples), Blade summary	24 threads, 3.0 GHz UCF, 2.5 GHz CF	2473.63 J	12 threads, 3.0 GHz UCF, 2.5 GHz CF	371.80 J (15.03%)	4.87 J of 2101.83 J (0.23%)
Runtime of function [s], Job info - hdeem	24 threads, 3.0 GHz UCF, 2.5 GHz CF	6.37 s	18 threads, 3.0 GHz UCF, 2.5 GHz CF	0.26 s (4.10%)	0.0073 s of 6.11 s (0.12%)

Run-time change with the energy optimal settings: -0.01 s (98.19% of default time)

This evaluation is performed not only for the main region (usually the whole application), but for its nested regions too. Users can also specify an iterative region which contains all the nested ones and which is called directly in the main region. In this way certain iterative schemes (e.g., iterative solvers of linear systems) are understood in detail, because every iteration (or phase) is evaluated separately.

With this feature users have information about the best static optima just for the main region (which serves as the best starting settings), information about optimal settings of nested regions in an average phase, and the above-mentioned information about optimal settings of nested regions in every individual phase. If we wanted to process multiple regions like one, we can group them under one *role*, as can be seen in Fig. 2, where *Projector_l* and *Projector_L2* are different regions comprising the region *Projector*. If multiple runs of the program are measured, then both the average run and separate runs are evaluated.

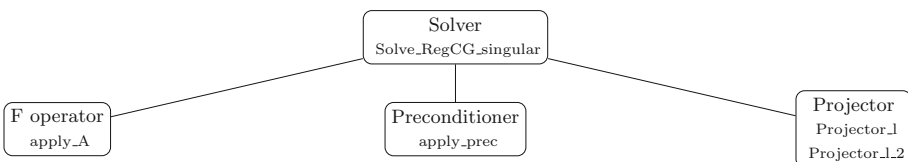


Fig. 2. Example of multiple regions on one role

For some programs such a report could be impractically long and so the generator offers the possibility to create a shorter version containing only the overall summary and the average phase evaluation.

The generator also supports evaluation in multiples of the original unit used in the measurement. Both the static and dynamic baseline for the energy consumption, i.e. the constant baseline and the baseline dependent on settings, are supported too.

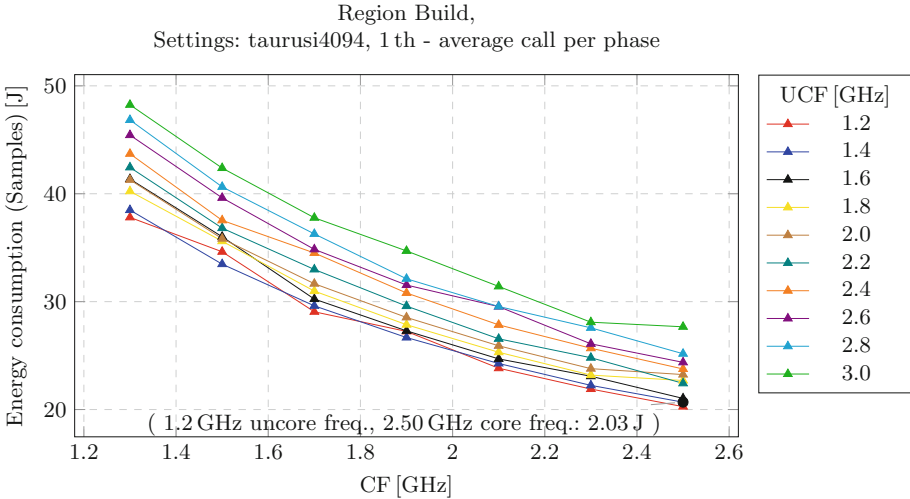


Fig. 3. Plot example generated by the RADAR generator showing the effect of using different CPU core and uncore frequencies from the energy consumption point of view.

Finally, the optimal settings for all regions and every measured quantity can be exported into the separated files, which can be used as an input for the MERIC tool, as described in Sect. 3.2.

All the above-mentioned settings are listed in the external configuration file, which is set by the generator's flag, so users can easily change several different settings for their reports.

5 Test Case

The ESPRESO library⁶ was selected to present MERIC and RADAR generator usage. The library is a combination of Finite Element (FEM) and Boundary Element (BEM) tools and TFETI/HTFETI [5, 15] domain decomposition solvers. The ESPRESO solver is a parallel linear solver, which includes a highly efficient MPI communication layer designed for massively parallel machines with thousands of compute nodes. The parallelization inside a node is done using OpenMP. Inside the application we have identified several regions of the code, that may have different optimal configuration see Fig. 4.

⁶ ESPRESO library website: <http://espresso.it4i.cz/>.

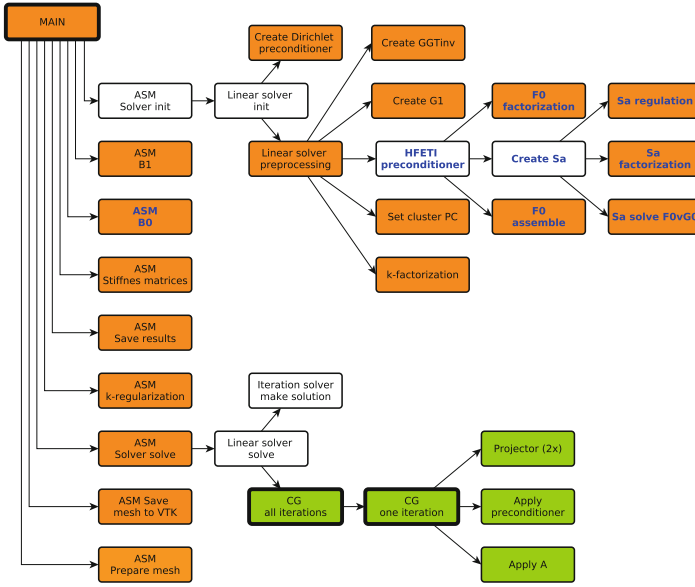


Fig. 4. Graph of significant regions in the ESPRESO library. The green boxes depict multiply called regions in an iterative solver, the orange ones are only called once during the application runtime. (Color figure online)

Table 5. Table of resultant static and dynamic savings of the ESPRESO library test. Rows respectively focus on possible savings from the energy and runtime points of view.

Overall application evaluation					
	Default settings	Default values	Best static configuration	Static savings	Dynamic savings
Energy [J] RAPL counters	12 threads, 3.0 GHz UCF, 2.5 GHz CF	4549.13 J	12 threads, 2.2 GHz UCF, 2.4 GHz CF	181.76 J (4.00%)	325.89 J of 4367.37 J (7.46%)
Runtime [s]	12 threads, 3.0 GHz UCF, 2.5 GHz CF	15.90 s	12 threads, 3.0 GHz UCF, 2.5 GHz CF	0.00 s (0.00%)	0.39 s of 15.90 s (2.43%)

The following test was performed on the IT4Innovations Salomon cluster powered by two Intel Xeon E5-2680v3 (Haswell-EP) processors per node using a RAPL counter with a 70 W baseline for the energy consumption measurement. The processor is equipped with 12 cores and allows for CPU core and uncore frequency scaling within the range of 1.2–2.5 GHz and 1.2–3.0 GHz, respectively.

We evaluated ESPRESO on a heat transfer problem with 2.7 million unknowns using one MPI process per socket.

Table 5 shows the possible savings made by using different numbers of OpenMP threads during the runtime, and by switching CPU core and uncore frequencies. This table shows that it is possible to save 4% of the overall energy just by statically setting different CPU core and uncore frequencies that can be applied even without instrumenting the application at all. Table 6 shows the impact of using different CPU frequencies in this test case, from the energy consumption point of view.

Another 7.46% of energy can be saved through dynamic switching of the tuned parameters to apply the best configuration for each significant region. Overall energy savings in this test case were 11.16%. Table 7 in the appendix of this paper contains the regions' best settings.

Table 6. An ESPRESO library energy consumption heat-map showing the impact of different CPU core and uncore frequencies when using 12 OpenMP threads.

$\frac{\text{UnCF [GHz]}}{\text{CF [GHz]}}$	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	5,971	5,825	5,764	5,725	5,698	5,783	5,859	5,962	6,127	6,232
1.4	5,519	5,350	5,238	5,220	5,208	5,219	5,357	5,432	5,513	5,639
1.6	5,226	5,029	4,902	4,840	4,829	4,819	4,890	4,986	5,093	5,185
1.8	5,080	4,897	4,739	4,711	4,649	4,656	4,720	4,760	4,859	4,956
2	5,054	4,852	4,707	4,587	4,565	4,528	4,585	4,636	4,736	4,817
2.2	4,985	4,774	4,605	4,520	4,464	4,442	4,469	4,540	4,632	4,653
2.4	4,984	4,783	4,593	4,442	4,391	4,367	4,408	4,438	4,503	4,578
2.5	5,211	4,858	4,675	4,547	4,479	4,422	4,445	4,439	4,482	4,549

6 Conclusion

The paper presented two tools that allow easy analysis of HPC applications' behavior, with the goal to tune hardware and runtime parameters to minimize the given objective (e.g., the energy consumption and runtime).

Resource consumption measurement and dynamic parameter changes are provided by the MERIC library. The currently supported parameters that can be switched dynamically include the CPU core and uncore frequencies, as well as the number of active OpenMP threads.

The RADAR generator analyses the MERIC measurement outputs and provides detailed L^AT_EX reports describing the behavior of the instrumented regions. These reports also contain information about the settings that should be applied for each region to reach maximum savings. The RADAR generator produces the MERIC configuration files that should be used for production runs of the user application to apply the best settings dynamically during the runtime.

Possible savings that can be reached when using MERIC and the RADAR generator are presented in [21], where we show that the energy savings can reach up to 10–30%.

Acknowledgement. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science - LQ1602” and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project “IT4Innovations National Supercomputing Center – LM2015070”.

The research leading to these results has received funding from the European Union’s Horizon 2020 Programme under grant agreement number 671657.

The work was additionally supported by VŠB – Technical University of Ostrava under the grant SP2017/165 and by the Barcelona Supercomputing Center under the grants 288777, 610402 and 671697.

Appendix

Table 7. Table of the regions analysis from the energy point of view for the test case presented in the Sect. 5. For every region, this table contains the percentage of energy the region consumed compared to the entire application, and each regions’ best configuration and energy savings if the configuration were applied during the application runtime in its the best static configuration.

Significant regions energy summary and its best dynamic configuration			
Region	% of 1 phase	Best dynamic configuration	Dynamic savings
Assemble Stiffness Matrices	16.77	12 threads, 2.0 GHz UCF, 2.4 GHz CF	6.51 J from 685.54 J (0.95%)
Assembler–Assemble-B0	0.2	12 threads, 2.0 GHz UCF, 2.5 GHz CF	0.10 J from 8.07 J (1.24%)
Assembler–Assemble-B1	4.12	12 threads, 2.0 GHz UCF, 2.5 GHz CF	6.61 J from 168.36 J (3.93%)
Assembler-K_Regularization	5.00	2 threads, 2.2 GHz UCF, 2.5 GHz CF	47.64 J from 204.24 J (23.32%)
Assembler–PrepareMesh	12.66	2 threads, 1.8 GHz UCF, 2.5 GHz CF	77.70 J from 517.68 J (15.01%)
Assembler–SaveMeshtoVTK	6.89	2 threads, 1.2 GHz UCF, 2.5 GHz CF	39.80 J from 281.63 J (14.13%)
Assembler–SaveResults	3.38	2 threads, 1.2 GHz UCF, 2.5 GHz CF	24.67 J from 138.34 J (17.83%)
Assembler–SolverSolve	27.92	12 threads, 2.2 GHz UCF, 1.6 GHz CF	114.50 J from 1141.58 J (10.03%)
Cluster–CreateF0-AssembleF0	5.67	12 threads, 2.2 GHz UCF, 2.4 GHz CF	0.00 J from 231.68 J (0.00%)

(continued)

Table 7. (continued)

Significant regions energy summary and its best dynamic configuration			
Region	% of 1 phase	Best dynamic configuration	Dynamic savings
Cluster-CreateG1-perCluster	0.43	12 threads, 2.2 GHz UCF, 2.0 GHz CF	0.64 J from 17.47 J (3.69%)
Create_GGT_Inv	0.21	2 threads, 2.2 GHz UCF, 2.5 GHz CF	2.01 J from 8.56 J (23.46%)
Cluster-CreateF0-FactF0	0.08	12 threads, 2.8 GHz UCF, 2.5 GHz CF	0.21 J from 3.26 J (6.36%)
Cluster-Kfactorization	14.47	12 threads, 2.2 GHz UCF, 2.4 GHz CF	0.00 J from 591.46 J (0.00%)
Cluster-CreateSa-SaFactorization	0.51	6 threads, 2.8 GHz UCF, 2.5 GHz CF	2.31 J from 20.70 J (11.30%)
Cluster-CreateSa-SolveF0vG0	0.86	6 threads, 2.8 GHz UCF, 2.5 GHz CF	3.02 J from 35.20 J (8.58%)
Cluster-SetClusterPC	0.85	12 threads, 2.4 GHz UCF, 2.4 GHz CF	0.18 J from 34.95 J (0.52%)

References

1. Allinea MAP - C/C++ profiler and Fortran profiler for high performance Linux code. <https://www.allinea.com/products/map>
2. High definition energy efficiency monitoring. <http://www.ena-hpc.org/2014/pdf/bull.pdf>
3. Brodowski, D.: Linux CPUFreq. <https://www.kernel.org/doc/Documentation/cpu-freq/index.txt>
4. BSC: Power monitoring on mini-clusters. https://wiki.hca.bsc.es/dokuwiki/wiki:prototype:power_monitor#jetson-tx1
5. Dostal, Z., Horak, D., Kucera, R.: Total FETI-an easier implementable variant of the FETI method for numerical solution of elliptic PDE. *Commun. Numer. Methods Eng.* **22**(12), 1155–1162 (2006). <https://doi.org/10.1002/cnm.881>
6. Eastep, J., et al.: Global extensible open power manager: a vehicle for HPC community collaboration on co-designed energy management solutions. In: Kunkel, J.M., Yokota, R., Balaji, P., Keyes, D. (eds.) *ISC 2017*. LNCS, vol. 10266, pp. 394–412. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58667-0_21
7. eLinux.org: Jetson/TX1 controlling performance. http://elinux.org/Jetson/TX1_Controlling_Performance
8. Hackenberg, D., Ilsche, T., Schuchart, J., Schöne, R., Nagel, W., Simon, M., Georgiou, Y.: HDEEM: high definition energy efficiency monitoring. In: *Energy Efficient Supercomputing Workshop (E2SC)*, November 2014

9. Hackenberg, D., Schöne, R., Ilsche, T., Molka, D., Schuchart, J., Geyer, R.: An energy efficiency feature survey of the Intel Haswell processor. In: 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), May 2015
10. Hähnel, M., Döbel, B., Völp, M., Härtig, H.: Measuring energy consumption for short code paths using rapl. SIGMETRICS Perform. Eval. Rev. **40**(3), 13–17 (2012). <http://doi.acm.org/10.1145/2425248.2425252>
11. Haidar, A., Jagode, H., Vaccaro, P., YarKhan, A., Tomov, S., Dongarra, J.: Investigating power capping toward energy-efficient scientific applications. *Concurr. Comput.: Pract. Exp.* e4485. <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4485>
12. NVIDIA: NVIDIA Jetson. <http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html>
13. Oleynik, Y., Gerndt, M., Schuchart, J., Kjeldsberg, P.G., Nagel, W.E.: Runtime exploitation of application dynamism for energy-efficient exascale computing (READEX). In: Plessl, C., El Baz, D., Cong, G., Cardoso, J.M.P., Veiga, L., Rauber, T. (eds.) 2015 IEEE 18th International Conference on Computational Science and Engineering (CSE), pp. 347–350. IEEE, Piscataway, October 2015
14. Rajovic, N., Rico, A., Mantovani, F., Ruiz, D., Vilarrubi, J.O., Gomez, C., Backes, L., Nieto, D., Servat, H., Martorell, X., Labarta, J., Ayguade, E., Adeniyi-Jones, C., Derradji, S., Gloaguen, H., Lanucara, P., Samma, N., Mehaut, J.F., Pouget, K., Videau, B., Boyer, E., Allalen, M., Auweter, A., Brayford, D., Tafani, D., Weinberg, V., Brömmel, D., Halver, R., Meinke, J.H., Bevide, R., Benito, M., Vallejo, E., Valero, M., Ramirez, A.: The mont-blanc prototype: an alternative approach for HPC systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016, pp. 38:1–38:12. IEEE Press, Piscataway (2016). <http://dl.acm.org/citation.cfm?id=3014904.3014955>
15. Riha, L., Brzobohaty, T., Markopoulos, A., Jarosova, M., Kozubek, T., Horak, D., Hapla, V.: Implementation of the efficient communication layer for the highly parallel total feti and hybrid total feti solvers. *Parallel Comput.* **57**, 154–166 (2016)
16. Rountree, B., Lowenthal, D.K., de Supinski, B.R., Schulz, M., Freeh, V.W., Bletsch, T.K.: Adagio: making DVS practical for complex HPC applications. In: ICS (2009)
17. Schoene, R.: x86.adapt. <https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/X86Adapt>
18. Schuchart, J., Gerndt, M., Kjeldsberg, P.G., Lysaght, M., Horák, D., Říha, L., Gocht, A., Sourouri, M., Kumaraswamy, M., Chowdhury, A., Jahre, M., Diethelm, K., Bouizi, O., Mian, U.S., Kružík, J., Sojka, R., Beseda, M., Kannan, V., Bendifallah, Z., Hackenberg, D., Nagel, W.E.: The READEX formalism for automatic tuning for energy efficiency. *Computing* 1–19 (2017). <https://doi.org/10.1007/s00607-016-0532-7>
19. Venkatesh, K., Lubomir, R., Michael, G., Anamika, C., Ondrej, V., Martin, B., David, H., Radim, S., Jakub, K., Michael, L.: Prace whitepaper: investigating and exploiting application dynamism for energy-efficient exascale computing (2017). www.prace-ri.eu
20. VI-HPS: Score-p user manual 3.1 (2017)

21. Vysocky, O., Beseda, M., Riha, L., Zapletal, J., Nikl, V., Lysaght, M., Kannan, V.: Evaluation of the HPC applications dynamic behavior in terms of energy consumption. In: Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering. Civil-Comp Press, Stirlingshire, Paper 3 (2017)
22. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* **52**(4), 65–76 (2009). <https://doi.org/10.1145/1498765.1498785>



Disc vs. Annulus: On the Bleaching Pattern Optimization for FRAP Experiments

Ctirad Matono¹(✉), Štěpán Papáček², and Stefan Kindermann³

¹ Institute of Computer Science, The Czech Academy of Sciences,
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
matono¹@cs.cas.cz

² Institute of Complex Systems, South Bohemian Research Center of Aquaculture
and Biodiversity of Hydrocenoses, Faculty of Fisheries and Protection of Waters,
University of South Bohemia in České Budějovice,
Zámek 136, 373 33 Nové Hradky, Czech Republic

³ Industrial Mathematics Institute, University of Linz,
Altenbergerstr. 69, 4040 Linz, Austria

Abstract. This study deals with the problem of optimal setting of experimental design variables, which controls the accuracy of the numerical process of determining model parameters from data. Our approach, although case independent, is formulated as an inverse problem of a diffusion coefficient estimation using the FRAP (Fluorescence Recovery After Photobleaching) experimental technique. The key concept relies on the analysis of the sensitivity of the measured output with respect to the model parameters. Based on this idea, we optimize an experimental design factor being the initial concentration of some particles. Numerical experiments on a 2D finite domain show that the discretized optimal initial condition attains only two values representing the existence or non-existence of diffusive particles. The number of jumps between these values determines the connectivity (or the bleaching pattern) and is dependent on the value of a diffusion coefficient, e.g., the annulus shaped initial condition is better than a disc for some specific range of model parameters.

Keywords: Optimization · Parameter identification · FRAP
Bleaching pattern · Initial boundary value problem
Sensitivity measure

1 Introduction

The continuous enhancement and sophistication of experimental devices in service of biological community take advantage of the fast development of informatics and image processing. However, it is not a rare case that a large amount of spatio-temporal data, e.g., in form of a time sequence of images, is routinely generated without a clear idea about further data processing.

The aim of this paper is to (re)establish the link between experimental conditions (experimental protocol) and the accuracy of the resulting data processing. Our simplified case study on FRAP (Fluorescence Recovery After Photobleaching) data processing [6, 14] serves as a paradigmatic example of the inverse problem of the diffusion parameter estimation from spatio-temporal measurements of fluorescent particle concentration. The FRAP technique is based on measuring the fluorescence intensity (proportional to non-bleached particles concentration) in a region of interest (being usually an Euclidian 2D domain) in response to a high-intensity laser pulse. The laser pulse (the so-called *bleach*) causes an irreversible loss in fluorescence of some particles residing originally in the bleached area, presumably without any damage to intracellular structures. After the bleach, we observe the change in fluorescence intensity in a monitored region reflecting the diffusive transport of fluorescent particles from the area outside the bleach [12, 17].

A natural question is how the experimental settings influence the accuracy of the resulting parameter estimates. There are many rather empirical recommendations related to the design of a photobleaching experiment, e.g., the bleach spot shape and size [2], the region of interest (its location and size), or the total time of measurement, see [16] and references within. However, we would have a more rigorous tool for the choice of experimental design factors. This is because the setting of initial conditions of an experiment not only influences the following data measurement process. There is somewhat hidden a related data processing part. Mainly in case of a model-based design of experiments and when the measured data are used in the frame of an inverse problem of model parameter estimation. Having a reliable process model, e.g. [11], we can perform the subsequent sensitivity analysis with respect to the model parameters [3]. Consequently, we are allowed to formulate the problem as the maximization of a sensitivity measure leading to the optimal initial condition (an optimal bleaching pattern).

