RHEINLAND-PFÄLZISCHE TECHNISCHE
UNIVERSITÄT KAISERSLAUTERN-LANDAU

DOCTORAL THESIS

Development and numerical study of efficient solvers for single-phase steady flows in tight porous media

---

Author:
Vladislav PIMANOV

Supervisor:
Prof. Dr. Oleg ILIEV

Reviewer 1:
Prof. Dr. Oleg ILIEV
Reviewer 2:
Prof. Dr. Stefan TUREK

Vom Fachbereich Mathematik der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau zur Verleihung des akademischen Grades ***Doktor der Naturwissenschaften (Doctor Rerum Naturalium, Dr. rer. nat.)*** genehmigte Dissertation

Disputation Date:
29 November 2023
DE-386

RHEINLAND-PFÄLZISCHE TECHNISCHE UNIVERSITÄT KAISERSLAUTERN-LANDAU

Abstract

Fachbereich Mathematik

Development and numerical study of efficient solvers for single-phase steady flows in tight porous media

by Vladislav PIMANOV

Single-phase flows are attracting significant attention in Digital Rock Physics (DRP), primarily for the computation of permeability of rock samples. Despite the active development of algorithms and software for DRP, pore-scale simulations for tight reservoirs — typically characterized by low multiscale porosity and low permeability — remain challenging. The term "multiscale porosity" means that, despite the high imaging resolution, unresolved porosity regions may appear in the image in addition to pure fluid regions. Due to the enormous complexity of pore space geometries, physical processes occurring at different scales, large variations in coefficients, and the extensive size of computational domains, existing numerical algorithms cannot always provide satisfactory results.

Even without unresolved porosity, conventional Stokes solvers designed for computing permeability at higher porosities, in certain cases, tend to stagnate for images of tight rocks. If the Stokes equations are properly discretized, it is known that the Schur complement matrix is spectrally equivalent to the identity matrix. Moreover, in the case of simple geometries, it is often observed that most of its eigenvalues are equal to one. These facts form the basis for the famous Uzawa algorithm. However, in complex geometries, the Schur complement matrix can become severely ill-conditioned, having a significant portion of non-unit eigenvalues. This makes the established Uzawa preconditioner inefficient. To explain this behavior, we perform spectral analysis of the Pressure Schur Complement formulation for the staggered finite-difference discretization of the Stokes equations. Firstly, we conjecture that the no-slip boundary conditions are the reason for non-unit eigenvalues of the Schur complement matrix. Secondly, we demonstrate that its condition number increases with increasing the surface-to-volume ratio of the flow domain. As an alternative to the Uzawa preconditioner, we propose using the diffusive SIMPLE preconditioner for geometries with a large surface-to-volume ratio. We show that the latter is much more efficient and robust for such geometries. Furthermore, we show that the usage of the SIMPLE preconditioner leads to more accurate practical computation of the permeability of tight porous media.

As a central part of the work, a reliable workflow has been developed which includes robust and efficient Stokes-Brinkman and Darcy solvers tailored for low-porosity multiclass samples and is accompanied by a sample classification tool. Extensive studies have been conducted to validate and assess the performance of the workflow. The simulation results illustrate the high accuracy and robustness of the developed flow solvers. Their superior efficiency in computing permeability of tight rocks is demonstrated in comparison with the state-of-the-art commercial solver for DRP.

Additionally, the Navier-Stokes solver for binary images from tight sandstones is discussed.

RHEINLAND-PFÄLZISCHE TECHNISCHE UNIVERSITÄT KAISERSLAUTERN-LANDAU

Zusammenfassung

Fachbereich Mathematik

Development and numerical study of efficient solvers for single-phase steady flows in tight porous media