The paper is organized as follows. In Sect. 2, we introduce the problem, define the sensitivity measure and formulate the optimization problem. Section 3 describes some numerical issues of sensitivity measure evaluation. In Sect. 4, we provide a numerical example to show that our theoretical basis is well founded and that the optimal initial condition strongly depends on the diffusion coefficient and leads to a variety of bleaching patterns. Finally, some conclusions are presented in Sect. 5.

2 Problem Formulation

Let us consider the Fickian diffusion with a *constant* diffusion coefficient $D > 0$ and assume a spatially radially symmetric observation domain, i.e., the data are observed on a cylinder with the radius R and height T [8]. Taking into account the usual case of radial symmetry of the FRAP experiment, the simplest governing equation for the spatio-temporal distribution of fluorescent particle concentration $u(r, t)$ is the diffusion equation as follows

$$\frac{\partial u}{\partial t} = D \left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} \right), \quad (1)$$

where $r \in (0, R]$, $t \in [0, T]$, with the initial and Neumann boundary conditions

$$u(r, 0) = u_0(r), \quad \frac{\partial u}{\partial r}(R, t) = 0. \quad (2)$$

We consider the diffusion equation in polar coordinates since both the whole boundary value problem and the bleaching pattern used in FRAP experiments have the rotational (radial) symmetry.¹

The main issue in FRAP and related inverse problems of parameter estimation is to find the value of the underlying model parameters (e.g., the diffusion coefficient D) from spatio-temporal measurements of the concentration $u(r, t)$, see [13, 14].

Obviously, the measured data are discrete and each data entry quantifies the variable u at a particular spatio-temporal point (r, t) in a finite domain, i.e.,

$$u(r_i, t_j) \in [0, u_{max}], \quad i = 0 \dots n, \quad j = 0 \dots m,$$

where i is the spatial index uniquely identifying the pixel position where the value of fluorescence intensity u is measured and j is the time index (the initial condition corresponds to $j = 0$). Usually, the data points are uniformly distributed both in time (the time interval Δt between two consecutive measurements is constant) and space, i.e., on an equidistant mesh with the step-size Δr , see [8]. Having $N_{\text{data}} := (m + 1) \times (n + 1)$ the total number of spatio-temporal data points, we can define a forward map (also called a parameter-to-data map)

$$\begin{aligned} F : \mathcal{R} &\rightarrow \mathcal{R}^{N_{\text{data}}} \\ (D) &\rightarrow u(r_k, t_k)_{k=1}^{N_{\text{data}}}. \end{aligned} \quad (3)$$

Our regression model is now

$$F(D) = u_e, \quad (4)$$

where the data $u_e \in \mathcal{R}^{N_{\text{data}}}$ are modeled as contaminated with additive white noise

$$u_e = F(D_T) + e = u(r_k, t_k)_{k=1}^{N_{\text{data}}} + (e_k)_{k=1}^{N_{\text{data}}}. \quad (5)$$

Here D_T denotes the true coefficient and $e \in \mathcal{R}^{N_{\text{data}}}$ is a data error vector, which we assume to be normally distributed with variance σ^2 for each time instant t_j , i.e., $e_j \in \mathcal{N}(0, \sigma^2)$, $j = 0, \dots, m$, $e_j \in \mathcal{R}^{n+1}$.

Given some data, the aim of the parameter estimation problem is to find D_T , such that Eq. (4) is satisfied in some appropriate sense. Since Eq. (4) usually consists of an overdetermined system (there are more data points than unknowns), it cannot be expected that it holds with equality, but instead an appropriate notion of a solution is that of a least-squares solution D_c (with $\| \cdot \|$ denoting the Euclidean norm on $\mathcal{R}^{N_{\text{data}}}$):

$$\| u_e - F(D_c) \|^2 = \min_{D > 0} \| u_e - F(D) \|^2. \quad (6)$$

¹ In our preceding papers [6–8, 14], we employed the Cartesian coordinate system.

Remark 1. The above defined parameter identification problem (6) is usually ill-posed (in a Hadamard sense) for non-constant coefficients [5] and a regularization technique has to be employed. Let us state, that the theory of regularization of ill-posed problems is well developed, see [4] and references within there. For some practical examples related to FRAP data processing see our works [6, 7, 14]. However, if we a-priori restrict the coefficients D to be constant, then the identification problem becomes well-posed.

Having the noisy data as in (5), the estimated value D_c of the true diffusion coefficient D_T can be computed numerically by solving the inverse problem to initial boundary value (IBV) problems (1)–(2). It can be shown [1, 3, 8], that for our case of single scalar parameter estimation and white noise as data error assumed, the expected relative error in D_T depends on the data noise level and a factor, which we call the global semi-relative squared sensitivity S_{GRS} , as follows

$$\mathbb{E} \left(\left| \frac{D_c - D_T}{D_T} \right|^2 \right) \sim \frac{\sigma^2}{S_{GRS}}, \quad (7)$$

where \mathbb{E} is the expected value and σ^2 denotes the variance of the additive Gaussian noise. The sensitivity measure S_{GRS} , that depends on the initial condition u_0 , is defined on a spatio-temporal mesh by

$$S_{GRS}(u_0) = D_T^2 \sum_{i=0}^n \sum_{j=1}^m \left[\frac{\partial}{\partial D} u(r_i, t_j) \Big|_{D=D_T} \right]^2, \quad (8)$$

where $\frac{\partial}{\partial D} u(r_i, t_j)$ is the usual sensitivity of the model output at the spatio-temporal point (r_i, t_j) with respect to the parameter D .

It is obvious from this estimate that if the noise level is fixed, the estimation of D_T can only be improved by switching to an experimental design with a higher sensitivity. The sensitivity measure (8) involves several design parameters. If the spatio-temporal grid for the data measurement is given, i.e., all the above parameters $R, T, \Delta r, \Delta t$ are fixed, there is only one way to maximize the sensitivity measure S_{GRS} : to consider the *initial condition* u_0 in (2) as the experimental design parameter. It means, for the discretized version of the IBV problems (1)–(2), the aim is to find the initial condition $(u_{00}, \dots, u_{0n})^T = (u_0(r_0), \dots, u_0(r_n))^T \in \mathcal{R}^{n+1}$ such that S_{GRS} is maximized and hence the expected error in D_T is minimized. In order to do so, we establish the bounds where the initial condition is considered: $\underline{u}_0 \leq u_{0i} \leq \overline{u}_0$, $i = 0, \dots, n$, where $\underline{u}_0, \overline{u}_0 \in \mathcal{R}$, $\underline{u}_0 < \overline{u}_0$. The optimization problem is formulated as follows

$$u_0^{opt} = \arg \max_{u_0 \in \mathcal{R}^{n+1}} S_{GRS}(u_0) \quad \text{subject to} \quad \underline{u}_0 \leq u_{0i} \leq \overline{u}_0, \quad i = 0, \dots, n. \quad (9)$$

Without loss of generality, we set $\underline{u}_0 = 0$ (zero components) and $\overline{u}_0 = 1$ (non-zero components).

Thus, the scaled sensitivity measure (8) has the form

$$S_{GRS} = \delta_T^2 \sum_{i=0}^n \sum_{j=1}^m \left[\frac{\partial}{\partial \delta} u(\tilde{r}_i, \tilde{t}_j) \Big|_{\delta=\delta_T} \right]^2 = \sum_{i=0}^n \sum_{j=1}^m \left[\tilde{t}_j \frac{\partial}{\partial \tilde{t}} u(\tilde{r}_i, \tilde{t}) \Big|_{\tilde{t}=\tilde{t}_j} \right]^2. \quad (14)$$

Replacing the derivative with a finite difference, and using the fact that $\tilde{t}_j = j\Delta\tilde{t}$, the sensitivity measure S_{GRS} can be approximated as follows

$$\begin{aligned} S_{GRS}(u_0) &\approx \sum_{i=0}^n \sum_{j=1}^m \left[j\Delta\tilde{t} \frac{u(\tilde{r}_i, \tilde{t}_j) - u(\tilde{r}_i, \tilde{t}_{j-1})}{\Delta\tilde{t}} \right]^2 \\ &= \sum_{j=1}^m j^2 \sum_{i=0}^n [u_{i,j} - u_{i,j-1}]^2 =: S_{app}(u_0(\tilde{r})), \end{aligned} \quad (15)$$

where the values $u_{i,j}$ are computed from $u_{i,j-1}$ using (13), thus no extra work is necessary.

As we already proved in [9], the components of the vector u_0^{opt} (in discrete points $\tilde{r}_0, \dots, \tilde{r}_n$) attain only two values \underline{u}_0 and \overline{u}_0 . The jumps between these values in fact represent the discontinuities in bleached domain leading to more complex bleaching patterns, see [9] for more details.

Remark 2. A variety of patterns arises as optimal for slow diffusive transport (for low values of δ), e.g., the bleaching pattern is called a *disc*, an *annulus*, a *disc & annulus*, or an *annulus & annulus* if

$$\begin{aligned} u_0^{opt} &= [\overline{u}_0, \dots, \overline{u}_0, \underline{u}_0, \dots, \underline{u}_0]^T, \\ u_0^{opt} &= [\underline{u}_0, \dots, \underline{u}_0, \overline{u}_0, \dots, \overline{u}_0, \underline{u}_0, \dots, \underline{u}_0]^T, \\ u_0^{opt} &= [\overline{u}_0, \dots, \overline{u}_0, \underline{u}_0, \dots, \underline{u}_0, \overline{u}_0, \dots, \overline{u}_0, \underline{u}_0, \dots, \underline{u}_0]^T, \\ u_0^{opt} &= [\underline{u}_0, \dots, \underline{u}_0, \overline{u}_0, \dots, \overline{u}_0, \underline{u}_0, \dots, \underline{u}_0, \overline{u}_0, \dots, \overline{u}_0, \underline{u}_0, \dots, \underline{u}_0]^T, \end{aligned}$$

respectively, see Fig. 1. Note that this formulation is well suited for both (i) the optimization of size of a specific bleached domain geometry (shape or pattern), and (ii) the optimization between all possible patterns (as explained above).

4 Bleaching Pattern Optimization for FRAP Experiments

Previously, in [15], we found that there exists an optimal size of the bleached domain when the pattern is restricted to the simply connected (disc) shape only. The quantitative aspect of our result announced in [15], i.e., $r_{opt} = 1.728\sqrt{D_c T}$, valid for an infinite domain, was confirmed once again in this study, see Fig. 2. Indeed, comparing two cases with $D = 0.1$ (the dashed curve) and $D = 0.001$ (the dotted curve), we see that the optimal disc radius with 100 times lower diffusive mobility is 10 times smaller. Here, we have numerically (for $T = 1$), $r_{opt} = 0.56$ and $r_{opt} = 0.06$, for $D=0.1$ and $D=0.001$, respectively; while the

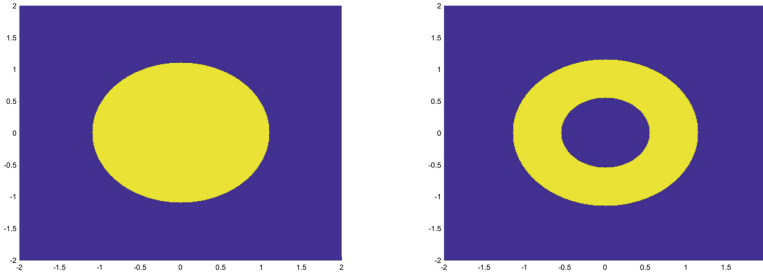


Fig. 1. Optimal pattern of initial bleach for two different values of $\delta = \frac{DT}{R^2}$, see [9]. Left: $\delta = 1/4$ (*disc*), right: $\delta = 1/12$ (*annulus*).

“analytical formula” according to [15] leads to values $r_{opt}(D=0.1) = 0.546$ and $r_{opt}(D=0.001) = 0.055$, respectively.

In our work [9], we found various optimal initial conditions for an inverse problem of diffusion constant estimation on an infinite two-dimensional domain. An interesting pattern of increasingly complicated (with respect to connectivity) optimal initial shapes was discovered by an efficient algorithm of numerical optimization. Figure 1 shows the result for two values of dimensionless diffusion constant δ . The exhaustive explanation of such an interesting result is far of scope of this paper. Nevertheless, the principal cause behind this phenomenon is clear: with the restricted observation domain and fixed overall measurement time interval, the more complex pattern provides a better exploitation of spatio-temporal data because the sensitivity measure S_{GRS} , see (8), gives a higher value than for any simpler shape, e.g., a disc (common bleaching shape used in FRAP community).

As follows, we present a numerical case study related to the FRAP experiment on a finite domain (with the Neumann homogeneous boundary condition) when the bleaching pattern (or shape) is not restricted previously. The goal is twofold: (i) to demonstrate (once again) the influence of S_{app} on a solution of inverse problem (6), and (ii) to show the unexpected variety of patterns corresponding to optimal initial conditions, see Fig. 1.

Further, in order to perform our virtual FRAP experiments (getting several sets of virtual experimental data), we used two values of $D = D_T$ and two levels of noise σ . The parameters defining the experimental protocol were fixed, more specifically:

$$R = 1, \quad T = 1, \quad n = 100, \quad m = 300.$$

The sequence of numerical computation is the following:

1. Choose a diffusion coefficient D_T and the non-zero components of $u_0 \in \mathcal{R}^{n+1}$.
2. Generate the time evolution $u_{i,j}$ computed using (13).
3. Produce the noisy data from $u_{i,j}$ using (5).
4. Solve problem (6) and find a solution D_c .

Figure 3 shows illustrative examples of exact and noisy data, which represents a time sequence of “row” data for further processing. The dashed lines are the

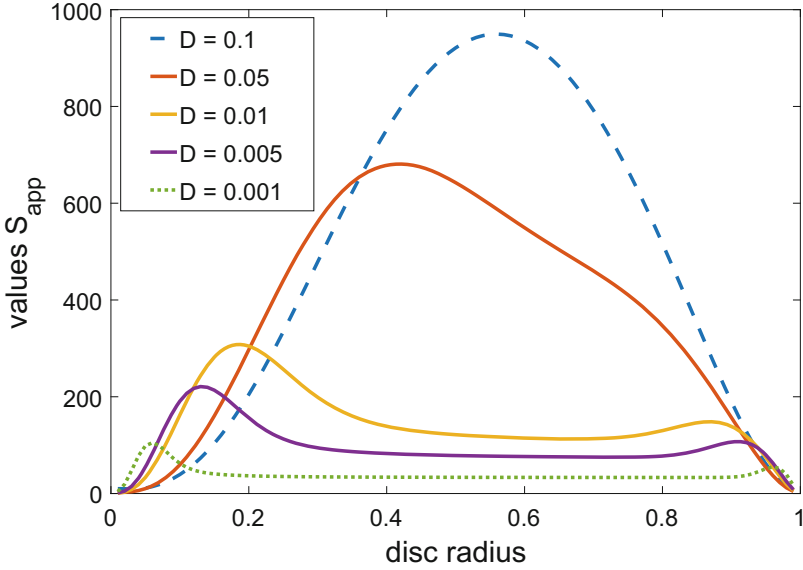


Fig. 2. The values S_{app} vs. disc radius r for 5 different diffusion coefficients D . The existence of optimal disc radii for each case is clearly visible.

initial conditions u_0 ($j = 0$) and the solid lines from top to bottom correspond to every 30-th time instant of exact data, i.e., for $j = 30, 60, \dots, 300$. The noisy data are plotted only for $j = 0, 150, 300$. The known values of D_T and σ were used only *a posteriori* for the evaluation of relative error \mathbb{E} and value $\frac{\sigma^2}{S_{app}}$, cf. (7), used in Figs. 4 and 5.

In order to generate our virtual “row” data, we used several following different initial conditions $u_0 \in \mathcal{R}^{101}$ with non-zero components u_{0i} , $i \in \{0, \dots, 100\}$, listed below, with corresponding values of S_{app} . For the first set of initial conditions IC1–IC5 the value $D_T = 0.1$ was used (with two levels of the noise $\sigma = 0.01$ and $\sigma = 0.1$), while for the second set of initial conditions IC6–IC9 we chose the slower diffusion $D_T = 0.01$ (and both levels of the noise) in order to allow a more complicated bleaching pattern as optimal.

The first set of initial conditions for $D_T = 0.1$:

- IC1: u_0 has non-zero components for $i = 0, \dots, 55 \Rightarrow S_{app} = 949.5$
(maximal value of S_{app} among all discs)
- IC2: u_0 has non-zero components for $i = 19, \dots, 51 \Rightarrow S_{app} = 589.2$
(the initial condition leading to maximal value of S_{app} among all annuli in case of using another value $D_T = 0.01$)
- IC3: u_0 has non-zero components for $i = 0, \dots, 79 \Rightarrow S_{app} = 516.7$
(suboptimal disc of a large radius)
- IC4: u_0 has non-zero components for $i = 0, \dots, 29 \Rightarrow S_{app} = 478.9$
(suboptimal disc of a small radius)

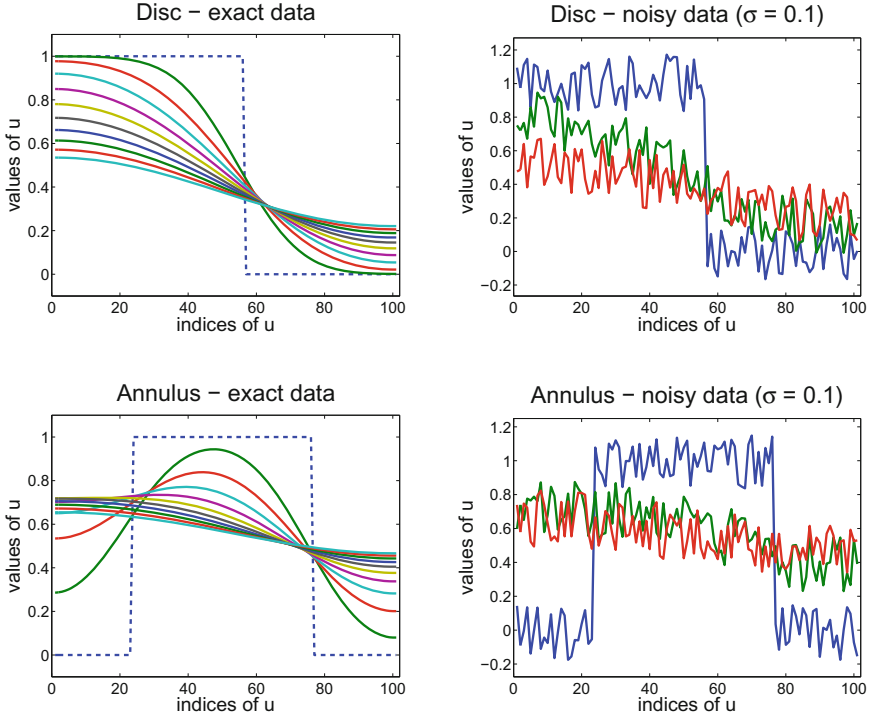


Fig. 3. Initial values of u_0 (the dashed stepwise curve) and the time evolution of the solution u_{ij} computed using (13) for $D_T = 0.1$. On top: disc – the non-zero components of u_0 have indices $i = 0, \dots, 55$. Beneath: annulus – the non-zero components of u_0 have indices $i = 23, \dots, 75$. Left: exact data. Right: corresponding noisy data with $\sigma = 0.1$.

- IC5: u_0 has non-zero components for $i = 44, \dots, 55 \Rightarrow S_{app} = 100.5$ (suboptimal annulus).

The second set of initial conditions for $D_T = 0.01$:

- IC6: u_0 has non-zero components for $i = 19, \dots, 51 \Rightarrow S_{app} = 458.5$ (maximal value of S_{app} among all annuli)
- IC7: u_0 has non-zero components for $i = 0, \dots, 18 \Rightarrow S_{app} = 307.7$ (maximal value of S_{app} among all discs)
- IC8: u_0 has non-zero components for $i = 42, \dots, 69 \Rightarrow S_{app} = 305.2$ (suboptimal annulus)
- IC9: u_0 has non-zero components for $i = 0, \dots, 57 \Rightarrow S_{app} = 115.7$ (suboptimal disc).

Remark 3. It could be objected that the optimal annulus is missing among the initial conditions IC1–IC5. This is true and the reason is simple. In case a disc is the optimal bleaching pattern, the optimal annulus ends at the domain boundary.

The proof relies on the evaluation of S_{GRS} (and obviously S_{app} , as well), which leads to the identical results, cf. (14). This kind of “impure” annuli will be excluded. Note that in case of a “pure annulus” as the optimal bleaching pattern, the “complement” is the combination of a disc and “impure” annulus. The reason why to exclude the bleaching domain from the border region as well as the detailed analysis of the phenomenon of spurious optima is left to the near future.

The sensitivity measure S_{app} , computed from exact data (without noise), was used for determining the values $\frac{\sigma^2}{S_{app}}$ which theoretically give the upper bound on the expected value \mathbb{E} , see (7). In Tables 1–2, we resume the theoretical values of expected relative errors of a diffusion constant estimate, see (7), for all initial conditions (IC1–IC9) and both levels of noise (18 quantities in total).

Table 1. Expected values $\frac{\sigma^2}{S_{app}}$ for initial conditions IC1–IC5 (with $D_T = 0.1$) defining the concentration $u_0(r_i)$. Two levels of noise are defined by σ .

IC	IC1	IC2	IC3	IC4	IC5
S_{app}	949.5	589.2	516.7	478.9	100.5
σ	0.01	0.01	0.01	0.01	0.01
σ^2/S_{app}	1.05E-7	1.70E-7	1.94E-7	2.09E-7	9.95E-7
σ	0.1	0.1	0.1	0.1	0.1
σ^2/S_{app}	1.05E-5	1.70E-5	1.94E-5	2.09E-5	9.95E-5

Table 2. Expected values $\frac{\sigma^2}{S_{app}}$ for initial conditions IC6–IC9 (with $D_T = 0.01$) defining the concentration $u_0(r_i)$. Two levels of noise are defined by σ .

IC	IC6	IC7	IC8	IC9
S_{app}	458.5	307.7	305.2	115.7
σ	0.01	0.01	0.01	0.01
σ^2/S_{app}	2.18E-7	3.25E-7	3.28E-7	8.64E-7
σ	0.1	0.1	0.1	0.1
σ^2/S_{app}	2.18E-5	3.25E-5	3.28E-5	8.64E-5

The 18 data sets defined by 9 different initial bleaching patterns IC1–IC9 and two noise levels were further processed and compared mutually. That is, two different true diffusion coefficients and two different levels of a Gaussian white noise ($\sigma = 0.1$ and $\sigma = 0.01$) were chosen in order to generate 1000 trajectories representing the concentration u (measured as profiles of fluorescent level). Using our method of diffusion parameter estimation [14] we obtained 1000 values of D_c for each data set. All these values were statistically processed. Figures 4 and 5 illustrate the results. They are ordered into 4 groups: (i) IC1–IC5 (for $D_T = 0.1$) and $\sigma = 0.01$, (ii) IC1–IC5 (for $D_T = 0.1$) and $\sigma = 0.1$,

(iii) IC6–IC9 (for $D_T = 0.01$) and $\sigma = 0.01$, and (iv) IC6–IC9 (for $D_T = 0.01$) and $\sigma = 0.1$.

The distribution of the squared error $|\frac{D_c - D_T}{D_T}|^2$ is shown in Fig. 4. As we expected according to Tables 1–2, the smallest relative error for $D_T = 0.1$ corresponds the IC1 (an optimal disc, $S_{app} = 949.5$), while for a slower diffusion (for $D = 0.01$) the optimal annulus (IC6, $S_{app} = 485.5$) gives smaller relative error than the optimal disc (IC7, $S_{app} = 307.7$).

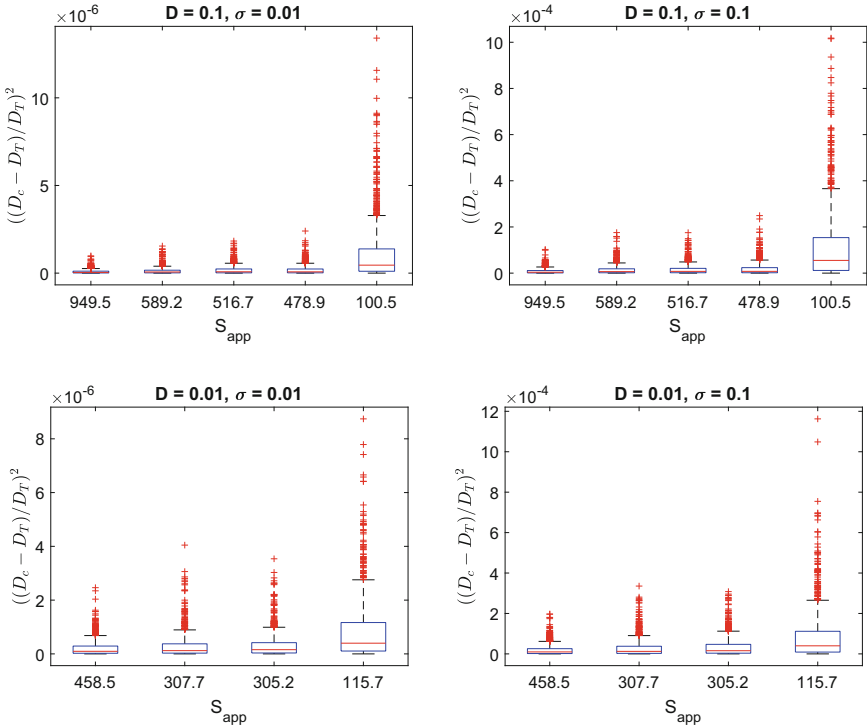


Fig. 4. The results of numerical estimation of diffusion coefficient. The boxplots show the distribution of the squared error $|\frac{D_c - D_T}{D_T}|^2$, each based on 1000 noisy signal samples. The cross marks (+) indicate the outliers. The higher value of S_{app} corresponds to the smaller box (and shorter confidence interval). Note that for both the upper left and right graphical results (for $D = 0.1$) the IC1 (an optimal disc) gives the smallest relative error, while for lower two cases of a slower diffusion (for $D = 0.01$), the optimal annulus (IC6) beats the optimal disc (IC7).

In order to graphically illustrate the difference between theoretically and numerically determined values of the relative error in a diffusion constant estimation, we plotted Fig. 5. It was reached using (7), i.e., the difference $\frac{\sigma^2}{S_{app}} - \mathbb{E} \left(\left| \frac{D_c - D_T}{D_T} \right|^2 \right)$ was calculated and consequently statistically processed.

Again, the respective boxplots show these distributions over data sets with different physical (D_T), technical (σ) and experimental (initial conditions IC1–IC9) attributes. The discrepancy is low for all cases, thus, we can rely on our sensitivity based approach to the optimum experiment design. Indeed, we see that the initial condition with maximal value of S_{app} exhibits the narrowest intervals and a distribution that is on average closest to zero than other initial conditions (which confirms the above statement about sensitivity S_{app} based approach).

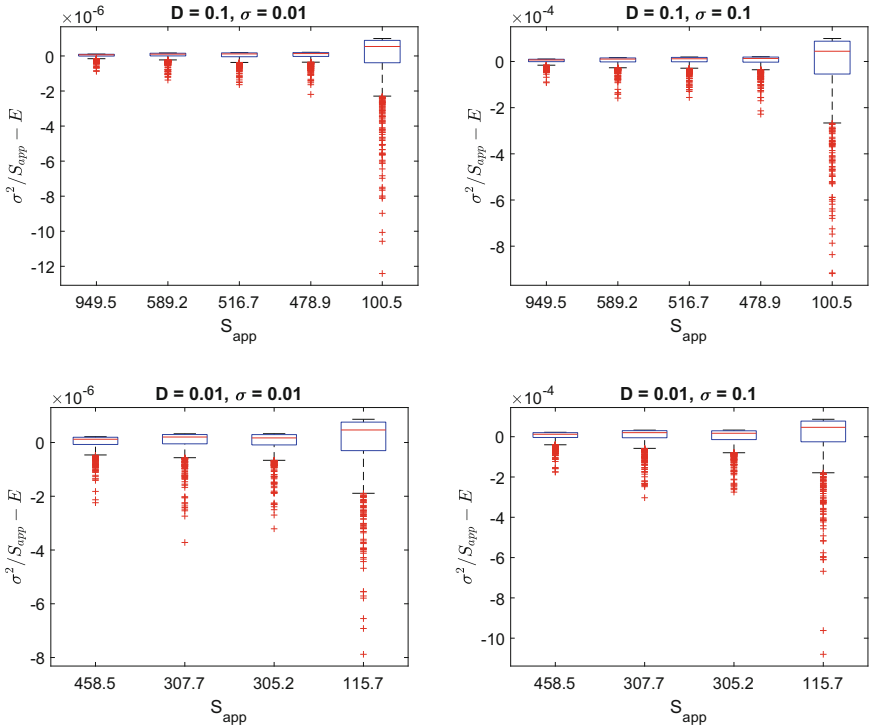


Fig. 5. The graphical results show the difference between the theoretically and numerically determined values of the relative error in a diffusion constant estimation.

Remark 4. Let us add some comments about the solution process of optimization problem (9) and finding the optimal initial condition u_0^{opt} . We used a global optimization method from the UFO system [10]. This method uses local optimization methods for finding local minima. Briefly speaking, we choose an initial $u_0^{(0)} = (1/2, \dots, 1/2)^T$ and for $k = 0, 1, \dots$, until the optimality conditions are satisfied, we update the next iterate $u_0^{(k+1)}$ from $u_0^{(k)}$ based on the function value $S_{app}(u_0^{(k)})$ and its gradient. For each D we obtained a solution on the boundary of the feasible region. Thus, $u_0^{opt}(r_i) \in \{1, 0\}$ is a binary-valued vector (there exist non-zero components of u_0^{opt}). As we already explained, a small number

of jumps between 1 and 0 in u_0^{opt} occurs for large values of a scaled diffusion coefficient δ . When δ decreases, the number of jumps increases.

5 Conclusion

In this paper, we have shown the importance of an interconnection between two important activities in performing model based experiments: (i) the setting of experimental factors, and (ii) data processing based on an underlying mathematical model containing the specific experimental conditions as parameters. Although our approach is illustrated on a specific case of photobleaching experiment only, it has general validity. The key concept is based on evaluation of sensitivity of measured data with respect to estimated parameters value. To do so, first we formulated the problem of parameter estimation in precise terms of parameter-to-data map, parameter estimates and their relative errors.

Only after that, our idea of the model-based optimization of experimental conditions was presented on a numerical case study. We set up the numerical procedure leading simultaneously to the optimal size and shape of a bleached domain for which the sensitivity measure reaches the maximal value, hence assuring the smallest relative error of the estimated parameter. Numerical calculations revealed rather surprising results. For high values of the dimensionless diffusion coefficient, the disc is the optimal shape and for smaller values, shapes with more and more components (i.e., annuli-type shapes) become optimal. In particular, this is not the disc which is the best shape, but, for practically relevant values of the experimental settings, sometimes an annulus can be better because it leads to a significant improvement in the confidence interval. Hence, it is proved that the bleach size and shape can be readily optimized and the bleaching pattern represents one of the most important experimental design factors in photobleaching experiments.

We hope that our findings will be incorporated into a novel generation of the FRAP experimental protocols – it is not computationally expensive and the enhancement of the parameter estimation process can be substantial.

Acknowledgement. This work was supported by the long-term strategic development financing of the Institute of Computer Science (RVO:67985807) and by the Ministry of Education, Youth and Sports of the Czech Republic - projects “CENAKVA” (No. CZ.1.05/2.1.00/01.0024), “CENAKVA II” (No. LO1205 under the NPU I program) and ‘The CENAKVA Centre Development’ (No. CZ.1.05/2.1.00/19.0380).

References

1. Bates, D.M., Watts, D.G.: *Nonlinear Regression Analysis: Its Applications*. Wiley, New York (1988)
2. Blumenthal, D., Goldstien, L., Edidin, M., Gheber, L.A.: Universal approach to FRAP analysis of arbitrary bleaching patterns. *Scientific reports* 5, Article no. 11655 (2015). <https://doi.org/10.1038/srep11655>

3. Cintrón-Arias, A., Banks, H.T., Capaldi, A., Lloyd, A.L.: A sensitivity matrix based methodology for inverse problem formulation. *J. Inv. Ill-Posed Probl.* **17**, 545–564 (2009)
4. Engl, H., Hanke, M., Neubauer, A.: *Regularization of Ill-Posed Problems*. Kluwer, Dordrecht (1996)
5. Hadamard, J.: *Lectures on the Cauchy Problem in Linear Partial Differential Equations*. Yale University Press, New Haven (1923)
6. Kaňa, R., Kotabová, E., Lukeš, M., Papáček, Š., Matonoha, C., Liu, L.N., Prášil, O., Mullineaux, C.W.: Phycobilisome mobility and its role in the regulation of light harvesting in red algae. *Plant Physiol.* **165**, 1618–1631 (2014)
7. Kaňa, R., Matonoha, C., Papáček, Š., Soukup, J.: On estimation of diffusion coefficient based on spatio-temporal FRAP images: an inverse ill-posed problem. In: Chleboun, J., Segeth, K., Šístek, J., Vejchodský, T. (eds.) *Programs and Algorithms of Numerical Mathematics 16*, pp. 100–111 (2013)
8. Kindermann, S., Papáček, Š.: On data space selection and data processing for parameter identification in a reaction-diffusion model based on FRAP experiments. *Abstr. Appl. Anal.* Article ID 859849 (2015)
9. Kindermann, S., Papáček, Š.: Optimization of the shape (and topology) of the initial conditions for diffusion parameter identification (2016). <https://arxiv.org/pdf/1602.03357.pdf>
10. Lukšan, L., Tůma, M., Matonoha, C., Vlček, J., Ramešová, N., Šiška, M., Hartman, J.: UFO 2014 - interactive system for universal functional optimization. Technical report V-1218. Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague (2014). <http://www.cs.cas.cz/luksan/ufo.html>
11. Mai, J., Trump, S., Ali, R., Schiltz, R.L., Hager, G., Hanke, T., Lehmann, I., Attinger, S.: Are assumptions about the model type necessary in reaction-diffusion modeling? A FRAP application. *Biophys. J.* **100**(5), 1178–1188 (2011)
12. Mueller, F., Mazza, D., Stasevich, T.J., McNally, J.G.: FRAP and kinetic modeling in the analysis of nuclear protein dynamics: what do we really know? *Curr. Opin. Cell Biol.* **22**, 1–9 (2010)
13. Mullineaux, C.W., Tobin, M.J., Jones, G.R.: Mobility of photosynthetic complexes in thylakoid membranes. *Nature* **390**, 421–424 (1997)
14. Papáček, Š., Kaňa, R., Matonoha, C.: Estimation of diffusivity of phycobilisomes on thylakoid membrane based on spatio-temporal FRAP images. *Math. Comput. Modell.* **57**, 1907–1912 (2013)
15. Papáček, Š., Kindermann, S.: On optimization of FRAP experiments: model-based sensitivity analysis approach. In: Ortuño, F., Rojas, I. (eds.) *IWBBIO 2016*. LNCS, vol. 9656, pp. 545–556. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31744-1_49
16. Sadegh Zadeh, K., Montas, H.J., Shirmohammadi, A.: Identification of biomolecule mass transport and binding rate parameters in living cells by inverse modeling. *Theor. Biol. Med. Modell.* **3**, 36 (2006)
17. Sbalzarini, I.F.: *Analysis, modeling and simulation of diffusion processes in cell biology*. VDM Verlag Dr. Muller (2009)



Modeling and Simulation of Microalgae Growth in a Couette-Taylor Bioreactor

Štěpán Papáček¹, Ctirad Matonoha²(✉), and Karel Petera³

¹ Institute of Complex Systems, South Bohemian Research Center of Aquaculture and Biodiversity of Hydrocenoses, Faculty of Fisheries and Protection of Waters, University of South Bohemia in České Budějovice, Zámek 136, 373 33 Nové Hradky, Czech Republic

`spapacek@frov.jcu.cz`

² Institute of Computer Science, The Czech Academy of Sciences, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic

`matonoha@cs.cas.cz`

³ Faculty of Mechanical Engineering, Czech Technical University in Prague, Technická 4, 166 07 Prague 6, Czech Republic

`Karel.Petera@fs.cvut.cz`

Abstract. Despite the fact that biotechnology with microalgae is attracting a lot of research interest since 1950s, a reliable computational tool for simulation of microalgal bioreactors is still lacking. In this work, a unified multidisciplinary modeling framework for microalgae culture systems is presented. Our framework consists of the model of microalgae growth in form of advection-diffusion-reaction system within a phenomenological model of photosynthesis and photoinhibition. The fluid dynamics is described by the Navier-Stokes equations and the irradiance field inside a reactor closes the equation system. The main achievement resides in successful integration of computational fluid dynamics code ANSYS Fluent and reaction kinetics, which makes our approach reliable and simple to implement. As a case study, the simulation of microalgae growth in a Couette-Taylor bioreactor is presented. The bioreactor operation leads to hydrodynamically induced fluctuating light conditions and the flashing light enhancement phenomenon, known from experiments. The presented model thus exhibits features of a real system.

Keywords: Microalgae · Mathematical modeling · Photosynthesis
CFD · Couette-Taylor bioreactor · Flashing light enhancement

1 Introduction

After the failure of first generation biofuels based on corn and soya, which created a food shortage in the third world, the focus of scientific community aimed on simple photosynthetic organisms, cyanobacteria and microalgae [7]. These incredibly versatile microorganisms are attracting a lot of research interest for over a half-century, mainly for high photosynthetic efficiency, biomass growth

and lipids content, see [19] and references within there. Microalgae are oxygenic unicellular phototrophs, which utilize the light energy to fix inorganic carbon (CO_2) to synthesize more complex organic molecules in photosynthetic reactions. Moreover, they mitigate carbon dioxide but also consume inorganic nitrogen and phosphorus and thus may participate in waste-water treatment processes [19].

However, there are technological and knowledge-based barriers preventing optimized mass cultivation of photosynthetic microorganisms. The reason resides in the fact that a reliable model of microalgae growth have to deal with complex problems, mainly with three-dimensional multiphase (gas-liquid-solid) flow dynamics, irradiance distribution and multi-level functionality of cellular processes. Moreover, all these parts interact across different timescales. Thus, one has to solve both theoretical (coping with multi-timescale phenomena) and practical (dealing with enormous computational requirements) issues. Although the correct integration of a CFD (computational fluid dynamics) code and photosynthetic reaction kinetics is essential for meaningful solution, most studies of general microalgae culture systems (MCS), e.g., photobioreactors (PBRs), are focused on partial problems without clear connection to the whole production process, cf. [3, 13] and references within. Resuming, reliable methods with a predictive power for *in silico* simulation of microbial growth in MCS are rather slowly emerging than being well established [4].

In this study, we present a unified modeling framework integrating the fluid dynamics and photosynthetic reaction kinetics (Sect. 2). As a virtual experimental system par excellence we took the Couette-Taylor bioreactor (CTBR), first reported in [8]. Results of numerical simulation of microalgae growth in CTBR are shown and discussed in Sect. 3. Finally, in Sect. 4, we draw some conclusions and future goals.

2 Model Description

Our framework for MCS (photobioreactor, open or raceway ponds) consists of

1. the state system – mass balance equations in form of advection-diffusion-reaction partial differential equations – PDEs (1)–(3),
2. the fluid flow equations, i.e., Navier-Stokes equations (4), and
3. the irradiance distribution inside CTBR (5).

All three parts of the model are interconnected, i.e., the mass balance equations for the state variables (species characteristics, nutrients, gases, etc.) have to be solved simultaneously with the fluid dynamics (momentum balances, continuity equations). Nevertheless, we assume that the stationary flow field inside MCS is not affected by mass transfer and reactions [2]. This assumption permits the separation of biological and environmental states and thus biological and environmental parts (models) can be solved with different numerical method and with different spatial-temporal discretization, dramatically reducing the computational demands. This separation can be total (for the stationary flow field in a continuous system) or stepwise, e.g., reflecting some sequence of quasi-steady states in a production system operated in batch mode.

2.1 State Model

The biological part of modeling framework is based on material balance equations for the state variables of interest describing the time dependent (non-stationary) transport and reaction among the species or compounds [2]:

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (vc_i) - \nabla \cdot (D_e \nabla c_i) = R(c_i) + S(c_i), t \in [t_0, T], i = 1, \dots, m, \quad (1)$$

where t_0 and T are initial and final time, respectively, $c_i = c_i(x, t)$ is a conservative quantity (concentration or cell density), $v(x, t)$ is the velocity flow field ruled by the fluid-dynamic model, cf. (4), and $x \in \Omega \subset R^3$ stands for a position vector in a coordinate system (e.g., Cartesian or cylindrical). The dispersion coefficient $D_e(x)$ is a tensor of second order, which corresponds to the diffusion coefficient in microstructure description. $D_e(x)$ is empirical parameter describing mixing in the system and is influenced by the molecular diffusion and velocity profile, i.e., it is not a material constant.

The reaction kinetics and time changes of the reacting species are described by the reaction term $R(c_i)$ and source terms $S(c_i)$, respectively. The source term, e.g., the external load of nutrients into MCS, is usually modeled as a corresponding boundary condition. However, in order to simplify the analytic study of the optimal solution existence, see [1], while respecting that the location of discharge of some material could be inside the domain Ω , we prefer the above form of (1). The initial condition and boundary condition (impermeability of the domain boundary, e.g., PBR walls) are following:

$$c_{i_0} = c_i(x, t_0), x \in \Omega \subset R^3, i = 1, \dots, m, \quad (2)$$

$$\nabla c_i(x, t) = 0, x \in \partial\Omega, t \in [t_0, T], i = 1, \dots, m. \quad (3)$$

2.2 Fluid-Dynamic Model

Microalgae cells are solid particles and the consumed CO_2 and evolved O_2 are gases, thus the system should be described as multiphase flow and transport. However, neglecting the gaseous phase, it is possible to consider the microalgae culture within a bioreactor as a suspension where microalgae represent the dispersed phase. Also, considering the cell density about 10 kg m^{-3} for the dry weight of biomass, i.e., 1% of mass content, and assuming that an average diameter of a spherical microalgae cell is about ten micrometers [19], we can classify our flow system as single-phase within the employed computational software ANSYS Fluent [10].

Mass density of the suspension is determined as $\rho = \rho_w (1 - k) + \rho_s k$, where ρ_w is the mass density of the medium and ρ_s the cell mass density, and k is the volume fraction. However, one can assume $\rho = \rho_w$ because of a uniform distribution of algal cells (no aggregations) and the fact that microalgae are floating in the medium. Furthermore, the inter-particle distances in our case of dilute suspension are sufficiently large to calculate flow field over each particle or cell [2], i.e., particles do not interfere with the flow field.

Thus, we model the incompressible liquid phase (suspension of water, nutrients and microalgae), hence the classical system of Navier-Stokes equations and the continuity equation is used as fluid-dynamic model:

$$\frac{\partial v}{\partial t} + (v \cdot \nabla)v = f - \frac{1}{\rho} \nabla p + \nu \nabla^2 v, \quad \nabla \cdot v = 0, \quad (4)$$

in $[t_0, T] \times \Omega$, with suitable boundary conditions on $[t_0, T] \times \partial\Omega$ and initial conditions in Ω , and where v, p, f, ρ and ν denote the fluid velocity, the pressure, the body forces, fluid density and kinematic viscosity, respectively.