by Vladislav PIMANOV

Einphasenströmungen sind von großem Interesse in der digitalen Gesteinsphysik (englisch DRP), hauptsächlich für die Berechnung der Permeabilität von Gesteinsproben. Trotz der aktiven Entwicklung von Algorithmen und Software für DRP, bleiben Simulationen auf der Porenskala für dichte Reservoirs – in der Regel charakterisiert durch eine niedrige Mehrskalen-Porosität und niedrige Permeabilität - herausfordernd. Der Begriff "Mehrskalen-Porosität" beschreibt hierbei das Phänomen, dass trotz hoher Bildauflösung der Probe zusätzlich zu reinen Fluid Regionen auch poröse Regionen im Bild vorhanden sind. Aufgrund der Komplexität der Porenraumgeometrien, physikalischer Prozesse, die auf mehreren Längenskalen ablaufen, starker Variation in Koeffizienten und der enormen Größe des Berechnungsbereiches können existierende numerische Algorithmen nicht immer zufriedenstellende Ergebnisse liefern.
Selbst wenn der Porenraum voll aufgelöst wird, stagnieren konventionelle Stokes-Löser, die für die Berechnung der Permeabilität für höhere Porosität entwickelt wurden, in manchen Fällen bei dichten Gesteinsproben. Für entsprechende Diskretisierungen der Stokes-Gleichungen ist wohlbekannt, dass das Schur-Komplement Spektral äquivalent zur Identitätsmatrix ist. Weiterhin wird für simple Geometrien häufig beobachtet, dass ein Großteil der Eigenwerte den Wert eins annimmt. Diese Feststellungen formen die Basis des bekannten Uzawa-Algorithmus. In komplexen Geometrien kann jedoch ein signifikanter Anteil der Eigenwerte des Schur-Komplements Nichteinheitswerte annehmen, wodurch die Matrix schlecht konditioniert ist. Dadurch wird der etablierte Uzawa-Preconditioner ineffizient. Um dieses Verhalten zu erklären, wird in dieser Arbeit eine Spektralanalyse für die Druck-Schur-Komplement-Formulierung mit Finite-Differenzen-Diskretisierung der Stokes-Gleichungen auf versetzten Gittern durchgeführt. Zunächst wird eine Vermutung gerechtfertigt, wonach die Nichteinheits-Eigenwerte des Schur-Komplements durch no-slip Randbedingungen hervorgerufen werden. Anschließend wird demonstriert, dass die Konditionszahl mit steigendem Verhältnis von Oberfläche zu Volumen der Fluiddomäne steigt. Als Alternative zum Uzawa-Preconditioner wird der diffusive SIMPLE-Preconditioner für Geometrien mit hohem Oberflächen-zu-Volumen-Verhältnis vorgeschlagen. Es wird gezeigt, dass dieser für solche Geometrien deutlich effizienter und robuster ist. Weiterhin wird gezeigt, dass die Berechnung der Permeabilität dichter, poröser Medien durch Einsatz des SIMPLE-Preconditioners mit höherer Genauigkeit durchgeführt werden kann.

Einen zentralen Punkt dieser Arbeit stellt die Entwicklung eines zuverlässigen Workflows mit robusten und effizienten Stokes-Brinkman sowie Darcy Lösern dar, die spezifisch für Proben mit niedriger Porosität designt wurden. Zusätzlich beinhaltet der Workflow ein Klassifizierungstool für Proben. Ausgiebige Studien zur Validierung und Ermittlung der Performanz des Workflows werden durchgeführt. Die Simulationsergebnisse verdeutlichen die hohe Genauigkeit und Robustheit der entwickelten Fluid Löser. Ihre überlegende Effizienz für die Berechnung der Permeabilität dichter Gesteine wird durch Vergleich mit dem state-of-the-art kommerziellen Löser für DRP deutlich gemacht.
Zusätzlich wird eine Erweiterung zu einem Navier-Stokes Löser für Binärbilder von dichtem Sandstein diskutiert.

Acknowledgements

I want to express my gratitude to my supervisors: Prof. Oleg Iliev, who supported me and guided me to the end of my doctoral studies, and Prof. Ivan Oseledets, who initiated my scientific journey. I am also thankful to my colleagues and friends from the SMS department of Fraunhofer ITWM for their warmth and support. I also want to thank my former colleagues from Skoltech and my various mentors, including Prof. Sergey Utyuzhnikov, Prof. Ekaterina Muravleva, Prof. Denis Orlov, Prof. Andrzej Cichocki, Prof. Maxim Rakhuba, Dr. Pavel Toktaliev, Dr. Aleksandr Katrutsa, and others. I want to express my gratitude to my entire family, especially to my parents for their unconditional support, my brothers for paving the way, and my fiancée for being there during the most challenging moments.

Contents

List of Abbreviations

PDE	**P**artial **D**ifferencial **E**equation
BVP	**B**oundary **V**alue **P**roblem
DRP	**D**igital **R**ock **P**hysics
CFD	**C**omputational **F**luid **D**ynamics
FD	**F**inite **D**difference
MAC	**M**arker and **C**ell
BC	**B**oundary **C**ondition
DHHD	**D**iscrete **H**elmholtz-**H**odge **D**ecomposition
AMG	**A**lgebraic **M**ulti**G**rid
FEM	**F**inite **E**lement **M**ethod
SEM	**S**canning **E**lectron **M**icroscope
CT	**C**omputer **T**omography
CG	**C**onjugate **G**radient
MINRES	**Min**imal **Res**idual
GMRES	**G**eneralized **M**inimal **Res**idual
PCG	**P**reconditioned **C**onjugate **G**radient
SIMPLE	**S**emi-**I**mplicit **M**ethod for **P**ressure **L**inked **E**quations
SCoPeS	**S**koltech **Co**mputing **Pe**rmeability **S**olver
SCoPeS-S	SCoPeS-**S**tokes
SCoPeS-D	SCoPeS-**D**arcy
SCoPeS-SB	SCoPeS-**S**tokes-**B**rinkman
SCoPeS-NSE	SCoPeS-**N**avier-**S**tokes **E**quations
VIPO	**V**elocity-**I**nlet-**P**ressure-**O**utlet
PIPO	**P**ressure-**I**nlet-**P**ressure-**O**utlet