2.3 Irradiance Field Within CTBR and a Suitable Reaction Model

The photosynthetic reactions depend on many variables such as irradiance, accessibility of both inorganic (Fe, P, etc) and organic nutrients (e.g., carbohydrates), concentration of gases, osmolality, and many others. In this work, we focus only on the key variable determining the reaction kinetics, the fluctuations of irradiance; the other variables are assumed to be externally controlled to not limit the cellular growth. Generally, both light attenuation and scattering should be taken into account in order to describe the irradiance distribution inside CTBR [5]. Here, for the sake of simplicity, only the attenuation is taken into account. For the special case of CTBR with inner and outer radii r_{in}, R , respectively, illuminated from outside by the incident irradiance I_0 , the following relations were derived for the irradiance in a spatial point (determined by a radial position r) and the average irradiance in whole system [5]

$$u(r) = 2u_0 \frac{R}{r} \frac{\cosh(\Lambda \frac{r}{R})}{\cosh(\Lambda) + \sinh(\Lambda)},$$

$$u_{av} = \frac{4u_0}{1 - (\frac{r_{in}}{R})^2} \frac{1/\Lambda}{\cosh(\Lambda) + \sinh(\Lambda)} \left[\sinh(\Lambda) - \sinh(\Lambda \frac{r_{in}}{R}) \right], \quad (5)$$

where $u = I/I_{opt}$, I_{opt} reaches the value of $250 \mu\text{E m}^{-2} \text{s}^{-1}$ [22], $u_0 = I_0/I_{opt}$ is the incident irradiance on the outer surface ($r = R$), and Λ is the dimensionless attenuation coefficient. It is further convenient to define this quantity as follows: $\Lambda := \frac{\ln(2)R}{r_{1/2}}$, where $r_{1/2}$ is the length interval in which the intensity of light diminishes to one half (in a linear case).

Remark 1. It seems that optimal conditions for microalgae culture growth occurs when the averaged irradiance within CTBR has the value $u_{av} = 1$. However, this optimization problem depends on mixing intensity as well, and it is discussed elsewhere [6]. Only for the lumped parameter system, the value $u = 1$ is the optimal one (by definition).

Based on the decades of experimental research of photosynthesis, the reliable model of reaction kinetics in MCS has to cover at least three time scales across photosynthetic reactions. The following phenomena: (i) activation of the light

harvesting complex (photosynthetic unit – PSU) in light reactions, (ii) biomass production, and (iii) photoinhibition, i.e., damage of a part of PSU by an excessive irradiance, has to be reflected in the reaction term $R(c_i)$ in (1). As a suitable candidate for reaction model, we adopted the three-state model of photosynthetic factory (PSF) [9, 14, 16, 22, 23]. PSF model considers 3 states in which microalgae cells may exist: *activated* – A , *inhibited* – B , *rested* – R . Although the fluid-dynamical properties of cells in each of three states are identical, these states are impacted differently by dynamic changes of the environment, i.e., spatial-temporal changes of states concentrations determine the biomass production, cf. (6). Let be the concentrations of respective components c_A , c_B , and c_R (with the same units as for the microalgae cell density c_x in whole PBR – generally 10^6 cell ml^{-1} as in [22]). Then the following relation holds (for $\forall t \in [t_0, t_\infty]$, and $\forall x \in \partial\Omega$): $c_A(x, t) + c_B(x, t) + c_R(x, t) = c_x(x, t)$. The dimensionless scalar values $y_A = c_A/c_x$, $y_B = c_B/c_x$ and $y_R = c_R/c_x$ (molar fractions) are respective states of the PSF model; for more details see Sect. 2.4. It is important to emphasize here, that according to [9, 22], the spatial-temporal averaged rate of photosynthetic production (specific growth rate \dot{c}_x/c_x) is proportional to the activated state fraction as follows:

$$\dot{c}_x = \frac{\kappa\gamma}{\text{meas}(\Omega)T} \int_0^T \int_\Omega [y_A(I(x), t) - Me] c_x \, dx \, dt, \quad (6)$$

where term Me represents a cellular maintenance factor describing internal metabolism; Me is considered constant in our case although it may vary if hydrodynamical shear stress is considered [23].

Remark 2. Observing the term $\kappa\gamma$ in (6), which has a value approximately 10^{-4} [s^{-1}], we see the reason why the transition from the time-scale of light fluctuation to time-scale of biomass growth (macro-scale) is reached without loss of accuracy. State y_A of the PSF model, in the range of $[0, 1]$ as well as the other PSF model states, is sensitive to the light fluctuations and the integral in (6) can be evaluated separately. Afterwards, “the scale jump”, factor $\kappa\gamma$ in (6), provides the value of a real specific growth rate.

2.4 Model of Photosynthetic Factory – PSF Model

Model of photosynthetic factory (PSF model) proposed by Eilers and Peeters [9] and further developed by Wu and Merchuk [22, 23] and Papáček and Celikovský [14, 18] is used for the reaction term $R(c_i)$ derivation in the transport equation (1). The state vector of the PSF model is three dimensional, $y = (y_R, y_A, y_B)^\top$, where the respective components represents the probability that PSF is in the resting, activated and inhibited state. It is supposed that the photosynthetic reactions depend on the irradiance level only. This is the input or forcing function $u(t)$ in (7), which represents the time-dependent irradiance (in relative-dimensionless unit). The values of PSF model parameters in its original form, i.e., $\alpha, \beta, \gamma, \delta, \kappa$, are taken from [22], where Wu and Merchuk [22] identified these values for the microalga *Porphyridium* sp.: $\alpha = 1.935 \times 10^{-3} \mu\text{E}^{-1} \text{m}^2$,

$\beta = 5.785 \times 10^{-7} \mu\text{E}^{-1} \text{m}^2$, $\gamma = 1.460 \times 10^{-1} \text{s}^{-1}$, $\delta = 4.796 \times 10^{-4} \text{s}^{-1}$, $\kappa = 3.647 \times 10^{-3}$ and $Me = 5.9 \times 10^{-2} \text{h}^{-1}$. For details regarding the experimental design for parameter estimation and the identifiability study, see [18]. Aiming to use the singular perturbation method, the following re-parametrization was introduced in [18]: $q_1 := \sqrt{\frac{\gamma\delta}{\alpha\beta}}$, $q_2 := \sqrt{\frac{\alpha\beta\gamma}{\delta(\alpha+\beta)^2}}$, $q_3 := \kappa\gamma\sqrt{\frac{\alpha\delta}{\beta\gamma}}$, $q_4 := \alpha q_1$, $q_5 := \beta/\alpha$. Consequently, the PSF model acquires the following form:

$$\dot{y} = [\mathcal{A} + u(t)\mathcal{B}]y, \quad (7)$$

$$\mathcal{A} = q_4 \begin{bmatrix} 0 & q_2(1+q_5) & \frac{q_5}{q_2(1+q_5)} \\ 0 & -q_2(1+q_5) & 0 \\ 0 & 0 & -\frac{q_5}{q_2(1+q_5)} \end{bmatrix}, \quad \mathcal{B} = q_4 \begin{bmatrix} -1 & 0 & 0 \\ 1 & -q_5 & 0 \\ 0 & q_5 & 0 \end{bmatrix}. \quad (8)$$

Two negative eigenvalues result for the irradiance level of $250 \mu\text{E m}^{-2} \text{s}^{-1}$ (the value of the parameter q_1 which maximizes the steady-state growth): $\lambda_1 = -0.63$, $\lambda_2 = -0.59 \cdot 10^{-3}$, classifying the ODE system (7) as *stiff* (due to value 10^3 of the ratio $\frac{\lambda_1}{\lambda_2}$). This fact points to the existence of two processes widely separated in point of view of their characteristic times, being (i) photosynthetic light and dark reactions, and (ii) photoinhibition. The “slow” state can be under some conditions (constant level of the mean input) “frozen” and the so-called fast reduction used [16], i.e., the behavior of the system (7) is characterized by the only one following ODE

$$\frac{d}{dt}y_A = -q_4(u_{av} + q_2) \left[\frac{u(r) + q_2}{u_{av} + q_2} y_A - \frac{u(r)}{u_{av}} y_{A_{ss}}(u_{av}) \right], \quad (9)$$

and the “slow” variable y_B can be regarded as a constant depending on the averaged value $u = u_{av}$. According to our works [6, 16] it holds

$$y_{A_{ss}}(u) = \frac{u/q_2}{u^2 + u/q_2 + 1}, y_{B_{ss}}(u) = \frac{u^2}{u^2 + u/q_2 + 1}. \quad (10)$$

These equations representing the PSF model, either (7) and (9), have been implemented as a User-Defined Function (UDF) in ANSYS Fluent, see Sect. 3.1.

Remark 3. PSF model clearly satisfies the requirement for model candidate; its 5 parameters can be seen in re-parametrized form (7) as three time constants (inverse of reaction rates): the first time scale corresponding to the light and dark reactions is in order of seconds ($1/q_4$), the second one, corresponding to the photoinhibition ($1/q_5$), is in order of minutes, and finally, the third one corresponds to microalgae growth in order of hours ($1/q_3$), one parameter means the optimal irradiance (q_1), and the last parameter represents the shape of the steady state growth curve (q_2), see [16, 18] for more details.

3 Case Study: The Couette-Taylor Bioreactor

Since 1950s, in frame of the biotechnology with microalgae, the microalgae photosynthesis in flashing light conditions is attracting a lot of research interest

[11]. In our early works, where the PSF model was applied in a distributed parameter system [15, 16], we theoretically confirmed the so-called phenomenon of flashing light enhancement known from experiments [11, 12, 21]. However, our models suffered from over-simplification because Navier-Stokes equations were not solved there within a fluid dynamic model. Here, we introduce a general approach applicable to any geometry and operating conditions of MCS, thus ready for industrial applications. For the validation of our integrated model, we have chosen a Couette-Taylor bioreactor (CTBR); ideally when the so-called Taylor vortex flow regime takes place; Taylor number $Ta = \frac{\Omega^2 r_{in} (R - r_{in})^3}{\nu^2}$ exceeds the first critical value [20], where Ω is the angular velocity of the inner cylinder of CTBR, ν is the kinematic viscosity, and the inner and outer cylinder radii are r_{in} , and R , respectively [20]. This situation is illustrated by ANSYS Fluent simulation, see Fig. 1, left part, when the axial velocity components in axial cross section of CTBR are shown. The so-called Taylor vortex flow (in laminar flow regime) is forcing microalgae cells to periodically travel between illuminated wall and the dark side of the bioreactor. This hydrodynamically induced flashing light regime causes the light averaging and consequently the flashing light enhancement may occurs [14]. A similar results, i.e., the vortex flow in a laminar flow regime, can be observed within the idealized 2-D square cavity with moving upper wall, see Fig. 1, right part [17].

For the initial condition and other parameters from Table 1, the case study was resolved using UDF included in CFD code ANSYS Fluent. Within this software, PDEs (1)–(4) were embedded and the reaction kinetics was implemented as special UDF according to the description in Sect. 2.4. While various mixing intensities (controlled by Ω) were tested, only one irradiance level set by u_0 was used. This specific value of $u_0 = 5.469$ represents such value of incident irradiance (irradiance falling to the outer CTBR cylinder), which assures that the average irradiance in the culture has optimal value $u_{av} = 1$.

Remark 4. Having $R = 66.6$ [mm], $r_{in} = 50$ [mm] in the relation (5), the outer and inner radii, respectively, then setting $\Lambda = 36 \ln(2)$, means that $r_{1/2} = 66.6/36$ [mm], and the evaluation of average irradiance u_{av} gives as result $u_{av} = \frac{u_0}{5.469}$. Thus, in order to have $u_{av} = 1$, the incident irradiance $u_0 = 5.469$, see Table 1.

Table 1. Parameters needed for the simulation of microalgae growth in the Couette-Taylor bioreactor: u_0 is incident irradiance, r stands for CTBR radial coordinate interval, $r \in [r_{in}, R]$, Re is the Reynolds number, Λ stands for attenuation coefficient, q_2 , q_4 , q_5 , $y_R(t_0)$ and $y_A(t_0)$, represent 3 model parameters and 2 initial conditions (for rested and activated state) of the PSF model, respectively (see Sect. 2.4).

u_0	r [mm]	$Re = \frac{\Omega r_{in} (R - r_{in})}{\nu}$	Λ	q_2	q_4 [s^{-1}]	q_5	$y_R(t_0)$	$y_A(t_0)$
5.469	[50, 66.6]	0–100000	$36 \ln(2)$	0.3	0.5	0.0003	1	0

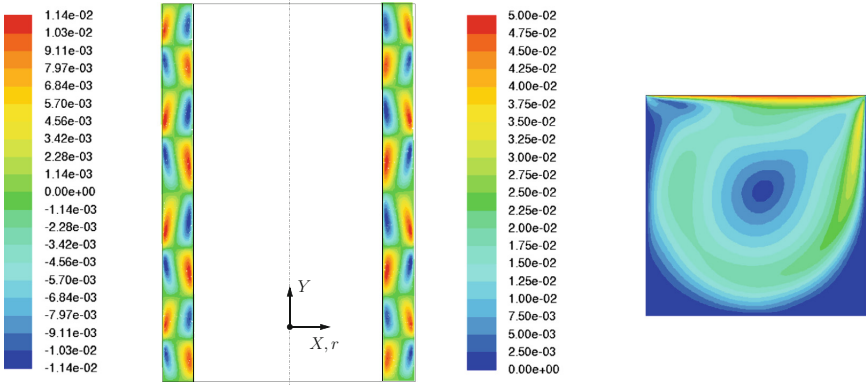


Fig. 1. (left) Contours of Y-velocity component (reaching both positive and negative values) in the cylindrical gap of the Coutte-Taylor bioreactor (Reynolds number is $Re = 1000$, and $\Omega = 1.2 \text{ rad s}^{-1}$). (right) Contours of steady-state velocity magnitude of the microalgae suspension within the idealized 2-D square cavity. The vortex flow is imposed by the moving upper wall ($Re = 1000$, i.e., laminar flow regime).

3.1 Numerical Aspects of PDEs (1)–(4) Implementation and Solution

The implementation of our model consisting of PDEs (1)–(4) is straightforward. The key role plays the reaction term $R(c_i)$ in (1); this term is “local” in sense that it depends on the actual local transport and reaction rates, and it contains all relevant time-scales, thus, it perfectly fits to our requirements. The technical simplicity of PDE based approach, which takes advantage of the sophisticated CFD codes, makes whole approach reliable and robust. There are well established methods and CFD software packages to solve the PDEs (1)–(4) defining our modeling framework. Hence, using a suitable numerical method, PDEs are transformed into a system of linear algebraic equations. In the fluid flow problems, the finite volume approach (FVM) is more popular in obtaining the discretized form of equations than other approaches like FEM or FDM (finite element or finite differences method, respectively), especially with unstructured grids. The system of discretized equations is solved, usually iteratively, to find the values of velocities and other scalar quantities like concentrations in all grid points.

The key issue is in the integration of a CFD code and photosynthetic reaction kinetics in one modeling framework. Our approach is based on the implementation of a UDF within the commercial CFD code ANSYS Fluent, which provides the possibility to define an arbitrary reaction term. ODE system (7) can be reduced to only one differential Eq.(9). The right-hand side of this equation represents the rate of change of activated state y_A which can be used in the definition of UDF. Macro `DEFINE_VR_RATE` provides the possibility to define an arbitrary reaction term in (1), including its dependency on the spatial coordinate as represented by (5). Macro `C_CENTROID` can retrieve corresponding coordinates

of the current mesh cell. The three states of the microalgae culture were represented as individual species with same molar weights. Therefore, mass and molar fractions should be identical here. Definition of the user-defined reaction rate employed in our simulations for the unreduced system (7) transformed to only two equations as the third state, y_R , is complement to 1:

```

DEFINE_VR_RATE (PSFrate, c,t,r,Mw,omega,rate,rr_t)
{
  real pos[ND_ND], rd,u,x,y,z,ya,yb,dya,dyb,ctot;
  C_CENTROID(pos,c,t);
  x = pos[0]; z = pos[2]; /* x, z-coordinate */
  y = pos[1]; /* y-coordinate, rotational axis */
  rd = sqrt( SQR(x) + SQR(z) ); /* radius coordinate */
  u = 2*U0*R/rd*cosh(LAMBDA/R*rd)/(cosh(LAMBDA)+sinh(LAMBDA));
  ctot = C_R(c,t)/Mw[ALG]; /* total molar concentration */
  ya = omega[ACTIVE]; /* active state */
  yb = omega[INHIBITED]; /* inhibited */
  *rate = -dya*ctot; *rr_t = *rate;
  if (!strcmp(r->name,"reaction-1")) {
    dya = -Q4*(1+Q5)*(Q2+u)*ya + Q4*u*(1-yb);
    *rate = -dya*ctot;
  }
  else if (!strcmp(r->name,"reaction-2")) {
    dyb = Q4*Q5*u*ya - Q4*Q5/(Q2*(1+Q5))*yb;
    *rate = -dyb*ctot;
  }
  *rr_t = *rate;
}

```

A mesh with 200 thousand hexahedral elements was used in our simulations (see Fig. 2 on the left). Even though a larger mesh size would be preferable we stayed with this relatively small mesh because our main restriction lies in the number of necessary time steps giving a steady-state solution. We used time step 0.1 s, and according to the approach used for estimation of the Grid Convergence

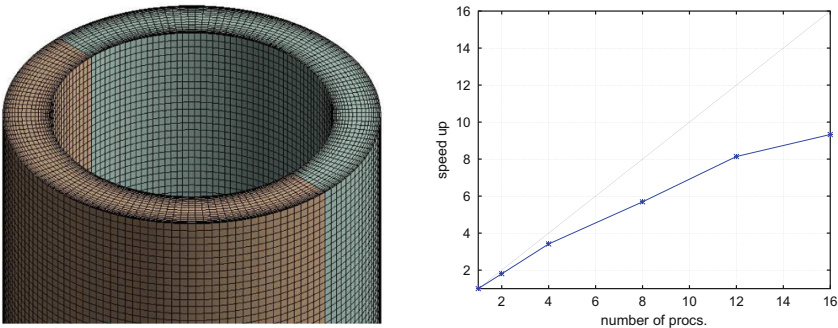


Fig. 2. (left) Illustration of the mesh used in our simulations. (right) Parallel speed-up of our simulations.

Index [24], we estimated the accuracy for this time step as 0.85%. It was based on the spatial average of the activated state concentration at the end of time range 0–5 s. The simulation of the whole time range 0–20000 s necessary for non-reduced system (7) took approximately 24 h using 8 parallel processes on our computational server (with Xeon CPU E5-4627, 2.60 GHz). It corresponds to real parallel speed-up around 6, see Fig. 2(right) describing the parallel speed-up of our simulations (based on 250 s time range and corresponding number of parallel processes: 1, 2, 4, 8, 12 and 16) and its deviation from the ideal linear dependency. Computational time for the reduced order system (9) and time range 0–40 s was around 23 min on the same computer, that is approximately 60 times smaller than for the non-reduced system. The problem of different accuracy of both PSF model formulations is treated theoretically in [6] and practically in the following Sect. 4.

Concerning our CFD simulations: we used non-slip boundary conditions at the cylindrical walls of the Couette-Taylor device, with a prescribed rotational rate for the outer wall. Symmetry-type boundary conditions were applied at the top and bottom of the geometry. Height of the cylindrical geometry was 200 mm, inner and outer diameters were 100 mm and 133.33 mm, that is their ratio is 0.75. Every simulation consisted of two steps. In the first step, the Navier-Stokes equations (4) were solved iteratively to get a converged solution of the steady-state flow field for the corresponding mixing rate (Reynolds number). In the second step, the flow field was assumed to be fully developed and only Eq. (1) describing the transport and reaction of individual species (states of the algae in our case) were solved. This approach is more efficient but in cases when the flow field exhibits a transient behavior, the Navier-Stokes equations should be solved simultaneously with the species transport equations. For example, the Couette-Taylor device shows an inherently transient behavior at some specific rotations rates [20].

4 Results and Discussion

Our specific case study, i.e., the microalgae growth simulation in CTBR, serves as a proof of concept of a CFD code integration with a microbial kinetics model. Both the time dependent and steady-state quantities were calculated. First, the dynamics of spatially averaged activated states y_A and y_B has been calculated using both reduced order and non-reduced systems, (9) and (7), respectively. While the reduced order system, *via* the so-called fast reduction, supposes y_B is frozen ($y_B = y_{B_{ss}}(u_{av})$), the non-reduced (full) system can simulate both fast and slow dynamics. Thus, in the case of applied fast reduction (left graph in Fig. 3), fast transitions from the initial state $y_A = 0$ to its steady state takes place for all mixing intensities (described by $Re = 0, 1000, 100000$) in few seconds. For the non-reduced PSF model, two phases can be observed: (i) fast phase - transition from the initial value $y_A(t_0) = 0$ to the maximal value occurring in seconds, and (ii) slow phase - gradually growing inhibited state y_B occurring in minutes (right graph in Fig. 3). Due to the slow dynamics, the value of y_A

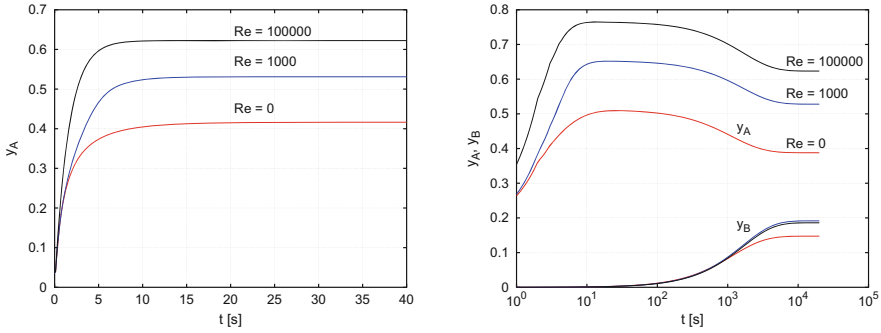


Fig. 3. State y_A (spatially averaged) vs. time t , for three different mixing intensities, $Re = 0, 1000, 100000$, with and without fast reduction, respectively. On the left, y_A vs. t according to reduced system (9), for time range 0–40 s. On the right, the time dependency of activated state y_A and inhibited state y_B (both spatially averaged) of the non-reduced system (7) is shown for time range 0–20000 s.

diminishes in far time horizon (which does not occur in left graph due to a constant level of y_B). The comparison of accuracy of respective results is discussed further, cf. comments on Fig. 4(right).

In biotechnological applications, a performance index J , which either classifies the quality of bioreactor design or the optimality of operating conditions, is usually defined. Such performance index depends on certain parameters, which can be optimized afterwards. The role of mixing on spatial distribution of the activated state y_A is shown in Fig. 4(left).

In order to quantify the impact of mixing (or more precisely the dependency of CTBR angular velocity Ω on the cellular growth), we define the objective function J_{CTBR} as the volumetric productivity:

$$J_{CTBR} = \frac{1}{\text{meas}(\Omega) T} \int_0^T \int_{\Omega} [y_A(u, r, t)] c_x \, dr \, dt. \quad (11)$$

In case of steady state operation in continuous cultivation mode (c_x is constant), it is only the activated state fraction $y_A(u, r, t)$, which is changing within CTBR volume, i.e., J_{CTBR} according to (11) can be simplified (it is only the normalized space-averaged integral, which is further evaluated): $J = \frac{J_{CTBR}}{c_x} = \frac{1}{\pi(R^2 - r_{in}^2)} \int_{\Omega} y_A(u, z, t) \, dz$.

In Fig. 4(right), we show the dependence of the performance index J on the mixing rate. There are 4 different cases within Fig. 4(right). For two geometries (square cavity and CTBR) are used both the reduced (one ODE) and non-reduced (two ODEs) system. In all cases, for lower Da (larger mixing rate - bigger Re), we get better performance J approaching its maximal theoretical value corresponding to the growth in averaged continuous light, cf. (10). These results confirm the experimentally measured flashing light enhancement phenomenon [12]. In case of poor mixing (low Re), the discrepancy between reduced order and

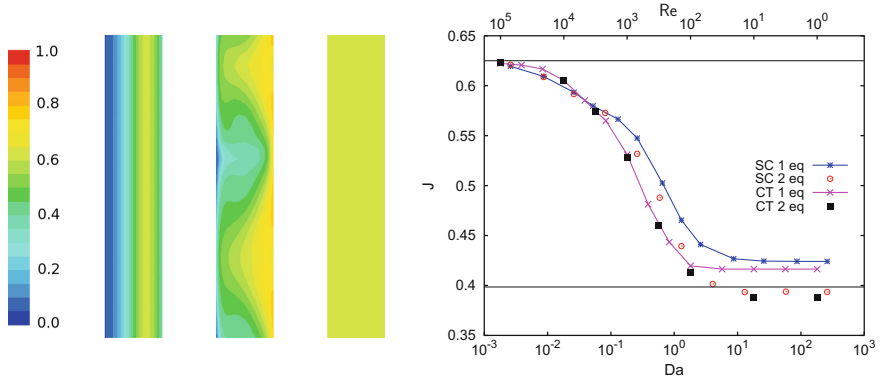


Fig. 4. (left) Contour plots of molar fraction y_A for different mixing rates - Reynolds numbers from left to right: 0, 1000, 100000. (right) Performance index J vs. Da (Re), gained for time range 0–20000 s, with time step 0.1 s. Red circles and black squares represent simulation results of the whole non-reduced (2 ODE) system (7) for the square cavity and CTBR, respectively. Asterisk and cross marks represent simulation results of the reduced (1 ODE) system (9) for the square cavity and CTBR, respectively. (Color figure online)

non-reduced models is visible. The reason resides in the violation of “sufficient mixing condition” permitting the averaging of the “slow” state y_B .

We note that our simulated data have an illustrative and testing purpose only and for real quantitative simulations one has to employ parameters for specific MCS and particular microalgae species. However, the result of biological performance for our CTBR clearly shows the growth enhancing role of mixing induced by convective motion, i.e., the simulated flashing light enhancement is comparable to experiments [12, 17]. The simulated results in Fig. 4(right) imply that for lower Da (bigger Re , better mixing) leads to better performance J (reaching actually its maximal theoretical value 0.625). However, the harmful influence of the hydrodynamically induced shear stress originally considered in Me term [23], see (6), is not taken into account and thus in reality certain velocity of mixing would start damaging the cellular integrity.

Remark 5. Analysis of our numerical model indicates that the numerical process is controlled by the *Damköhler number* $Da := q_r/q_{tr}$, which submits into relation the reaction rate q_r and mass transport rate (q_{tr}). For the reaction rate we have from (9): $q_r = [q_4(u_{av} + q_2)]$. Convective mass transport rate in CTBR is expressed as $q_{tr} = \frac{\Omega r_{in}}{R - r_{in}}$. Hence, for q_r fixed and employing the Re number for the Couette-Taylor flow in form of $Re = \frac{\Omega r_{in}(R - r_{in})}{\nu}$, it holds:

$$Da := q_r \frac{(R - r_{in})^2}{\nu} \frac{1}{Re}. \quad (12)$$

5 Conclusion

The crucial point for commercial success of a production plant, e.g., for algae biofuels, is the availability of a reliable software tool with the predictive capacity enabling simulation and optimization of system performance. In this work, we presented the general unified modeling framework for microalgae culture systems, which is independent of the actual production system geometry or microbial strain. All parts of the multidisciplinary framework, i.e., the state system, fluid-dynamic model, model of irradiance distribution, are interconnected within one CFD code ANSYS Fluent.

In the illustrative case study on a Couette-Taylor bioreactor, we have shown that a three-state model of photosynthetic factory well behaves under hydrodynamically induced high frequency light-dark cycles regime and copes with the requirement imposed on the reaction model, i.e., it correctly describes both the quasi steady-state and dynamic phenomena.

We conclude that our model provides an accurate description of microalgae growth in a CTBR, thus it is well suited for the optimal control problem formulation as well. Ongoing modeling efforts will continue in order to show how an optimization problem can be formulated and solved, and how to develop even more reliable model considering the effect of hydrodynamical shear stress on microalgae growth.

Acknowledgment. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic - projects “CENAKVA” (No. CZ.1.05/2.1.00/01.0024), “CENAKVA II” (No. LO1205 under the NPU I program) and The CENAKVA Centre Development (No. CZ.1.05/2.1.00/19.0380) and by the long-term strategic development financing of the Institute of Computer Science (RVO: 67985807).

References

1. Alvarez-Vázquez, L., Fernández, F.: Optimal control of bioreactor. *Appl. Math. Comput.* **216**, 2559–2575 (2010)
2. Beek, W.J., Muttzall, K.M.K., van Heuven, J.W.: *Transport Phenomena*. Wiley, Hoboken (2000)
3. Bernard, O., Mairet, F., Chachuat, B.: Modelling of microalgae culture systems with applications to control and optimization. *Adv. Biochem. Eng. Biotechnol.* **153**, 59–87 (2016)
4. Bernardi, A., Perin, G., Sforza, E., Galvanin, F., Morosinotto, T., Bezzo, F.: An identifiable state model to describe light intensity influence on microalgae growth. *Ind. Eng. Chem. Res.* **53**, 6738–6749 (2014)
5. Cornet, J.-F., Dussap, C.G., Gros, J.-B., Binois, C., Lasseur, C.: A simplified monodimensional approach for modeling coupling between radiant light transfer and growth kinetics in photobioreactors. *Chem. Eng. Sci.* **50**, 1489–1500 (1995)
6. Čelikovský, S., Papáček, Š., Cervantes-Herrera, A., Ruiz-León, J.: Singular perturbation based solution to optimal microalgal growth problem and its infinite time horizon analysis. *IEEE Trans. Autom. Control* **55**, 767–772 (2010)
7. Chisti, Y.: Biodiesel from microalgae. *Biotechnol. Adv.* **25**, 294–306 (2007)

8. Davis, E.A.: Turbulence. In: Burlew, J.S. (ed.) *Algal Culture from Laboratory to Pilot Plant*, vol. 600, pp. 135–138. Carnegie Institute, Washington, D.C. (1953)
9. Eilers, P.H.C., Peeters, J.C.H.: A model for the relationship between light intensity and the rate of photosynthesis in phytoplankton. *Ecol. Model.* **42**, 199–215 (1988)
10. ANSYS Fluent product documentation. <http://www.ansys.com/>
11. Kok, B.: Experiments on photosynthesis by *Chlorella* in flashing light. In: Burlew, J.S. (ed.) *Algal Culture From Laboratory to Pilot Plant*. Publ. no. 600, pp. 63–75. The Carnegie Institute, Washington, D.C. (1953)
12. Nedbal, L., Tichý, V., Xiong, F., Grobbelaar, J.U.: Microscopic green algae and cyanobacteria in high-frequency intermittent light. *J. Appl. Phycol.* **8**, 325–333 (1996)
13. Ooms, M.D., Dinh, C.T., Sargent, E.H., Sinton, D.: Photon management for augmented photosynthesis. *Nat. Commun.* **7**, 12699 (2016). <https://doi.org/10.1038/ncomms12699>
14. Papáček, Š., Čelikovský, S., Štys, D., Ruiz-León, J.: Bilinear system as modelling framework for analysis of microalgal growth. *Kybernetika* **43**, 1–20 (2007)
15. Papáček, Š., Štumbauer, V., Štys, D., Petera, K., Matonoha, C.: Growth impact of hydrodynamic dispersion in a Couette-Taylor bioreactor. *Math. Comput. Model.* **54**(7–8), 1791–1795 (2011)
16. Papáček, Š., Matonoha, C., Štumbauer, V., Štys, D.: Modelling and simulation of photosynthetic microorganism growth: random walk vs. finite difference method. *Math. Comput. Simul.* **82**(10), 2022–2032 (2012)
17. Papáček, S., Jablonsky, J., Petera, K.: Advanced integration of fluid dynamics and photosynthetic reaction kinetics for microalgae culture systems (2017, submitted)
18. Rehák, B., Čelikovský, S., Papáček, Š.: Model for photosynthesis and photoinhibition: parameter identification based on the harmonic irradiation O_2 response measurement. In: Joint Special Issue of TAC IEEE and TCAS, 101–108. IEEE (2008)
19. Richmond, A.: Biological principles of mass cultivation. In: Richmond, A. (ed.) *Handbook of Microalgal Culture: Biotechnology and Applied Phycology*, pp. 125–177. Blackwell Publishing, Hoboken (2004)
20. Taylor, G.I.: Stability of a viscous liquid containing between two rotating cylinders. *Phil. Trans. R. Soc.* **A223**, 289–343 (1923)
21. Terry, K.L.: Photosynthesis in modulated light: quantitative dependence of photosynthetic enhancement on flashing rate. *Biotechnol. Bioeng.* **28**, 988–995 (1986)
22. Wu, X., Merchuk, J.C.: A model integrating fluid dynamics in photosynthesis and photoinhibition processes. *Chem. Eng. Sci.* **56**(11), 3527–3538 (2001)
23. Wu, X., Merchuk, J.C.: Simulation of algae growth in a bench-scale bubble column reactor. *Biotechnol. Bioeng.* **80**, 156–168 (2002)
24. Celik, I., Ghia, U., Roache, P., Freitas, C., Coleman, H., Raad, P.: *J. Fluids Eng.* **130** (2008)



Karhunen-Loève Decomposition of Isotropic Gaussian Random Fields Using a Tensor Approximation of Autocovariance Kernel

Michal Bérés^{1,2,3}(✉)

¹ Department of Applied Mathematics, Faculty of Electrical Engineering and Computer Science, VŠB - Technical University of Ostrava,

Ostrava, Czech Republic

`michal.beres@vsb.cz`

² IT4Innovations National Supercomputing Center,

VŠB - Technical University of Ostrava, Ostrava, Czech Republic

³ Institute of Geonics of the CAS, Ostrava, Czech Republic

Abstract. Applications of random fields typically require a generation of random samples or their decomposition. In this contribution, we focus on the decomposition of the isotropic Gaussian random fields on a two or three-dimensional domain. The preferred tool for the decomposition of the random field is the Karhunen-Loève expansion. The Karhunen-Loève expansion can be approximated using the Galerkin method, where we encounter two main problems. First, the calculation of each element of the Galerkin matrix is expensive because it requires an accurate evaluation of multi-dimensional integral. The second problem consists of the memory requirements, originating from the density of the matrix. We propose a method that overcomes both problems. We use a tensor-structured approximation of the autocovariance kernel, which allows its separable representation. This leads to the representation of the matrix as a sum of Kronecker products of matrices related to the one-dimensional problem, which significantly reduces the storage requirements. Moreover, this representation dramatically reduces the computation cost, as we only calculate two-dimensional integrals.

Keywords: Random fields sampling
Karhunen-Loève decomposition · Tensor approximation
Numerical integration

1 Introduction

In mathematical modeling, we can encounter the need to analyze processes in some physical domain $\mathcal{D} \subset \mathbb{R}^d$ with only stochastic knowledge about the material. We can say that the material properties are described as a random field. By the random field on $\mathcal{D} \subset \mathbb{R}^d$ we understand a real valued function $X(x, \omega)$,

which for every fixed $x \in \mathcal{D}$ results in a random variable and for every fixed ω from the sample space Ω results in a function defined on \mathcal{D} , e.g. a function from $L^2(\mathcal{D})$.

A common and natural type of random field is the Gaussian random field (GRF). For a GRF $\forall x \in \mathcal{D} : X(x, \omega) \sim N(\mu(x); \sigma(x))$. GRF can be fully described by its mean value $\mu(x)$ and autocovariance function

$$c(x, y) = \mathbb{E}((X(x, \omega) - \mu(x)) \cdot (X(y, \omega) - \mu(y))).$$

Here we focus only on isotropic GRF, which are specified by an autocovariance function that depends only on physical distance of x and y . An example of the isotropic GRF is the behaviour of the porosity in samples of porous rocks like sandstone, see [7].

In typical applications, we usually require realizations (samples) of random field $X(x, \omega)$ (a sample is understood as a realization of $X(x, \omega)$ for some $\omega \in \Omega$) or a decomposition of the random field in the form of

$$X(x, \omega) \simeq \mu(x) + \sum_{i=1}^N \psi_i(x) \cdot \xi_i(\omega), \quad (1)$$

where $\|\psi_i(x)\|$ should be rapidly decreasing with increasing value of i .

A random field is an infinite-dimensional object, therefore we first need to perform a discretization. Basically, there are two ways of random field discretization: Point discretization and Karhunen-Loève decomposition.

The point discretization examines the GRF only on some finite set of the domain points $\{x_1, \dots, x_N\} \subset \mathcal{D}$. It leads to a random vector representation of the studied random field. For a sampling the random vector, the only difficult part is to take into account the covariance, which can be done through eigenvalue [1] or Cholesky decomposition [11] of the covariance matrix, use of circular embedding method [1] or Krylov subspace sampling method [2, 3].

The Karhunen-Loève decomposition (KLD) leads straightforwardly to the aforementioned decomposition form (1) of the random field. In this article we focus on the KLD and techniques for its efficient calculation. The article is a continuation of the work presented in [12].

This paper presents an effective approach to the KLD of isotropic GRF on a multi-dimensional interval. It consists of

- a tensor approximation of the isotropic autocovariance kernel (Subsect. 3.1),
- Galerkin approximation of the KLD of the tensor approximation (Sect. 2)
- and effective numerical integration in the construction of Galerkin matrices (Subsect. 3.2).

Proposed approach is supported by the numerical experiments in Sect. 4. We conclude in Sect. 5.

2 The Karhunen-Loève Decomposition

In this section, we will discuss the approximation of the random field by a function of a random vector and a physical variable. This can be achieved using the truncated Karhunen-Loève decomposition. The existence of the KLD is stated in the following theorem, see [1, Theorem 7.52]. The theorem is based on the fact that the space $L^2(\Omega, L^2(\mathcal{D}))$ is isometric isomorphic with the space $L^2(\Omega) \otimes L^2(\mathcal{D})$ (see [8, 9]).

Theorem 1. *Let $\mathcal{D} \subset \mathbb{R}^d$. Consider a random field $\{k(x; \omega) : x \in \mathcal{D}\}$ and suppose that $k \in L^2(\Omega, L^2(\mathcal{D}))$. Then*

$$k(x; \omega) = \mu(x) + \sum_{j=1}^{\infty} \sqrt{\lambda_j} \psi_j(x) \xi_j(\omega), \quad (2)$$

where the sum converges in $L^2(\Omega, L^2(\mathcal{D}))$,

$$\xi_j(\omega) := \frac{1}{\sqrt{\lambda_j}} \int_{\mathcal{D}} (k(x; \omega) - \mu(x)) \psi_j(x) dx, \quad (3)$$

and $\{\lambda_j, \psi_j\}$ denotes the eigenvalues and the eigenvectors of the autocovariance operator $\mathcal{C} : L^2(\mathcal{D}) \rightarrow L^2(\mathcal{D})$

$$(\mathcal{C}f)(x) := \int_{\mathcal{D}} c(x, y) f(y) dy, \quad (4)$$

where $c(x, y) := \text{cov}(k(x; \omega), k(y; \omega))$; $\lambda_1 \geq \lambda_2 \geq \dots \geq 0$ and $\lim_{k \rightarrow \infty} \lambda_k = 0$. The random variables ξ_j have zero mean, unit variance and are pairwise uncorrelated.

Note that GRF lies in space $L^2(\Omega, L^2(\mathcal{D}))$ (see for example [1, Corollary 4.41]), therefore its KLD exists.

In the case of the KLD of the GRF, the random variables ξ_i will also be Gaussian $\xi_i \sim N(0, 1)$. Note that the random variables ξ_i are in general only uncorrelated, but for the Gaussian random variables, it implies independence.

2.1 Galerkin Method for Spectral Decomposition of the Autocovariance Operator

The most difficult part of the KLD is the spectral decomposition of the operator \mathcal{C} , which we can obtain by solving the eigenvalue problem

$$\int_{\mathcal{D}} c(x, y) \psi_i(y) dy = \lambda_i \psi_i(x), \quad \forall i \in \mathbb{N}. \quad (5)$$

This is the Fredholm integral equation, which can be solved by the Galerkin method.

We can obtain the weak formulation of the Eq. (5) by multiplying it by a test function $v \in L^2(\mathcal{D})$ and integrating over the domain \mathcal{D} .

$$\left\{ \begin{array}{l} \text{Find } \psi_i(x) \in L^2(\mathcal{D}), \lambda_i \in \mathbb{R}^+ : \\ \int_{\mathcal{D}} v(x) \int_{\mathcal{D}} c(x, y) \psi_i(y) \, dy \, dx = \\ = \lambda_i \int_{\mathcal{D}} v(x) \psi_i(x) \, dx, \forall v(x) \in L^2(\mathcal{D}) \end{array} \right. . \quad (6)$$

The next step is the discretization of the weak formulation. First consider a basis $\langle \phi_1(x), \dots, \phi_n(x) \rangle = V_n \subset L^2(\mathcal{D})$, so the Galerkin formulation takes the form of

$$\left\{ \begin{array}{l} \text{Find } \psi_i(x) = \sum_{j=1}^n \bar{\psi}_{ij} \phi_j(x), \lambda_i \in \mathbb{R}^+ : \\ \int_{\mathcal{D}} \phi_j(x) \int_{\mathcal{D}} c(x, y) \psi_i(y) \, dy \, dx = \\ = \lambda_i \int_{\mathcal{D}} \phi_j(x) \psi_i(x) \, dx, \forall \phi_j(x) \end{array} \right. , \quad (7)$$

where $\bar{\psi}_i$ is a representation of the eigenvector ψ_i in the V_n basis. The solution of (7) can be rephrased into a generalized eigenvalue problem

$$\mathbb{A} \bar{\psi}_i^n = \lambda_i^n \mathbb{W} \bar{\psi}_i^n, \quad (8)$$

where

$$\mathbb{A}_{ij} = \int_{\mathcal{D}} \int_{\mathcal{D}} c(x, y) \phi_i(y) \phi_j(x) \, dy \, dx, \quad (9)$$

$$\mathbb{W}_{ij} = \int_{\mathcal{D}} \phi_i(x) \phi_j(x) \, dx. \quad (10)$$

Note that the size of the matrix \mathbb{A} , assuming the same resolution of the discretization in each dimension, grows by the power of the dimension of the problem. The matrix \mathbb{A} is also a dense matrix and generally, we cannot achieve sparsity by some specific choice of the basis.

2.2 Problem Constraints and the Choice of the Basis

First we need to emphasize that method proposed in this paper is only applicable for domains \mathcal{D} in the form of multidimensional intervals. In this section we assume 2-dimensional domain in the form $\mathcal{D} = \langle a, b \rangle \times \langle c, d \rangle$, but the ideas behind can be easily used for more dimensional domains. We choose a discretization $\langle \phi_1(x), \dots, \phi_n(x) \rangle = V_n$ of the space $L^2(\mathcal{D})$ as a tensor product of 1-dimensional bases of the spaces $L^2(\langle a, b \rangle)$ and $L^2(\langle c, d \rangle)$. Therefore basis functions are in the form of

$$\phi_i(x) = \varphi_i^1(x_1) \varphi_i^2(x_2), \quad (11)$$

where

$$\varphi_i^1(x_1) \in V_n^1 := \langle \varphi_1^1(x_1), \dots, \varphi_{m_1}^1(x_1) \rangle$$

and

$$\varphi_i^2(x_2) \in V_n^2 := \langle \varphi_1^2(x_2), \dots, \varphi_{m_2}^2(x_2) \rangle$$

are 1-dimensional functions. The dimension of the space $V_n = V_n^1 \otimes V_n^1$ then equals $m_1 \cdot m_2$.