Chapter 1

Introduction

1.1 Tight reservoirs: industrial motivation

Leading oil companies in the global energy sector are expected to continue their significant efforts to achieve consistent and commercially viable production from tight and ultra-tight hydrocarbon resources. These resources often have initial reserves that are substantially larger than those of conventional resources. Despite all technological achievements, the development of unconventional resources might yet be a risky investment if it is not supported well with detailed and accurate reservoir studies [1]. Given the demands of reservoir studies and the associated challenges, the development of techniques, methods, and instruments for the thorough evaluation of unconventional hydrocarbon resources has emerged as a critical research area [2]. Regardless of many attempts that have been made to modify the conventional methods of core analysis suitable to be applied for the evaluation of unconventional hydrocarbon resources [3], they are still time-consuming, expensive, and inaccurate [4]. It is mainly due to the existence of sub-micron pores and throats that have strong effects on the storage and flow of tight and ultra-tight oil and gas resources [5, 6].
To fit the risk attitude of the decision-makers, applying modern methods of core analysis like Digital Rock Physics (DRP) has mainly been focused on over the last decade [7, 8]. Besides the fact that DRP is an efficient cost control and risk management method, it allows the researchers to perform multiple numerical experiments on exactly the same sample and implement various analyses simultaneously [9, 10].
Single-phase flows have gained significant attention in Digital Rock Physics area, particularly in the context of computing the permeability of rock samples. Numerous commercial [11, 12, 13] and academic [14, 15, 16, 17] flow solvers have been developed to address this area of research. These solvers are actively employed in solving a wide range of scientific, environmental, and industrial problems, as well as in benchmark studies [18, 19]. Despite the continuous advancement of algorithms and software in DRP, simulating pore-scale processes in tight reservoirs, characterized by low, multiscale porosity, remains a formidable challenge. Even in cases where the rocks do not exhibit multiscale porosity, conventional Stokes solvers designed for computing permeability at higher porosities may struggle to converge when applied to images of tight rocks. Moreover, the motivation for solving the Stokes problem for samples with low porosity arises when considering specific geothermal applications, certain ceramic filters, catalytic filters, and other man-made porous materials. As a result, the demand for advanced, customized algorithms in this area has significantly increased in recent years. Though less frequent, there is also a need to solve the Navier-Stokes equations at the pore scale in the case of faster flows.

1.2 Digital Rock Physics

Digital Rock Physics combines microtomographic imaging with advanced numerical simulations of effective material properties. A key application of DRP lies in the oil and gas industry, where understanding the characteristics of reservoir rocks is practically important. Specifically, in this study we focus on determining the flow permeability of porous rock samples.
The general DRP pipeline is as follows:

1. **Image Acquisition:** Rock sample is imaged using high-resolution 3D techniques like X-ray micro-computed tomography (μCT). The image captures the rock's pore structure and grain configuration in detail.

2. **Image Processing:** The acquired image undergo processing to distinguish between solid, void (pore), and possibly gray (unresolved porosity) regions. This stage involves noise reduction, segmentation, and filtering to ensure a clear differentiation between the rock's different components.

3. **Digital Representation**: The processed image is converted into a digital model, typically represented by a 3D array or grid. This digital model, often called as "digital rock" or "digital twin", represents the rock's microstructure and serves as an input data for subsequent flow simulations.

4. **Fluid Flow Simulations:** Simulations are conducted on the digital rock to estimate its absolute permeability. This encompasses solving the (Navier-)Stokes or the Stokes-Brinkman equations, depending on the flow regime, complexity of the pore network, and appearance of unresolved porosity regions.

5. **Validation:** The computed permeability from simulations is validated by comparing it with laboratory measurements conducted on the real rock sample.

For this thesis, the focus is on the stages 3 and 4, with the stages 1 and 2 being briefly covered in Chapter 3. Validation through a physical experiment is not within the scope of this work. Instead, validation is achieved through comparisons with commercial flow solvers, which have been validated through experiments.
The primary task considered in the study is to calculate the absolute permeability of tight porous media in the context of slow Stokes and Stokes-Brinkman flows. The absolute permeability is an effective property of the porous material. A key to computing permeability is the Darcy's law, which can be written as follows:

$$\langle u \rangle = -\frac{k^{eff}}{\mu}(\nabla p - f), \tag{1.1}$$

where k^{eff} denotes absolute permeability, μ is the viscosity, $\langle u \rangle$ is the Darcy velocity, ∇p is the pressure gradient, and f is a possible density force.
According to the Darcy's law (1.1), performing numerical experiment to calculate permeability typically consists in determining either the mean velocity given a pressure gradient, or determining the pressure gradient given a mean velocity. In both cases, it is required to study how flow propagates through porous samples at pore-scale (micro-scale) resolution.
The following specifics of our problem should be outlined in this context:

- In our industrial application, the flow geometry is known approximately as it is captured by Computer Tomography (CT). In this case, solving the underlying

equations with high accuracy is not necessary. Our primary objective is computation of the absolute permeability of porous samples (see Section 2.2.3 for details), and typically one percent accuracy in computing the objective functional is sufficient.