The multidimensional form of the domain together with basis functions in the tensor form are sufficient for the use of the tensor approximation of the autocovariance kernel. But we can add some more (not so constraining) requirements to the basis, which will lead to significant reduction of computational effort. It is useful to consider the spaces V_n^i with specific properties. These properties will be demonstrated on 1-dimensional basis denoted by $\langle \varphi_1(x), \dots, \varphi_m(x) \rangle$.

The Orthogonality of $\langle \varphi_1(x), \dots, \varphi_m(x) \rangle$. The complexity of the problem (8) decreases if we transform the generalized eigenvalue problem into a standard eigenvalue problem ($\mathbb{W} = \mathbb{I}$). We can achieve this by assuring all of the 1-dimensional bases $\langle \varphi_1(x), \dots, \varphi_m(x) \rangle$ to be orthonormal

$$\forall i, j : \int_a^b \varphi_i(x) \varphi_j(x) \, dx = \delta_{i,j}. \tag{12}$$

The Evenness/Oddness of $\langle \varphi_1(x), \dots, \varphi_m(x) \rangle$. The matrix \mathbb{A} will generally be dense, but some unique properties of the chosen autocovariance function can be utilized to obtain partial sparsity of the matrix. We will demonstrate it on the 2-dimensional example. First, we define the function $p(x, y)$, which is related to the autocovariance function by

$$p(|x_1 - y_1|, |x_2 - y_2|) = c((x_1, x_2), (y_1, y_2)). \tag{13}$$

Next, consider the translation of the integral over the domain $\mathcal{D} = \langle a, b \rangle \times \langle c, d \rangle$ into the integral over the domain $\langle -\alpha, \alpha \rangle \times \langle -\beta, \beta \rangle$, where $\alpha = \frac{b-a}{2}$ and $\beta = \frac{d-c}{2}$ (translate the center of \mathcal{D} into $(0, 0)$). Note, that this will not affect the function $p(|x_1 - y_1|, |x_2 - y_2|)$. For simplicity we denote the basis functions $\varphi_i(x) = \varphi_i^1(x_1) \varphi_i^2(x_2)$ as before the translation of the integral. After these modifications, the formula for the elements of the matrix \mathbb{A} is

$$\mathbb{A}_{i,j} = \int_{-\beta}^{\beta} \int_{-\beta}^{\beta} f(x_2, y_2) \varphi_i^2(x_2) \varphi_j^2(y_2) \, dy_2 dx_2, \tag{14}$$

where

$$f(x_2, y_2) = \int_{-\alpha}^{\alpha} \int_{-\alpha}^{\alpha} p(|x_1 - y_1|, |x_2 - y_2|) \cdot \varphi_i^1(x_1) \varphi_j^1(y_1) \, dx_1 dy_1. \tag{15}$$

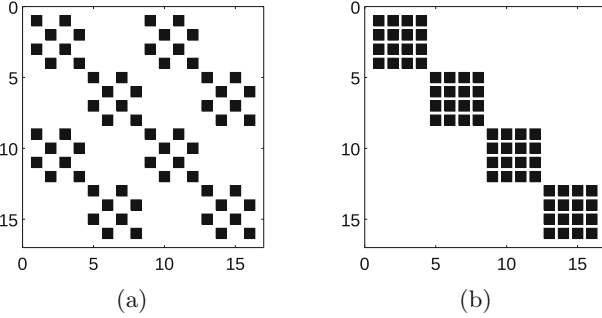


Fig. 1. (a) Standard full tensor. (b) Permuted

If $\varphi_i^1(x_1)$ is even and $\varphi_j^1(y_1)$ is odd (or vice versa), the function $f(x_2, y_2)$ is constant zero and $\mathbb{A}_{i,j} = 0$. The same holds for the functions $\varphi_i^2(x_2)$ and $\varphi_j^2(y_2)$.

The evenness/oddness (or “parity”) of the $\langle \varphi_1(x), \dots, \varphi_m(x) \rangle$ basis results into a stronger property than just a partial sparsity of the matrix \mathbb{A} . The nonzero pattern of the matrix \mathbb{A} for the standard ordering (alternately even, odd) (see Fig. 1a) can be reordered into a block diagonal structure (see Fig. 1b). The permutation can be done by reordering the basis functions according to their parity in each dimension. The block diagonal structure leads to the solution of four smaller eigenvalue problems instead of one bigger eigenvalue problem in two dimensions. Note that in n dimensions we get 2^n diagonal blocks.

The particular choice of basis functions is arbitrary. We can use e.g. polynomials [12], trigonometric functions [12], piece-wise constant functions [12], or wavelets [10].

3 Efficient Computation of the KLD Using Tensor Approximation of Autocovariance Kernel

3.1 Tensor Representation of the Autocovariance Kernel

The isotropic autocovariance function takes only physical distance (euclidean) of the two points as a parameter

$$c(x, y) = c(\|x - y\|).$$

Therefore all of its properties are described by one dimensional function taking positive parameters $c(\|\mathbf{x}\|) = c(d), d \in \mathbb{R}^+$ (for simplicity we denote $\mathbf{x} = x - y$). Than the separable representation of $c(\|\mathbf{x}\|)$ of the rank M takes the following form:

$$c(\|\mathbf{x}\|) \approx \sum_{k=1}^M g_k e^{f_k \|\mathbf{x}\|^2} = \sum_{k=1}^M g_k \prod_{d=1}^n e^{f_k x_d^2}. \tag{16}$$

The coefficients g_k, f_k can be estimated using different approaches such as sinc quadrature (see [4, 5]) or optimization. The form of the representation can be

derived using the inverse Laplace transformation of the autocovariance kernel, see [4, 5]. In this paper, we focus on the optimization approach, where we minimize the norm of the difference between the discretization of the autocovariance kernel (its 1D representation $c(d)$) and its approximation. We assume coefficients $\bar{g} = (g_1, \dots, g_M)$, $\bar{f} = (f_1, \dots, f_M)$ given as the solution of following optimization problem:

$$\arg \min_{\bar{g}, \bar{f}} \sum_{j=1}^R (c(d_j) - \tilde{c}(d_j, \bar{g}, \bar{f}))^2, \quad (17)$$

where R is number of reference points d_j and

$$\tilde{c}(d_j, \bar{g}, \bar{f}) = \sum_{k=1}^M g_k e^{-f_k d_j^2}$$

is the tensor approximation.

The choice of the points d_i is arbitrary, but it greatly affects the quality of the approximation in different regions. The optimal result would be obtained by minimizing the (squared) L^2 norm of the difference. We can have a good approximation of the L^2 norm using the Gauss–Laguerre quadrature

$$\int_0^{\infty} (c(x) - \tilde{c}(x, \bar{g}, \bar{f}))^2 dx \approx \sum_{j=1}^R (c(d_j) - \tilde{c}(d_j, \bar{g}, \bar{f}))^2 w_j,$$

which does not change the nature of the problem and provides a good choice of the points d_j .

Searching for the global minimum in (17) is very difficult because the functional has a large number of local minima and saddle points (e.g. the couples g_i, f_i are interchangeable). We propose a simple reformulation of the problem, which leads to more “well-posed” problem:

- We can separate the minimization into two levels: minimization of coefficients \bar{g} and minimization of coefficients \bar{f} . The minimization of coefficients \bar{g} is a standard least squares problem and can be expressed as a solution of a system of linear equations. The minimization of coefficients \bar{f} is non-linear, we use e.g. the Newton method only on coefficients \bar{f} (coefficients \bar{g} will be calculated (easily) in each step of the optimization method).
- The standard behaviour of the tensor approximation is that coefficients f_i grow exponentially (see Fig. 2). Origins of this behaviour can be seen in the inverse Laplace transformation of the autocovariance function, see [4, 5]. This property causes difficulties in determining the step size of the optimization method. We propose exponential transformation of the parameters:

$$f_i = e^{v_i}.$$

- Finally we can get rid of vast majority of local minima and saddle points, caused by the interchangeability of the couples g_i, f_i , by optimizing the increments ($u_1 = v_1, u_i = v_i - v_{i-1}$) of the coefficients v_i :

$$v_i = \sum_{j=1}^i u_j.$$

The final form of the rephrased tensor approximation is

$$\tilde{c}(d_j, \bar{g}, \bar{u}) = \sum_{k=1}^M g_k \exp \left(- \exp \left(\sum_{i=1}^k u_i \right) d_j^2 \right).$$

The aforementioned reformulations makes the problem easily solvable by the Newton method (gradient and hessian for \bar{u} can be expressed analytically) with reasonable starting point. The original coefficients \bar{g} and \bar{f} of tensor approximation of the exponential kernel ($c(x, y) = e^{-\|x-y\|}$) can be seen in Fig. 2.

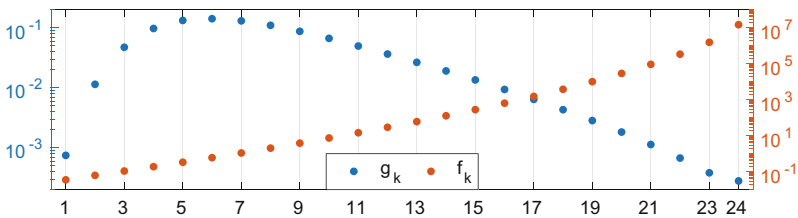


Fig. 2. Coefficients of the tensor representation of rank 24

3.2 Calculation of the Galerkin Matrix

Once we have the autocovariance kernel in the following tensor form:

$$c(\|\mathbf{x}\|) \approx \sum_{k=1}^M g_k \cdot \exp \left(-f_k \cdot \|\mathbf{x}\|^2 \right) = \sum_{k=1}^M g_k \cdot \prod_{d=1}^n \exp \left(-f_k \cdot x_d^2 \right),$$

and test functions also in the tensor form:

$$\phi_i(\mathbf{x}) = \prod_{d=1}^n \phi_i^{(d)}(x_d),$$

we can rewrite the calculation of matrix \mathbb{A} elements in the following way:

$$\mathbb{A}_{ij} = \int_{\mathcal{D}} \int_{\mathcal{D}} \left[\sum_{k=1}^M g_k \cdot \prod_{d=1}^n \exp \left(-f_k \cdot (x_d - y_d)^2 \right) \right] \cdot \prod_{d=1}^n \phi_i^{(d)}(y_d) \cdot \phi_j^{(d)}(x_d) dy dx$$

$$\mathbb{A}_{ij} = \sum_{k=1}^M g_k \prod_{d=1}^n \int_{a_d}^{b_d} \int_{a_d}^{b_d} \exp\left(-f_k \cdot (x_d - y_d)^2\right) \cdot \phi_i^{(d)}(y_d) \cdot \phi_j^{(d)}(x_d) dy_d dx_d.$$

This means that matrix \mathbb{A} can be assembled using the Kronecker product of smaller sub-matrices:

$$\mathbb{A} = \sum_{k=1}^M g_k \cdot \bigotimes_{d=1}^n \mathbf{A}_{k,d},$$

$$(\mathbf{A}_{k,d})_{ij} = \int_{a_d}^{b_d} \int_{a_d}^{b_d} \exp\left(-f_k \cdot (x_d - y_d)^2\right) \cdot \phi_i^{(d)}(y_d) \cdot \phi_j^{(d)}(x_d) dy_d dx_d. \quad (18)$$

The calculation of sub-matrices can be numerically challenging because the coefficients f_k can be very high. Therefore we propose a substitution method based on the Duffy transformation (see [6]). For simplicity we show the substitution on $(0, 1)$. First, the high value of the coefficient f_k concentrates the whole information to the diagonal of the square $\langle 0, 1 \rangle^2$ (function $\exp\left(-f_k \cdot (x_d - y_d)^2\right)$ is almost zero everywhere except $x = y$). We tear the square domain into two triangles and transform them back into the same squares:

$$\int_0^1 \int_0^1 \exp\left(-f \cdot (x - y)^2\right) \cdot \phi(x) \cdot \psi(y) dy dx =$$

$$\int_0^1 \int_0^1 x \cdot \exp\left(-f \cdot x^2 \cdot y^2\right) \cdot [\phi(x) \cdot \psi((1 - y) \cdot x) + \phi(1 - x) \cdot \psi((y - 1) \cdot x + 1)] dy dx.$$

This shifts the area of information to the edges of the domain ($x = 0$ or $y = 0$). Then we can stretch the area of the information by the substitution $x = a^n, y = b^n$, where n is arbitrary. The parameter n should be chosen according to the value of f_k .

In standard applications of the KLD, we only need first few most significant elements of the decomposition. This corresponds to the calculation of the first few eigenpairs $\{\lambda_j, \psi_j\}$. In common iterative methods for partial eigenvalue decomposition, we only need to effectively perform a matrix-vector multiplication $\mathbb{A}\mathbf{v}$ with

$$\mathbb{A} = \sum_{k=1}^M g_k \cdot \bigotimes_{d=1}^n \mathbf{A}_{k,d}, \quad \mathbf{v} \in \mathbb{R}^m,$$

where $m = m_1 \cdot \dots \cdot m_n$.

3.3 Kronecker Product Matrix Multiplication

This can be done very effectively without explicit construction of the matrix \mathbb{A} . In the case of $n = 2$, the multiplication is performed as

$$\mathbb{A}\mathbf{v} = \text{vec} \left(\sum_{k=1}^M g_k \mathbf{A}_{k,2} \bar{\mathbf{v}} \mathbf{A}_{k,1}^T \right), \quad (19)$$

where $\bar{\mathbf{v}} \in \mathbb{R}^{m_2 \times m_1}$ is the input vector reshaped to the matrix (column-wise) and function $\text{vec}(\cdot)$ reshape matrix $m_2 \times m_1$ back to vector. In higher dimensions we treat vector \mathbf{v} as a tensor $\mathbf{V} \in \mathbb{R}^{m_1 \times \dots \times m_n}$. In each summand we sequentially fold-unfold \mathbf{V} according to the corresponding dimension and multiply by the matrices $\mathbf{A}_{k,d}$. The unfolding scheme is illustrated in Fig. 3.

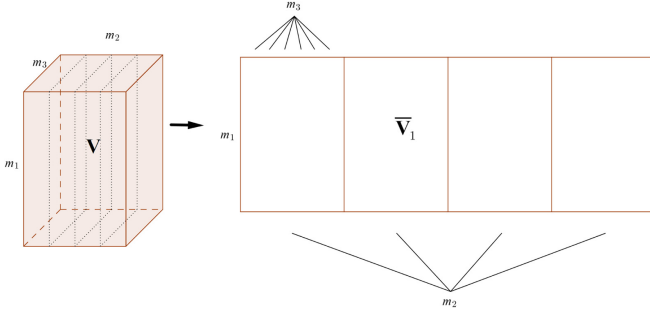


Fig. 3. Unfolding of tensor into 2D matrix

4 Numerical Results

This section is devoted to the numerical experiments of the aforementioned methods. It is divided into three parts: numerical tests of the optimization procedure for tensor approximation, numerical tests of the effectiveness of the numerical integration in the calculation of sub-matrices and numerical tests of the precision of sampling and eigenvalue approximation by tensor approximation.

4.1 Examples of Autocovariance Functions

For testing purposes we use some standard isotropic autocovariance functions (kernels):

- Squared exponential covariance function

$$c(\mathbf{x}, \mathbf{y}) = e^{-\lambda \|\mathbf{x} - \mathbf{y}\|^2}, \quad \lambda > 0 \quad (20)$$

- Exponential covariance function (Matérn autocovariance for $\nu = 1/2$)

$$c(\mathbf{x}, \mathbf{y}) = e^{-\lambda \|\mathbf{x} - \mathbf{y}\|}, \quad \lambda > 0 \quad (21)$$

– Matérn family of covariance functions

$$c(\mathbf{x}, \mathbf{y}) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \|\mathbf{x} - \mathbf{y}\|\right)^\nu K_\nu \left(\sqrt{2\nu} \|\mathbf{x} - \mathbf{y}\|\right) \tag{22}$$

where $\nu > 0$, Γ is the gamma function, K_ν is the modified Bessel function of the second kind.

- e.g. Matérn autocovariance for $\nu = 3/2$ is

$$c(\mathbf{x}, \mathbf{y}) = \left(1 + \sqrt{3}\lambda \|\mathbf{x} - \mathbf{y}\|\right) e^{-\sqrt{3}\lambda \|\mathbf{x} - \mathbf{y}\|}$$

and Matérn autocovariance for $\nu = 5/2$ is

$$c(\mathbf{x}, \mathbf{y}) = \left(1 + \sqrt{5}\lambda \|\mathbf{x} - \mathbf{y}\| + \frac{5}{3}\lambda^2 \|\mathbf{x} - \mathbf{y}\|^2\right) e^{-\sqrt{5}\lambda \|\mathbf{x} - \mathbf{y}\|}$$

Mentioned autocovariance functions together with their eigenvalues for $\mathcal{D} = \langle 0, 5 \rangle$ can be seen in the following Fig. 4.

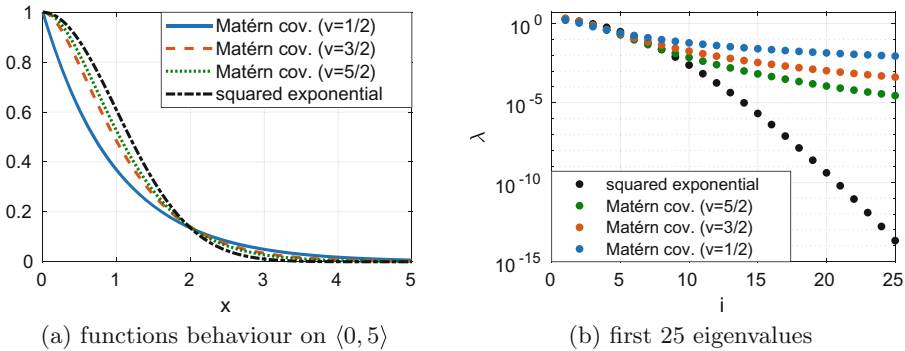


Fig. 4. Examples of the most popular autocovariance functions

4.2 Optimization Approach to the Tensor Approximation

In this part, we show the precision of the tensor approximation. We measure the approximation error as L^2 norm of the difference between original c and its tensor approximation \tilde{c} :

$$\|c - \tilde{c}\|_{L^2} = \sqrt{\int_0^\infty (c(x) - \tilde{c}(x))^2 dx.}$$

The behaviour of the quality of the tensor approximation when increasing rank can be seen in Fig. 5. We can see that with increasing rank we get almost exponential convergence. Another observation is that speed of the convergence is equivalent to the rate of decay of autocovariance kernel.

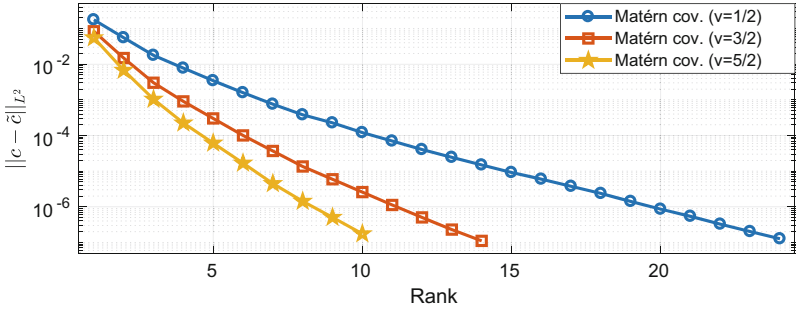


Fig. 5. Tensor approximation precision

4.3 Numerical Integration in Calculation of Sub-matrices $A_{k,d}$

In this part we test the approximation error of numerical integration of $A_{k,d}$ (Eq. (18)). For comparison, we use three different integration schemes: Rectangle method with the use of the Fast Fourier transform (see [4,5]), Gauss-Legendre quadrature and Gauss-Legendre quadrature with proposed substitution (we choose $n = 4$).

In the following numerical test, we compare the maximum of absolute errors of each of $A_{k,d}$ element using the basis of 100 Legendre polynomials. We compare the precision against the computation time for different coefficients f_k .

From the results in Fig. 6, we can see that the proposed substitution, makes the integrand “more analytic”. Therefore the Gaussian quadrature converges very fast even for very high values of f_k . The integration complexity increases only slightly when increasing the value of f_k in comparison to other approaches.

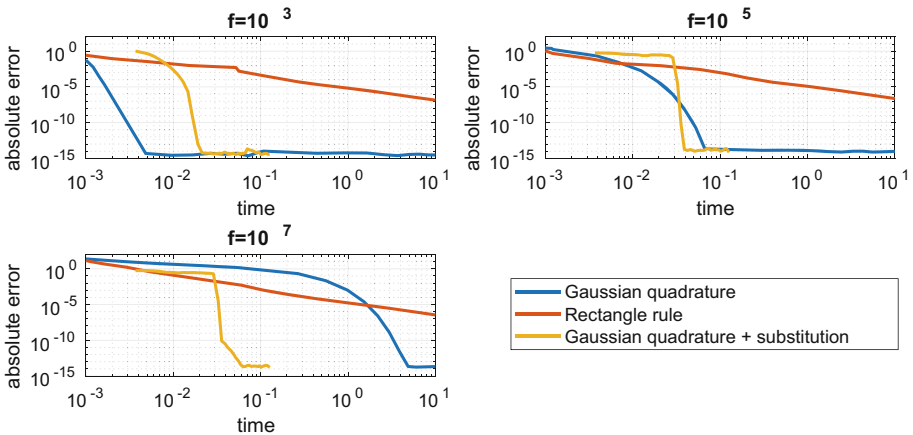


Fig. 6. Comparison of precision for different numerical integration schemes

4.4 Numerical Precision of Sampling and Eigen-Values Approximation

In the first part of the last numerical tests, we show the difference between samples from exact autocovariance kernel and its tensor approximation. We choose the exponential covariance (Matérn for $\nu = 1/2$) and $\mathcal{D} = \langle 0, 5 \rangle$ (we use only one-dimensional example because we cannot obtain a reasonable exact solution in higher dimensions). For comparison, we use a point discretization on an equidistant grid consisting of 5000 points and sampling using the eigenvalue decomposition.

In Fig. 7 we can see a realisation of one sample of $X(x, \omega)$ using exact autocovariance kernel and its low-rank approximations. We can see that relatively low-rank approximation is sufficient enough (for most applications). Another important observation is that lower rank approximations behave similarly to the truncated KLD.

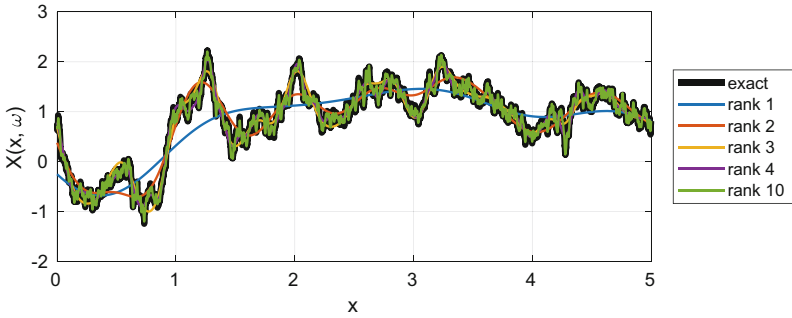


Fig. 7. Sample of random process and its low-rank approximations

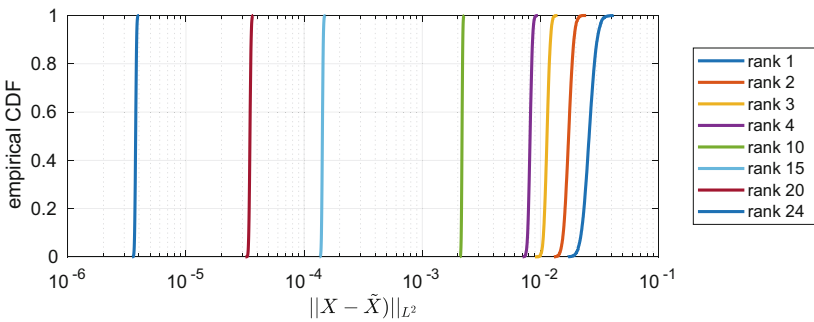


Fig. 8. Behaviour of the sampling error of low-rank approximations

In Fig. 8 we can see more extensive test examining L^2 error of the whole sample. We show empirical cumulative distribution functions (CDF) of approximation error based on 10000 samples. We can see that tensor approximations have very nice behaviour when increasing rank.

In the very last numerical test, we show the precision of the eigenvalue calculation using a low-rank approximation.

In Fig. 9 we can see exact eigenvalues and their tensor approximations. We can observe, that lower rank approximations have the faster decay of eigenvalues. In Fig. 10 we see the absolute error of tensor approximations. We can again observe a very nice behaviour of the approximations when increasing rank.

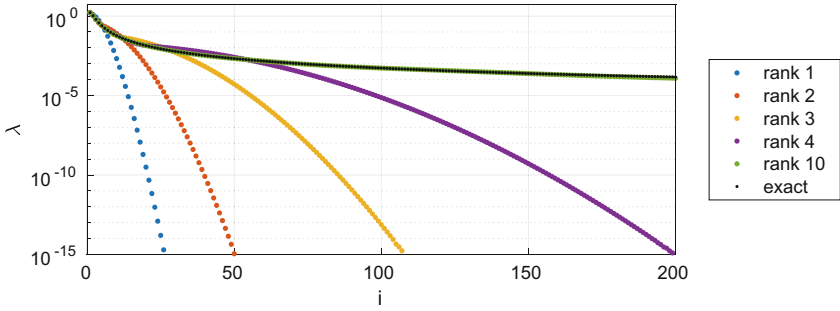


Fig. 9. Eigenvalues and their low-rank approximations

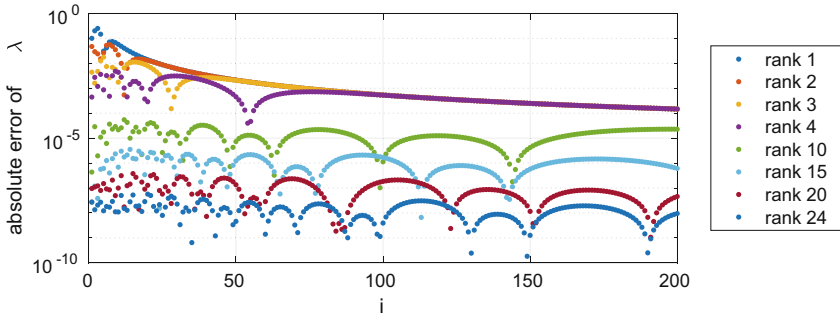


Fig. 10. Error of low-rank approximations of eigenvalues

5 Conclusions

This paper presents an effective approach to decomposition of isotropic GRF on a multi-dimensional interval. Our approach greatly reduces the memory and computation complexity of the KLD of random fields in 2D, 3D and higher dimensions in orders of magnitude. It also provides an opportunity for parallel implementation (calculation of sub-matrices $\mathbf{A}_{k,d}$ and matrix multiplication can be easily parallelized).

We also provided extensive numerical tests examining the construction of tensor approximation, the efficiency of numerical integration and precision of tensor approximation in sampling and eigenvalue approximation.

This paper has two main contributions. The first is in use of optimization approach to the tensor approximation calculation using Gauss–Laguerre quadrature and reformulation of the problem. The second consists in the substitution method for efficient calculation of sub-matrices $\mathbf{A}_{k,d}$ with coefficients f_k taking very high value.

Acknowledgment. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science - LQ1602”. The work was also partially supported by the project LD15105 “Ultrascale computing in geo-sciences”. The authors were also supported by Grant of SGS No. SP2017/56, VŠB - Technical University of Ostrava, Czech Republic.

References

1. Lord, G.J., Powell, C.E., Shardlow, T.: An Introduction to Computational Stochastic PDEs. Cambridge University Press, New York (2014)
2. Aune, E., Eidsvik, J., Pokern, Y.: Iterative numerical methods for sampling from high dimensional Gaussian distributions. *Stat. Comput.* **23**(4), 501–521 (2013)
3. Chow, E., Saad, Y.: Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions. *SIAM J. Sci. Comput.* **36**(2), A588–A608 (2014)
4. Mrovec, M.: Tensor approximation of Slater-type orbital basis functions. *Adv. Electr. Electron. Eng.* **15**(2), 314–321 (2017)
5. Mrovec, M.: Low-rank tensor representation of Slater-type and hydrogen-like orbitals. *Appl. Math.* **62**(6), 679–698 (2017)
6. Duffy, M.G.: Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM J. Numer. Anal.* **19**(6), 1260–1262 (1982)
7. Hoeksema, R.J., Kitanidis, P.K.: Analysis of the spatial structure of properties of selected aquifers. *Water Resour. Res.* **21**(4), 563–572 (1985)
8. Light, W.A., Cheney, E.W.: Approximation Theory in Tensor Product Spaces. Springer, New York (1985). <https://doi.org/10.1007/BFb0075391>
9. Da Prato, G., Zabczyk, J.: Second Order Partial Differential Equations in Hilbert Spaces. Cambridge University Press, Cambridge (2002)
10. Oliveira, S.P., Wisniewski, F., Azevedo, J.S.: A wavelet Galerkin approximation of Fredholm integral eigenvalue problems with bidimensional Haar functions. *Proc. Ser. Braz. Soc. Comput. Appl. Math.* **2**(1), 010060-1–010060-6 (2014)
11. Blaheta, R., Béréš, M., Domesová, S.: A study of stochastic FEM method for porous media flow problem. In: Applied Mathematics in Engineering and Reliability: Proceedings of the 1st International Conference on Applied Mathematics in Engineering and Reliability, p. 281 (2016)
12. Béréš, M., Domesová, S.: The stochastic Galerkin method for Darcy flow problem with log-normal random field coefficients. *Adv. Electr. Electron. Eng.* **15**(2), 267–279 (2017)



A Bayesian Approach to the Identification Problem with Given Material Interfaces in the Darcy Flow

Simona Domesová^{1,2,3}(✉) and Michal Bérés^{1,2,3}

¹ Department of Applied Mathematics,
Faculty of Electrical Engineering and Computer Science,
VŠB - Technical University of Ostrava, Ostrava, Czech Republic
simona.domesova@vsb.cz

² IT4Innovations National Supercomputing Center,
VŠB - Technical University of Ostrava, Ostrava, Czech Republic

³ Institute of Geonics of the CAS, Ostrava, Czech Republic

Abstract. The contribution focuses on the estimation of material parameters on subdomains with given material interfaces in the Darcy flow problem. For the estimation, we use the Bayesian approach, which incorporates the natural uncertainty of measurements. The main interest of this contribution is to describe the posterior distribution of material parameters using samples generated by the Metropolis-Hastings method. This method requires a large number of direct problem solutions, which is time-consuming. We propose a combination of the standard direct solutions with sampling from the stochastic Galerkin method (SGM) solution. The SGM solves the Darcy flow problem with random parameters as additional problem dimensions. This leads to the solution in the form of a function of both random variables and space variables, which is computationally expensive to obtain, but the samples are very cheap. The resulting sampling procedure is applied to a model groundwater flow inverse problem as an alternative to the existing deterministic approach.

Keywords: Bayesian inversion · Darcy flow · Metropolis-Hastings Identification problem · Posterior distribution
Uncertainty quantification

1 Introduction

By an identification problem we understand an inverse problem of determining material parameters of a boundary value problem. A solver for the direct problem is available, but the inversion is unknown. Problems of this kind are present in various engineering areas. Here, the identification problem is in fact a groundwater flow inverse problem. In the case of the direct problem, the material on the domain is given and the pore pressure and Darcy's velocity is calculated afterwards. On the contrary, in the case of the inverse problem, only

some measurements of the pore pressure or the Darcy's velocity are given and the objective is to identify the parameters of the material, i.e. the parameters that specify the material field on the given domain. Generally, if we consider a material in the form of the Gaussian random field or its function, the number of material parameters would be infinite. It would be necessary to approximate the material field using finite number of parameters first, e.g. using a proper decomposition, see [1]. Here we consider constant materials on subdomains with given interfaces, so we do not have to deal with this issue.

The identification problem is to find a vector $\mathbf{u} \in \mathbb{R}^n$ of unknown material parameters that satisfies

$$\mathbf{y} = G(\mathbf{u}) + \boldsymbol{\eta},$$

where $\mathbf{y} \in \mathbb{R}^m$ is a vector of measured values, $G : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is the observation operator that includes the solution of the boundary value problem and $\boldsymbol{\eta} \in \mathbb{R}^m$ represents the additive noise of the measurements. Generally, we can distinguish two approaches to the solution of identification problems, the deterministic and the stochastic one. Since in the real engineering applications the uncertainties in measurements are almost inevitable, we consider the stochastic approach. We solve the identification problem using the Bayesian framework, which presumes that the measurements are corrupted by noise. Therefore, the solution of the inverse problem formulated in the Bayesian way is not a point estimate, but a joint probability density of the vector of the uncertain material parameters.

A complication lies in the fact that the resulting probability distribution depends on the solution of the direct problem, i.e. on the solution of some partial differential equation. Therefore, the resulting joint probability density cannot be expressed analytically. Even samples cannot be provided directly, we have to use advanced sampling techniques. Here we consider sampling using the Metropolis-Hastings (MH) algorithm. This Markov chain Monte Carlo (MCMC) method is well suitable for this purpose, it is simple to use and to implement. But the drawback of the standard MH algorithm lies in the need of repeated solutions of the direct boundary value problem. Many techniques and modifications of the MH method have been developed to overcome this problem. Usually they are based on the construction of the approximation of the observation operator G and they differ in the way of using this approximation. For example [2] suggests a multi-level approach that works with the solution of the boundary value problem on a set of coarser grids. The paper [3] proposes a procedure of model error iterative updating. In the paper [4], a general modification of the MH algorithm with approximation is described. Furthermore it is proven that this modification does not change the limiting distribution of the resulting Markov chain, which makes this approach attractive to use.

This contribution focuses on the Darcy flow problem with given material interfaces. For this problem, we can compute the stochastic Galerkin method solution. The SGM provides a fairly good approximation of the direct problem solution for a low computation time per one material sample. In the following, we describe and analyze the use of the MH algorithm in combination with the SGM.

2 Bayesian Solution of the Inverse Problem

The solution using the Bayesian inversion differs from the deterministic approach, both in dealing with the input information and the interpretation of the results. The Bayesian approach leads to the posterior probability distribution of the random vector of material parameters. Therefore, the uncertainty of the measurements is incorporated. Further information about the unknown parameters can be also taken into account in the form of the prior distribution. The deterministic formulation of the inverse problem may not have a unique solution; however, the Bayesian approach naturally overcomes this problem. A comprehensive theoretical background for the Bayesian approach to the solution of inverse problems can be found in [5,6].

It is assumed that the distribution of the measurement error is known. For the additive noise it is natural to assume the Gaussian distribution with zero mean, i.e.

$$\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}_m; \Sigma_{\boldsymbol{\eta}}),$$

where $\mathbf{0}_m$ denotes a zero vector of length m and $\Sigma_{\boldsymbol{\eta}}$ denotes the covariance matrix.

The prior distribution of the random parameters expresses our prior belief about the material parameters. In the case of the Darcy flow boundary problem, we are speaking about the hydraulic conductivity parameters. In fact, we consider the logarithm of the hydraulic conductivity, which represents the porosity in this model. For the porosity, it is suitable to consider the Gaussian distribution, see [7]. Here, we expect apriori that the distribution of the random parameters is Gaussian with independent components, i.e. the prior information says that

$$\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}; \sigma^2 I_n),$$

where $\boldsymbol{\mu}$ is the prior mean, $\sigma > 0$ is the prior standard deviation (the same for all components of the random vector) and I_n is the identity matrix of size $n \times n$.

Based on the observed data and the prior information, the Bayesian theorem says that the posterior probability density function of the material parameters has the form

$$\begin{aligned} \pi(\mathbf{u}|\mathbf{y}) &\propto f_{\boldsymbol{\eta}}(\mathbf{y} - G(\mathbf{u})) \pi_0(\mathbf{u}) \propto \\ &\propto \exp\left(-\frac{(\mathbf{y} - G(\mathbf{u}))^T \Sigma_{\boldsymbol{\eta}}^{-1} (\mathbf{y} - G(\mathbf{u}))}{2} - \frac{\|\mathbf{u} - \boldsymbol{\mu}\|^2}{2\sigma^2}\right), \end{aligned} \tag{1}$$

where $f_{\boldsymbol{\eta}}$ denotes the joint probability density function of the noise and π_0 denotes the prior probability density function. Note that even though the probability density is known only up to a normalizing constant, it is fully specified by the formula (1).

Finding the formula for the posterior distribution of the material parameters was the first step of the Bayesian framework. At this point, we can proceed to the sampling procedure and generate samples from (1).

3 Sampling Using the MH Algorithm

For sampling from the posterior probability density function that is known only up to a normalizing constant, we can use the MH algorithm. Here we just briefly outline the principle of the method; for a detailed description of the MH algorithm, see e.g. [8]. The standard MH algorithm is based on proposing a chain of samples from an instrumental density. Each proposed sample is either accepted with a calculated acceptance probability or rejected, i.e. the previous sample is kept. This way a Markov chain with the limiting distribution $\pi(\mathbf{u}|\mathbf{y})$ is constructed.

Each step of the Markov chain requires the calculation of the acceptance probability, which includes the evaluation of the observation operator G for the proposed material sample. Therefore, the standard MH algorithm is computationally intensive. To reduce the computational complexity, we can consider an approximation \tilde{G} of the observation operator G and use it to enhance the MH algorithm. A straightforward way is to keep the MH algorithm unchanged and replace all the G evaluations by the approximated solutions of \tilde{G} ; however, this procedure changes the limiting distribution of the resulting Markov chain.

3.1 Modified MH Algorithm

To obtain the correct limiting distribution $\pi(\mathbf{u}|\mathbf{y})$, we use the modification of the MH algorithm, which is described in [4]. This modified MH algorithm works with an approximation $\tilde{\pi}$ of the posterior distribution π . Here, we define $\tilde{\pi}$ using the approximation of the observation operator, i.e.

$$\tilde{\pi}(\mathbf{u}|\mathbf{y}) \propto \exp\left(-\frac{(\mathbf{y} - \tilde{G}(\mathbf{u}))^T \Sigma_{\eta}^{-1} (\mathbf{y} - \tilde{G}(\mathbf{u}))}{2} - \frac{\|\mathbf{u} - \boldsymbol{\mu}\|^2}{2\sigma^2}\right).$$

We assume the symmetric instrumental density

$$q(\mathbf{u}, \mathbf{x}) = \frac{1}{\sqrt{(2\pi\sigma_{\text{MH}}^2)^n}} \exp\left(-\frac{(\mathbf{u} - \mathbf{x})^T (\mathbf{u} - \mathbf{x})}{2\sigma_{\text{MH}}^2}\right), \tag{2}$$

which leads to the following algorithm of the modified MH method:

- Choose \mathbf{u}_0 .
- For $t = 0, 1, \dots, T$
 - generate \mathbf{x} from $q(\mathbf{u}_t, \mathbf{x})$ and calculate the “pre”-acceptance probability

$$\tilde{\alpha}(\mathbf{u}_t, \mathbf{x}) = \min\left\{1, \frac{\tilde{\pi}(\mathbf{x}|\mathbf{y})}{\tilde{\pi}(\mathbf{u}_t|\mathbf{y})}\right\}, \tag{3}$$

- if the sample \mathbf{x} is “pre”-accepted, calculate the acceptance probability

$$\alpha(\mathbf{u}_t, \mathbf{x}) = \min\left\{1, \frac{\pi(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{u}_t|\mathbf{y})} \frac{\tilde{\pi}(\mathbf{u}_t|\mathbf{y})}{\tilde{\pi}(\mathbf{x}|\mathbf{y})}\right\}, \tag{4}$$

- set $\mathbf{u}_{t+1} = \mathbf{x}$ with probability $\alpha(\mathbf{u}_t, \mathbf{x})$, otherwise set $\mathbf{u}_{t+1} = \mathbf{u}_t$.

The last part of the sampling process does not differ from the standard MH algorithm. The autocovariance of the resulting Markov chain is analyzed and subsequently, certain number of samples is kept as final (almost uncorrelated).

3.2 Efficiency of the Sampling Process

Note that the calculation of the “pre”-acceptance probability does not contain the evaluation of the observation operator G . Only when a sample is “pre”-accepted, the acceptance probability is recalculated using G . Therefore, the settings of the algorithm should ensure a low “pre”-acceptance probability (3) and a high acceptance probability (4). A low “pre”-acceptance probability leads to a weakly autocorrelated Markov chain. And further a high acceptance probability causes that the computation time is not wasted by rejecting many samples, for which the expensive operator G has been evaluated.

The aim of the sampling process is to reduce the average number of the G evaluations per one final sample. By changing the variance parameter σ_{MH}^2 of the instrumental density (2), we can influence the acceptance rate, i.e. the ratio between the number of the accepted samples and the length of the Markov chain. In the numerical experiments (see Sect. 5.2), the dependency of the sampling process efficiency on the acceptance rate is studied.

3.3 Approximation of the Observation Operator

There are many possibilities of constructing the approximation \tilde{G} . For example, when G contains a solution of the boundary value problem using the finite element method (FEM), we can use the FEM computation on a coarser grid as the approximation. Other possibility is the stochastic collocation method (see [9, 10]) or various methods for a surrogate models construction.

In this contribution, the SGM is used for the approximation of the FEM solution of the Darcy flow problem. For example in the case of the model problem from Sect. 5, the SGM provides approximately $100\times$ to $600\times$ faster solution in comparison to the FEM, depending on the settings of the SGM method. Therefore, this modification of the MH algorithm promises a significant reduction of the computation time in comparison to the standard MH algorithm.

4 Stochastic Galerkin Method

By the SGM, we understand the standard Galerkin method applied both to the physical domain and to the parameter space, for a detailed description see [11].

We consider the Darcy flow problem including the random parameters given by the following equations. For simplicity, zero Dirichlet condition is considered on the whole boundary:

$$\begin{cases} -\operatorname{div}(k(x; \mathbf{Z}) \cdot \nabla p(x; \mathbf{Z})) = f(x) & \forall x \in \mathcal{D}, \forall \mathbf{Z} \\ p(x; \mathbf{Z}) = 0 & \forall x \in \Gamma_D, \forall \mathbf{Z}, \end{cases} \quad (5)$$

where $\mathbf{Z} = (Z_1, \dots, Z_n)$ is a vector of independent normal random variables $Z_i \sim \mathcal{N}(0; 1)$ for $i \in \{1, \dots, n\}$.

In this case, we consider constant materials on n subdomains Ω_i with given interfaces and log-normal prior distribution of each component of the material. Therefore, the random material $k(x; \mathbf{Z})$ has the form of

$$k(x; \mathbf{Z}) = k_0(x) + \sum_{i=1}^n k_i(x) \cdot g_i(\mathbf{Z}),$$

where $k_0(x)$ consists of the mean material field, $k_i(x)$ is the characteristic function of the subdomain Ω_i and

$$g_i(\mathbf{Z}) = \exp(a_i + b_i \cdot Z_i).$$

The values of a_i and b_i are parameters of the log-normal prior distribution of the i^{th} component and Z_i represent standard normal variables.