- Methods that are efficient for solving PDEs in simple domains like squares may not perform well when applied to our complex domains.
- Due to limited scanning accuracy, grid convergence is usually not studied when solving PDEs in geometries derived from CT images. The goal is to efficiently solve the problem in a geometry with a fixed resolution that coincides with the voxelized CT image.

1.3 Structure of the thesis

We distinguish two classes of images as they significantly influence the underlying physical scenario and the equations which govern the flow:

1. Binary images (only solid and void).
2. Multiclass images (grayscale).

The grayscale images can be seen as a continuum relaxation between the solid and void phases of binary images. Namely, as an input data we consider 3-dimensional arrays where each element, called voxel, has a prescribed porosity value between 0% (completely solid) and 100% (completely void). Such grayscale images provide a more comprehensive insight than binary segmentations. During our study, we focus on the Stokes equations in the case of binary images and on the Stokes-Brinkman equations in the case of multiclass images. The primary objectives of this work are twofold:

1. To develop and theoretically analyze an efficient flow solver for pore-scale single-phase Stokes flows in binary images from tight sandstones.
2. To create a reliable workflow for computing absolute permeability of multiclass images derived from tight sandstones.

The structure of the thesis is as follows:

- In Chapter 2, we consider the Stokes problem for binary images from tight porous media, with the contributions detailed in Section 2.1.4.
- In Chapter 3, we explore the Stokes-Brinkman equations for multiclass images and investigate their approximation with the Darcy equation. The presented workflow, including the image classifier, is described in Section 3.3.
- Generalization from the Stokes to the Navier-Stokes equations in the case of binary images is covered in Chapter 4, with the main contributions listed in Section 4.1.
- Summary is provided in Chapter 5.

Chapter 2

Stokes equations in tight porous media

2.1 Problem statement

Let's consider a bounded, open domain $\Omega^f \subset \mathbb{R}^d$, where $d = 2, 3$. This domain represents the region in which the fluid flow propagates. Within Ω^f, we consider the steady-state Stokes equations, given as follows:

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \mathbf{f} \text{ in } \Omega^f, \\ -\nabla \cdot \mathbf{u} &= 0 \text{ in } \Omega^f, \end{aligned} \tag{2.1}$$

where $\mathbf{u}$ is the fluid velocity, p is the fluid pressure, and $\mathbf{f}$ is the volumetric force driving the flow. The Boundary Value Problem (BPV) for the Stokes equations (2.1) is obtained by additionally imposing boundary conditions on the boundary $\partial\Omega^f = \Gamma_0 \cap \Gamma_{\text{ext}}$, where Γ_{ext} denotes the exterior boundary which is discussed later in Section 2.2.2. We consider the no-slip plus no-penetration boundary conditions on Γ_0, which is simply the zero Dirichlet condition imposed on the velocity:

$$\mathbf{u} = \mathbf{0} \text{ on } \Gamma_0. \tag{2.2}$$

The discretization of the BVP (2.1),(2.2) leads to a block system of linear equations of the following form:

$$\mathbb{A} \begin{bmatrix} \mathbf{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \mathbf{f}_h \\ 0 \end{bmatrix}, \quad \mathbb{A} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & 0 \end{bmatrix}. \tag{2.3}$$

Here the vectors $\mathbf{u}_h$, p_h, $\mathbf{f}_h$ denote the discretized velocity, pressure, force, respectively, and the subscript h denotes the mesh size parameter. The matrices $\mathbf{A}$ and $\mathbf{B}$ are discrete counterparts of the negative velocity Laplacian operator and the negative divergence operator. The matrix $\mathbf{B}^T$ denotes the discrete pressure gradient operator, which is adjoint to the negative divergence operator under proper discretization. Importantly, the no-slip boundary condition (2.2) is incorporated into the matrix $\mathbb{A}$.

2.1.1 Pressure Schur complement approach

A variety of methods for solving the system (2.3) can be roughly divided into two categories: coupled methods, in which the system is solved in the full form, and reduced methods, in which one of the variables is first eliminated from the equations and then restored. In what follows, we consider the Pressure Schur Complement formulation, according to the terminology from [20], which reduces the coupled system (2.3) to

an equivalent system for the pressure:

$$Sp_h = g_h, \tag{2.4}$$

where S is the Schur complement of the matrix $\mathbb{A}$ and g_h is the right-hand-side in the reduced equation, which are defined as follows:

$$S = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T, \quad g_h = \mathbf{B}\mathbf{A}^{-1}\mathbf{f}_h. \tag{2.5}$$

Once the pressure is computed, the velocity can be recovered by solving the following system for the velocity:

$$\mathbf{A}\mathbf{u}_h = \mathbf{f}_h - \mathbf{B}^T p_h. \tag{2.6}$$

In the present Chapter, we examine the impact of the no-slip boundary condition (2.2) on the conditioning of the Schur complement system (2.4), particularly when the surface-to-volume ratio of the flow domain Ω^f is high. The surface-to-volume ratio refers to the ratio between the surface area of the no-slip boundary Γ_0 and the volume of the flow domain Ω^f. Note, high surface-to-volume ratio is a distinctive feature for one of our critical applications — namely, computation of permeability of tight porous media. Importantly, our findings are not confined to this specific application.

2.1.2 Motivation: ill-conditioning of the Schur complement matrix

Specific samples from our practical applications for $d = 3$ are depicted in Fig. 2.1. In this context, Ω^f represents the void space of a porous medium (colored in black), and Γ_0 denotes the boundary between Ω^f and the solid domain Ω^s, colored in grey. Along with the no-slip boundary Γ_0, the computational domain here includes the outer faces of the cube, which constitute the exterior boundary Γ_{ext}. However, we apply the periodic boundary conditions on these faces, which do not impact the spectrum of the matrix S, though are also incorporated into the matrix $\mathbb{A}$ from (2.3). The complete formulation of the flow experiment considered in our study is provided in the subsequent Section 2.2, which includes a rigorous definition of the domains Ω^f, Ω^s and of the periodic boundary conditions applied to the exterior boundary Γ_{ext}. It is worth mentioning , that prescribing the Dirichlet boundary condition on the entire boundary Γ_0, along with the exterior periodic conditions, results in a special class of flow problems. For such flows, the pressure is determined only up to a constant nullspace, which requires special attention, as outlined later in Section 2.3.
If the Stokes equations (2.1) are properly discretized such that the discrete operators preserve important properties of the continuous ones, then, up to a constant in the nullspace, the Schur complement matrix S is known to be spectrally equivalent to the identity operator acting on the discrete pressure space. For example, in the context of the Finite Element Method, the equivalence to the pressure mass matrix takes place with the right choice of the LBB-stable elements (see, e.g., [22]). The spectral equivalence to the identity means that the minimal nonzero eigenvalue of the Schur complement matrix S, as well as its effective condition number, are bounded from below and above under the mesh refinement process, i.e. when $h \to 0$. Moreover, in practice it is often observed that most of the eigenvalues of S are equal to one (in the case of simple geometries). These facts form the basis for the famous Uzawa and Uzawa-like algorithms [23, 24, 25, 20, 26, 27], which are considered classical algorithms for solving the steady Stokes problem (2.1) and rely essentially on the

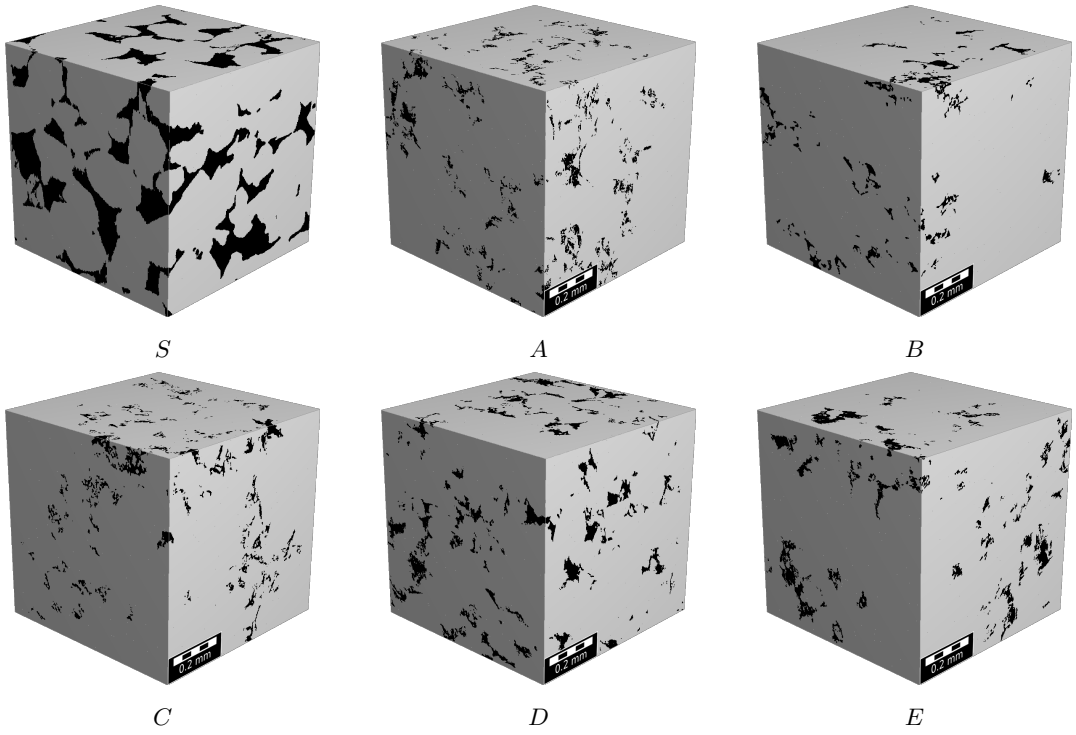


FIGURE 2.1: Binary images of porous media samples. The black color represents the void space Ω^f, where the fluid propagates, and the grey color represents the impermeable solid Ω^s. Samples A to E are ultra-tight images from [21], and S is a moderate porosity image of Berea sandstone. For further details about the samples, see Table 2.1.

preconditioning of S with the identity. Despite recent progress in developing efficient iterative methods for solving the Stokes problem, the Uzawa algorithm remains popular in science and engineering, especially when accelerated by Krylov subspace methods. However, in complex geometries, the Schur complement matrix can become severely ill-conditioned, having a significant portion of non-unit eigenvalues. This makes the established Uzawa preconditioner inefficient.