The weak formulation of the Darcy flow problem including random parameters has the form of

$$\left\{ \begin{array}{l} \text{Find } p(x; \mathbf{Z}) \in V, \forall v(x; \mathbf{Z}) \in V : \\ \int_{\mathbb{R}^N} \int_{\mathcal{D}} k(x; \mathbf{Z}) \cdot \nabla_x p(x; \mathbf{Z}) \cdot \nabla_x v(x; \mathbf{Z}) \, dx \, dF\mathbf{Z} = \int_{\mathbb{R}^N} \int_{\mathcal{D}} f(x) \cdot v(x; \mathbf{Z}) \, dx \, dF\mathbf{Z}, \end{array} \right.$$

where $V := L^2_{dF\mathbf{Z}}(\mathbb{R}^N, H_0^1(\mathcal{D})) = H_0^1(\mathcal{D}) \otimes L^2_{dF\mathbf{Z}}(\mathbb{R}^N)$. The tensor structure of the test space allows us to use the tensor product of the discretized bases of the spaces $H_0^1(\mathcal{D})$ and $L^2_{dF\mathbf{Z}}(\mathbb{R}^N)$ as the discretization of V .

4.1 Assembling the System of Equations

As the discretization of the physical domain, we choose standard linear finite elements, let us denote the basis by V_D^h and the number of the finite elements by N_D . As the discretization of the parameter space, we choose the Hermite polynomials basis, i.e. polynomials orthogonal with respect to the distribution of the parameters. We denote the basis by V_P^h and the number of the polynomials by N_P . Then the discretized test function space is given by

$$V^h := V_D^h \otimes V_P^h = \langle \varphi_1(x), \dots, \varphi_{N_D}(x) \rangle \otimes \langle \psi_1(\mathbf{Z}), \dots, \psi_{N_P}(\mathbf{Z}) \rangle \subset V.$$

The solution of the discretized problem takes the form of

$$p^h(x; \mathbf{Z}) = \sum_{i=1}^{N_D} \sum_{j=1}^{N_P} (\bar{p})_{ij} \cdot \varphi_i(x) \cdot \psi_j(\mathbf{Z}). \tag{6}$$

Using the discretization V^h , we obtain the system of linear equations

$$\mathbb{A} \cdot \bar{p} = \bar{b}. \tag{7}$$

The entries of the matrix $\mathbb{A} \in \mathbb{R}^{(N_D \cdot N_P) \times (N_D \cdot N_P)}$ and the vector $\bar{b} \in \mathbb{R}^{N_D \cdot N_P}$ are defined as

$$(\mathbb{A})_{ij,kl} = \int_{\mathbb{R}^N} \int_{\mathcal{D}} k(x; \mathbf{Z}) \cdot \nabla \varphi_i(x) \psi_j(\mathbf{Z}) \cdot \nabla \varphi_k(x) \psi_l(\mathbf{Z}) \, dx dF\mathbf{Z}, \quad (8)$$

$$(\bar{b})_{ij} = \int_{\mathbb{R}^N} \int_{\mathcal{D}} f(x) \cdot \varphi_i(x) \psi_j(\mathbf{Z}) \, dx dF\mathbf{Z}. \quad (9)$$

Using the tensor structure of the material function and the test functions, we can rewrite the calculation of \mathbb{A} into

$$\mathbb{A} = \sum_{m=0}^M G_m \otimes K_m,$$

where

$$(K_m)_{i,k} = \int_{\mathcal{D}} k_m(x) \cdot \nabla \varphi_i(x) \cdot \nabla \varphi_k(x) \, dx,$$

which arises from the discretization of the physical domain, and

$$(G_m)_{j,l} = \int_{\mathbb{R}^N} g_m(\mathbf{Z}) \cdot \psi_j(\mathbf{Z}) \cdot \psi_l(\mathbf{Z}) \, dF\mathbf{Z},$$

which follows from the parameter space discretization. Similarly \bar{b} can be rewritten as

$$\bar{b} = g \otimes f,$$

where

$$(f)_i = \int_{\mathcal{D}} f(x) \cdot \varphi_i(x) \, dx \text{ and } (g)_j = \int_{\mathbb{R}^N} \psi_j(\mathbf{Z}) \, dF\mathbf{Z}.$$

We solve the system using the reduced basis method, which involves a creation of the reduced rational Krylov approximation of the matrices K_m . For detailed description of the reduced basis method, see [13]. This work is a continuation of the research presented in [12], where we solved the Darcy flow problem with the material in the form of a random field and used iterative solvers.

5 Application to a Model Problem

The model problem is a groundwater flow inverse problem that is solved on a square domain with given material interfaces. It represents a pumping test of hydraulic conductivity in an aquifer. This inverse problem was taken from [14], where it was solved using a deterministic iterative method.

5.1 Description of the Model Problem

The material parameters are constant on each of the seven subdomains that are shown in Fig. 1; the subdomain Ω_4 consists of two separate parts. The task is to identify these seven material parameters in the Bayesian way, i.e. to estimate the posterior distribution of the random vector \mathbf{u} of length $n = 7$. The given vector of measurements \mathbf{y} contains the results of the pumping tests that serve as the input values for the identification problem.

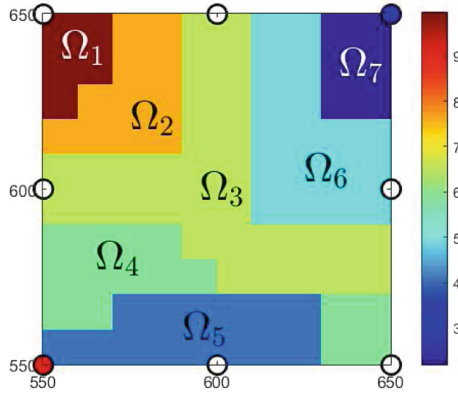


Fig. 1. Subdomains with given interfaces and the illustration of the wells (Color figure online)

The eight circles in Fig. 1 represent wells. The pumping test consists of 4 sub-tests, Fig. 1 illustrates one of them. In each sub-test, one of the wells is chosen as an injection (the red circle in Fig. 1), the opposite one is chosen as a drainage (the blue circle in Fig. 1), and the remaining six wells are points of measurements. The measured value is the pressure in the given point, which corresponds to the water column height in the well. We obtain six measured values from each sub-test, i.e. 24 measurements from the whole pumping test, possibly corrupted by noise.

To formulate the Bayesian solution of the inverse problem, the observation operator G has to be defined first. The observation operator includes the solution of the underlying boundary value problem four times for the same material sample but with different wells configurations. The square domain from Fig. 1 is surrounded by a bigger domain, see Fig. 2a. The four direct subproblems are for $i \in \{1, 2, 3, 4\}$ given by the Darcy flow boundary value problem

$$\begin{aligned}
 -\operatorname{div}(\exp(u) \cdot \nabla p_i) &= f_i \text{ in } D = \langle 0, 1200 \rangle \times \langle 0, 1200 \rangle, \\
 p_i &= 0 \text{ on } \partial D,
 \end{aligned}$$

where p_i is the pore pressure and f_i are the forcing terms that distinguish the four sub-tests. The exponential of the material function u represents the hydraulic

conductivity. For a given material, the results of the direct sub-problem are obtained by solving this boundary value problem using the FEM. This procedure defines the observation operator $G : \mathbb{R}^7 \rightarrow \mathbb{R}^{24}$ that for the material parameters on the seven subdomains returns the $m = 24$ values of the pressure,

$$G(\mathbf{u}) = (p_1, \dots, p_{24}).$$

Figure 2b shows an example of the solution of one of the sub-problems for the real material parameters

$$\mathbf{u}_{\text{real}} = (9.952, 7.601, 6.507, 5.940, 4.094, 4.956, 2.197).$$

On the surroundings, the exponential of the hydraulic conductivity has the value of 8.151, this value is not uncertain. This real material parameters were taken from [14].

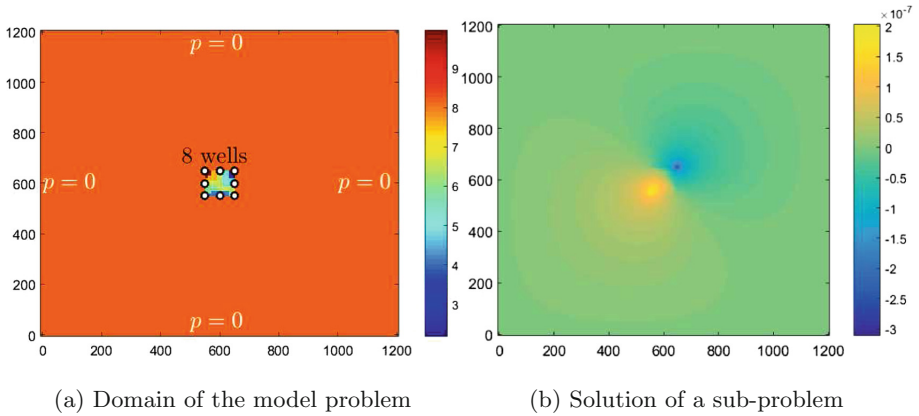


Fig. 2. Problem settings with the visualization of the real material parameters and the solution on the whole domain

To obtain the approximation \tilde{G} , the four subproblems were also solved using the SGM. For the discretization of the physical domain, the 240×240 grid was used; this yields 57121 finite elements. For the approximation in the material parameters, we used the complete Hermite polynomials on 7 random variables up to certain degree between 1 and 10. For example, there are 120 polynomials up to degree 3, which leads to a linear system with $6.8 \cdot 10^6$ unknowns; or eventually 19448 polynomials up to degree 10, which leads to $1.1 \cdot 10^9$ unknowns.

It is assumed that the vector of measurements is corrupted by the Gaussian additive noise with independent components. Therefore, Σ_η is a diagonal matrix. As the mean of the prior distribution, we choose the vector

$$\boldsymbol{\mu} = (9.787, 7.438, 6.697, 6.292, 4.291, 4.788, 1.792);$$

the same vector was taken as the initial iteration of the iterative method in the paper [14].

5.2 Numerical Experiments

In the first experiment, the SGM was used instead of the FEM in the standard MH algorithm, i.e. no FEM solutions were calculated. This experiment serves to compare the different accuracy of the SGM given by different maximal polynomial degree. As an example to visualize, we choose the resulting marginal posterior probability density functions calculated for the 5th material parameter, see Fig. 3. The final samples were fitted by the normal distribution, Table 1 shows the 95% confidence intervals for the mean and standard deviation of this distribution. In the case of the other marginal distributions, the situation was similar. These results show that the polynomials of lower degree should not replace the FEM solution in the MH algorithm, because they lead to a significantly different posterior distribution. This results emphasize the use of the modified MH algorithm that leads to the correct limiting distribution $\pi(\mathbf{u}|\mathbf{y})$.

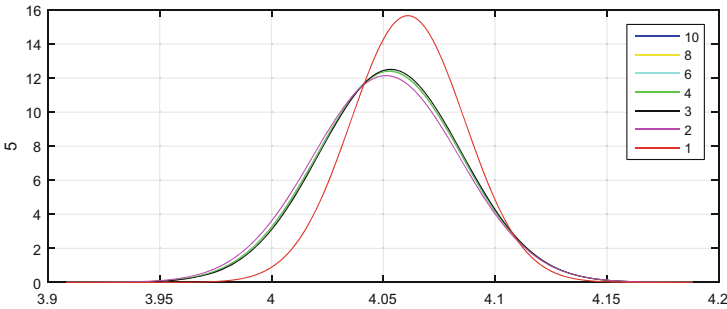


Fig. 3. Fitted normal distributions for different maximal polynomial degrees (the 5th material parameter as an example)

In the following, the samples were generated using the modified MH algorithm. For the SGM, complete polynomials up to degree 3 were used. This setting leads to a reasonable accuracy in a fairly short evaluation time. One evaluation of the approximation \tilde{G} using SGM took $1.1 \cdot 10^{-4}$ s on average. In contrast, the evaluation of the observation operator G took 0.69 s on average, using comparable implementation.

Figure 4 shows the marginal posterior distributions for the 7 material parameters approximated from the samples obtained using this procedure. The sample correlation matrix is displayed in Fig. 5. These results come from 24 independent Markov chains of length 10^6 .

In the following experiment, the Markov chain was generated several times with different variance parameters of the instrumental density. This resulted in different acceptance rates. Table 2 shows the number of FEM and SGM calculations carried out while generating this chains. It also contains the number of final samples that were kept after the autocovariance analysis and the number of FEM calculations per one final sample.

Table 1. 95% confidence intervals (c. i.) for the mean and the standard deviation (std) of the fitted normal distribution for different maximal polynomial degrees (the 5th material parameter as an example)

Maximal polynomial degree	Mean c. i.		Std c. i.	
	Lower	Upper	Lower	Upper
1	4.0607	4.0613	0.0252	0.0257
2	4.0508	4.0516	0.0325	0.0331
3	4.0529	4.0537	0.0316	0.0322
4	4.0523	4.0531	0.0319	0.0325
6	4.0522	4.0530	0.0318	0.0324
8	4.0523	4.0531	0.0318	0.0324
10	4.0522	4.0530	0.0318	0.0324

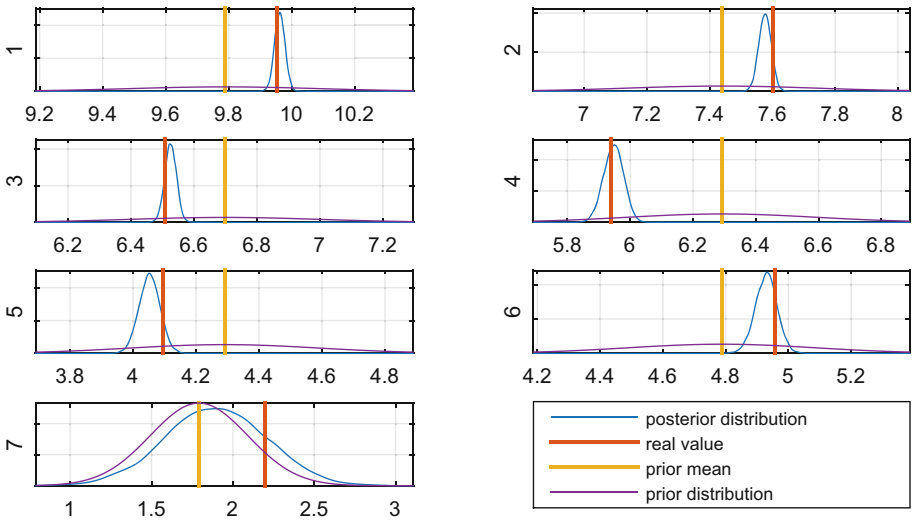


Fig. 4. Marginal posterior probability density functions for the 7 parameters

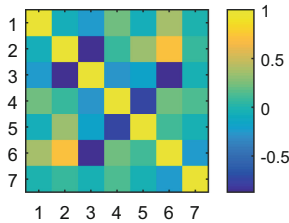
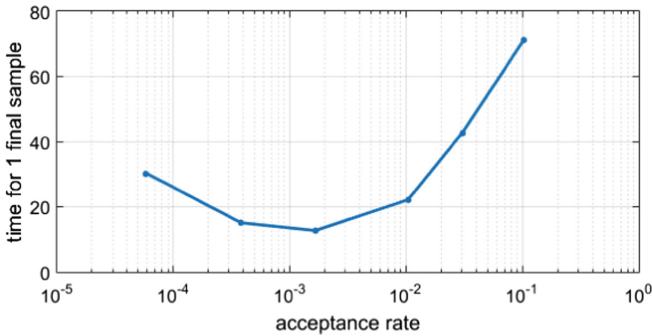


Fig. 5. Sample correlation matrix of the posterior distribution of the random vector of material parameters

Table 2. Efficiency of the MH algorithm depending on the acceptance rate

Average acceptance rate	$5.8e-5$	$3.8e-4$	$1.7e-3$	$1.0e-2$	$3.1e-2$	$1.0e-1$
Number of FEM calculations	1435	9300	40389	$2.53e5$	$7.39e5$	$2.46e6$
Number of SGM calculations	24000000					
Number of final samples	120	600	2400	7992	12000	24000
FEM calculations per one final sample	11.96	15.50	16.83	31.66	61.58	102.50
Time per one final sample [s]	30.25	15.09	12.71	22.17	42.72	70.96

For a more comprehensive comparison of the sampling efficiency, the computation time of the G and \tilde{G} evaluations should be also taken into consideration, see the last row of Table 2. Figure 6 shows the dependency of the sampling efficiency on the acceptance rate. The vertical axis indicates the total time of both the G and \tilde{G} evaluations per one final sample. The resulting curve shows that the time per one sample decreases with decreasing acceptance rate; but this only works to a certain point, from which the resulting Markov chain is too autocorrelated and a high number of samples is rejected.

**Fig. 6.** Dependence of the sampling efficiency on the acceptance rate

For comparison, the sampling from the posterior distribution was also performed using the standard MH algorithm; 240 independent Markov chains of length $4 \cdot 10^4$ were generated. Based on the autocovariance analysis, 9600 final samples were kept, i.e. the computation cost was 690s per one final sample. Therefore, the use of the modified MH algorithm reduced the computation time approximately by the factor of 50.

6 Conclusions

In this work we analyzed the possibilities of the use of the SGM for the Darcy flow inverse problem. The use of the SGM requires a solution of a large linear

system. However, the evaluation of the resulting approximation of the observation operator is much faster per one material sample than the FEM solution and accurate enough to be used with the modified MH algorithm from the paper [4].

When the SGM solution is constructed, the following sampling procedure remains simple in principle, such as the standard MH algorithm. Therefore, if the boundary value is solvable using the SGM, than the combination of the SGM and the MH algorithm leads to an effective sampling procedure. The numerical experiments confirmed that it is necessary to pay attention to the choice of the instrumental density. Here we used the symmetric instrumental density and we showed that its variance parameters influence the sampling efficiency significantly.

This approach can be understood as an alternative to deterministic numerical methods. While the deterministic approach is sensitive to the measurements accuracy, see e.g. [14], the Bayesian approach expects uncertainty in the measurements. Therefore, we obtain a comprehensive solution of the inverse problem, including the uncertainty quantification.

Acknowledgements. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science - LQ1602”. The work was also partially supported by EU under the COST programme Action IC1305 “Network for Sustainable Ultrascale Computing (NESUS)” and by the project LD15105 “Ultrascale computing in geo-sciences”. The authors were also supported by Grant of SGS No. SP2017/56, VŠB - Technical University of Ostrava, Czech Republic.

References

1. Domesová, S., Béréš, M.: Inverse problem solution using Bayesian approach with application to darcy flow material parameters estimation. *Adv. Electr. Electron. Eng.* **15**(2), 258–266 (2017)
2. Dodwell, T.J., Ketelsen, C., Scheichl, R., Teckentrup, A.L.: A hierarchical multi-level Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow. *SIAM/ASA J. Uncertain. Quantif.* **3**(1), 1075–1108 (2015)
3. Calvetti, D., Dunlop, M., Somersalo, E., Stuart, A.M.: Iterative Updating of Model Error for Bayesian Inversion. [arXiv:1707.04246](https://arxiv.org/abs/1707.04246) (2017)
4. Christen, J.A., Fox, C.: Markov chain Monte Carlo using an approximation. *J. Comput. Graph. Stat.* **14**(4), 795–810 (2005)
5. Stuart, A.M.: Inverse problems: a Bayesian perspective. *Acta Numerica* **19**, 451–559 (2010)
6. Bui-Thanh, T.: A gentle tutorial on statistical inversion using the Bayesian paradigm. Institute for Computational Engineering and Sciences, Technical report ICES-12-18 (2012)
7. Hoeksema, R.J., Kitanidis, P.K.: Analysis of the spatial structure of properties of selected aquifers. *Water Resour. Res.* **21**(4), 563–572 (1985)
8. Robert, C.: *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, Heidelberg (2007). <https://doi.org/10.1007/0-387-71599-1>

9. Nobile, F., Tempone, R., Webster, C.G.: A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.* **46**(5), 2309–2345 (2008)
10. Marzouk, Y., Xiu, D.: A stochastic collocation approach to Bayesian inference in inverse problems. Global-Science Press (2009)
11. Lord, G.J., Powell, C.E., Shardlow, T.: *An Introduction to Computational Stochastic PDEs*. Cambridge University Press, Cambridge (2014)
12. Béréš, M., Domesová, S.: The stochastic Galerkin method for darcy flow problem with log-normal random field coefficients. *Adv. Electr. Electron. Eng.* **15**(2), 267–279 (2017)
13. Powell, C.E., Silvester, D., Simoncini, V.: An efficient reduced basis solver for stochastic Galerkin matrix equations. *SIAM J. Sci. Comput.* **39**(1), A141–A163 (2017)
14. Haslinger, J., Blaheta, R., Hrtus, R.: Identification problems with given material interfaces. *J. Comput. Appl. Math.* **310**, 129–142 (2017)

Author Index

- Béřeš, Michal 188, 203
Beseda, Martin 144
Blaheta, Radim 59
Brzobohatý, Tomáš 130
- Čermák, Martin 115
- Domesová, Simona 203
Donfack, Simplicie 1
Dostál, Zdeněk 47
- Gergelits, Tomáš 73
- Hapla, Václav 115
Hnětynková, Iveta 73
Horák, David 115
- Jaros, Milan 88, 101
- Kannan, Venkatesh 144
Karasek, Tomas 88, 101
Kindermann, Stefan 160
Kozubek, Tomáš 130
Kružík, Jakub 59, 115
Kubinová, Marie 73
- Luber, Tomáš 59
Lysaght, Michael 144
- Markopoulos, Alexandros 130
Matonoha, Ctirad 160, 174
Meca, Ondřej 130
- Papáček, Štěpán 160, 174
Pecha, Marek 115
Petera, Karel 174
- Reps, Bram 1
Říha, Lubomír 130, 144
- Saad, Yousef 19
Sanan, Patrick 1
Schenk, Olaf 1
Strakos, Petr 88, 101
- Toraldo, Gerardo 47
- Ubaru, Shashanka 19
- Vanroose, Wim 1
Vasatova, Alena 88
Viola, Marco 47
Vlach, Oldřich 47
Vysocky, Ondrej 144
- Wu, Yangqingxiang 34
- Zapletal, Jan 144
Zikatanov, Ludmil 34