In fact, when solving the Stokes problem, the spectrum of S depends on the boundary conditions imposed. Based on our study, we conjecture that this is due to the no-slip boundary condition (2.2), the spectrum of the Schur complement matrix contains eigenvalues which are not equal to one. Thus, when one solves problems with a small surface-to-volume ratio, which is the case most commonly considered in papers analyzing iterative solvers for Stokes problems, only a small part of eigenvalues of the Schur complement are not equal to one. This justifies using a diagonal matrix or even the identity matrix as a preconditioner.

Numerous computational studies demonstrate the efficiency of the Uzawa algorithms in the case of simple geometries. A number of reviews and theoretical studies are dedicated to this subject advancing the knowledge in the area, see, e.g. [23, 28, 29]. Even a superlinear convergence of the Krylov-Uzawa algorithm can be established for general smooth geometries, see [27]. Unfortunately, for our practical application, we found that the Uzawa algorithm is not efficient. We observe that the Schur complement matrix S can become severely ill-conditioned, when computing flows in complex geometries like those representing the pore-space of tight rock sample, certain membranes, etc.. We conjecture that this issue arises due to a high surface-to-volume

ratio. In particular, we demonstrate this issue in Section 2.5.1 for specific rock samples from Fig.2.1, for which the condition number of S is greater than 10^5. Therefore, development of customized methods is required for solving Stokes problems in such geometries.

2.1.3 Preconditioners under investigation

The SIMPLE and the Uzawa preconditioners for the Schur complement matrix, denoted $\hat{S}_{\text{simple}}$ and $\hat{S}_{\text{uzawa}}$, are investigated in this Chapter. They can be written as follows:

$$\hat{S}_{\text{simple}} = \mathbf{B}\hat{\mathbf{A}}^{-1}_{\text{simple}}\mathbf{B}^T, \quad \hat{\mathbf{A}}_{\text{simple}} = \text{diag}(\mathbf{A}), \;\; \hat{S}_{\text{uzawa}} = I. \tag{2.7}$$

In fact, the preconditioner $\hat{S}_{\text{simple}}$ is widely-known in the CFD community since the same approximation is used in the SIMPLE iterative method (Semi-Implicit Method for Pressure Linked Equations), which is one of the classical methods for solving the stationary Navier-Stokes equations [30, 31, 32, 33, 34, 35], but not for the Stokes equations.
Recently, it was demonstrated in [36] that adding discrete diffusion to the established Uzawa preconditioner significantly reduces the number of iterations in the case of channel-dominated domains. Optimal weights for the diffusion and the identity operators in the proposed preconditioner are discussed there. Broad computational experiments are performed and discussed, varying geometries, weights, finite elements spaces. It is stated that the convergence strongly depends on the geometry complexity, with the channel width and aspect ratio considered as important parameters. In our case, unlike [36], we omit the identity in the considered SIMPLE preconditioner. Earlier in [37], we demonstrated that the diffusion-like SIMPLE preconditioner in the case of complex geometries from tight porous media performs very well. In the present article, we investigate several important aspects which were not discussed in [36] and [37]. We show that the condition number of the Schur complement matrix depends on the surface-to-volume ratio, since the number of its non-unit eigenvalues is related to the surface on which no-slip boundary conditions are prescribed. Furthermore, comparing the performance of the two preconditioners, we emphasize also the fact that the SIMPLE preconditioner allows for more accurate and robust computation of the permeability. Some other differences could be mentioned. Finite element discretization is used in [36], and iterative method (MINRES) with block diagonal preconditioner is applied for the coupled system. We use staggered finite-difference discretization of the Stokes equations, and apply the Conjugate Gradient method for the Schur complement system. As mentioned above, we identify certain geometric characteristics of the domain which are indicators for the performance of the respective Stokes solvers. Only 2D synthetic problems are considered in [36], while we perform also simulations on 3D samples from real tight reservoirs.

2.1.4 Chapter outline and contributions

The remainder of the Chapter is organized as follows. Section 2.2 is dedicated to the formulation of the flow experiment. The description of the considered iterative methods, namely the CG-Uzawa and CG-SIMPLE algorithms, is provided in Section 2.3. Validation of the developed solvers is presented in Section 2.4. In Section 2.5, we present and discuss the results of the computational experiments. These can be summarized as follows.

1. In Section 2.5.1, we investigate 3D binary samples from Fig. 2.1. Specifically, the samples with high surface-to-volume ratio coming from tight reservoirs. For these 3D samples, we compare the performance of the CG-SIMPLE and CG-Uzawa iterative methods and confirm by our numerical experiments that:
   - The preconditioner $\hat{S}_{\mathrm{simple}}$ provides orders of magnitude lower condition numbers than $\hat{S}_{\mathrm{uzawa}}$ hence ensuring robust and fast convergence of the CG-SIMPLE method while the CG-Uzawa method tends to stagnate;
   - Furthermore, we demonstrate that the CG-SIMPLE provides more accurate practical computation of the absolute permeability.
2. We explain this behavior in Section 2.5.2, by performing a systematic study using synthesized 2D geometries - random packings of squares. For the considered synthetic geometries, we numerically demonstrate that:
   - The condition number $\mathrm{cond}(S)$ of the Schur complement matrix increases linearly with increasing the surface-to-volume ratio;
   - The condition number $\mathrm{cond}(\hat{S}^{-1}_{\mathrm{simple}}S)$ of the Schur complement matrix preconditioned with the SIMPLE preconditioner decreases super-linearly with increasing the surface-to-volume ratio.
3. Additionally, for the considered synthetic geometries, we compute the full spectrum of the Schur complement matrix and observe that the number of its non-unit eigenvalues is determined by the number of boundary nodes where the Dirichlet boundary condition on the tangential velocity is imposed, and by the connectivity of the flow domain Ω^f.

Finally, in Section 2.6, in the case of simplest geometry we give theoretical justification of the results numerically observed in Section 2.5 .

2.2 Flow experiment on CT images

2.2.1 Representing pore-space geometries: binary images

In this subsection, we describe how the fluid region of a rock sample or a sample of other porous material, denoted as Ω^f in (2.1), is represented using voxel grids. Additionally, we formulate the periodic boundary conditions typical for the DRP flow experiment and provide the formula for computing absolute permeability.
In 3D, we represent CT images by the cubic domain $\overline{\Omega}_h = [0, L]^3$, where $L[m]$ is the physical size of a sample. The image is decomposed into n^3 voxels, where n is the number of voxels (image resolution) in each dimension:

$$\overline{\Omega}_h = \bigcup_{(i,j,k)\in \mathbb{I}^{\mathrm{n}}} \omega_{(i,j,k)}, \tag{2.8}$$

where $\mathbb{I}^{\mathrm{n}} = \{(i,j,k) : i,j,k \in 1,\dots,\mathrm{n}\}$ denotes a three-dimensional index set, and the voxels $\omega_{(i,j,k)}$ are defined as cubic regions of the length $h = L/\mathrm{n}$:

$$\omega_{(i,j,k)} = [(i-1)h; ih] \times [(j-1)h; jh] \times [(k-1)h; kh].$$

The entire image Ω_h is subdivided into two parts:

$$\overline{\Omega}_h = \overline{\Omega}^f_h \cup \overline{\Omega}^s_h, \tag{2.9}$$

that corresponds to a disjoint decomposition of the index set $\mathbb{I}^{\mathrm{n}}$:

$$\mathbb{I}^{\mathrm{n}} = \mathbb{I}^{\mathrm{n}}_f \sqcup \mathbb{I}^{\mathrm{n}}_s, \tag{2.10}$$

such that:

$$\overline{\Omega}^s_h = \bigcup_{(i,j,k)\in\mathbb{I}^{\mathrm{n}}_s} \omega_{(i,j,k)}, \quad \overline{\Omega}^f_h = \bigcup_{(i,j,k)\in\mathbb{I}^{\mathrm{n}}_f} \omega_{(i,j,k)}. \tag{2.11}$$

It should be noted, that arbitrary complex porous geometries can be approximated in such a way. Example of voxel-based geometries for the case $d = 3$ and $d = 2$ are shown in Figs. 2.1 and 2.7, respectively. In the solid region, denoted Ω^s_h, the fluid does not propagate since impermeable solid is considered here. So, the Stokes problem (2.1) is formulated in the fluid region Ω^f_h, and the no-slip boundary Γ_0 from (2.2) is given as follows:

$$\Gamma_0 = \overline{\Omega}^f_h \cap \overline{\Omega}^s_h. \tag{2.12}$$

It should be emphazized, that the voxel-based geometry as defined in (2.8) serves as the computation grid to discretize the Stokes problem in Ω^f_h. The subscript h here reflects the fact that the computational domain is inherently discretized since it comes from a binary CT image. In our work, we utilize the classical fully-staggered finite difference method, also known as the MAC scheme. Namely, the pressure p^h is discretized at the centers of the voxels $\omega_{(i,j,k)}$, while the velocity $\mathbf{u}_h$ is discretized on the voxel faces. Discretization in the case of simplest geometry is described in Section 2.6. Discretization for general voxel geometries used in our practical computations is not covered in the thesis and can be found, for example, in [38]. For a detailed description, see, for instance, [39, 38, 40].

2.2.2 Exterior periodic boundary conditions

As it was previously mentioned, apart from the interior boundary Γ_0 (2.12) between solid and fluid regions, the entire boundary of Ω^f_h comprises an additional exterior part Γ_{ext}. According to the homogenization theory, periodic boundary conditions are typically imposed on the exterior faces of the domain (see, e.g., [41]). As it was previously mentioned, they do not influence the spectrum of the matrix S. Considering the right-handed Cartesian coordinate system shown in Fig. 2.2, we define six boundaries, which are the outer faces of the entire cube domain Ω^h from (2.8):

$$\begin{aligned} &\Gamma_{\mathrm{x}=0}, \quad \Gamma_{\mathrm{y}=0}, \quad \Gamma_{\mathrm{z}=0}, \\ &\Gamma_{\mathrm{x}=L}, \quad \Gamma_{\mathrm{y}=L}, \quad \Gamma_{\mathrm{z}=L}. \end{aligned} \tag{2.13}$$

Then, the exterior boundary Γ_{ext} is given as follows:

$$\Gamma_{\mathrm{ext}} = (\Gamma_{\mathrm{x}=0} \cup \Gamma_{\mathrm{x}=L} \cup \Gamma_{\mathrm{y}=0} \cup \Gamma_{\mathrm{y}=L} \cup \Gamma_{\mathrm{z}=0} \cup \Gamma_{\mathrm{z}=L}) \cap \overline{\Omega}^f_h. \tag{2.14}$$

In the DRP flow experiment, typically one direction is selected as the *flow direction* and two other are selected as the *tangential directions*. Without loss of generality, we assume that z direction is fixed as the flow direction, and x, y directions as the tangential ones. In the tangential directions, we impose the periodic boundary condition for both the velocity and pressure. For the x direction, we have:

$$\mathbf{u}|_{\Gamma_{\mathrm{x}=0}} = \mathbf{u}|_{\Gamma_{\mathrm{x}=L}}, \quad p|_{\Gamma_{\mathrm{x}=0}} = p|_{\Gamma_{\mathrm{x}=L}} \text{ on } \Gamma_{\mathrm{ext}},$$

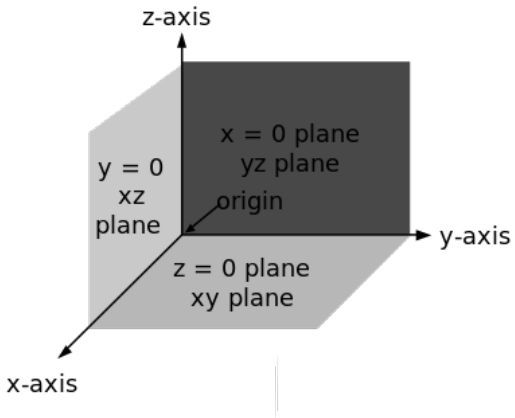


FIGURE 2.2: The right-handed Cartesian coordinate system in 3D.

and, similarly, the periodic boundary condition is imposed for the y direction. In z direction, additionally a pressure difference should be accounted which drives the flow:

$$\mathbf{u}|_{\Gamma_{\mathrm{z}=0}} = \mathbf{u}|_{\Gamma_{\mathrm{z}=L}}, \quad p|_{\Gamma_{\mathrm{z}=0}} = p|_{\Gamma_{\mathrm{z}=L}} + dp \text{ on } \Gamma_{\mathrm{ext}}, \tag{2.15}$$

where dp is a given pressure jump. In the case of periodic boundary conditions, the geometry is assumed to be periodic too. Namely, periodicity of the flow domain Ω_f^h in all three directions can be expressed as follows:

$$(i,j,k) \in \mathbb{I}_f^{\mathrm{n}} \implies (\mathrm{n}-i,j,k),(i,\mathrm{n}-j,k),(i,j,\mathrm{n}-k) \in \mathbb{I}_f^{\mathrm{n}}. \tag{2.16}$$

In the case when the geometry $\mathbb{I}_f^{\mathrm{n}}$ is not periodic, it can be periodized by symmetric reflection. It is important to emphasize, that the flow domain Ω_h^f must be connected to ensure the well-posedness of the formulated BVP (2.1). Additionally, the constraint $\int_{\Omega_h^f} p = 0$ on the pressure is typically assumed for uniqueness of the solution.
Note, other than periodic boundary conditions for the DRP flow experiment are discussed in Appendix 2.A.

2.2.3 Computing permeability

The computation of the permeability tensor according to the homogenization theory can be found, for example in [41]. However, for brevity we consider an equivalent approach most often used in engineering literature [42]. For the selected flow direction, z, the respective component of the permeability tensor, denoted $k_{\mathrm{zz}}[m^2]$, is determined according to the Darcy's law, which is given as follows:

$$\frac{Q}{A} = -\frac{k_{\mathrm{zz}}}{\mu} \cdot \frac{dp}{L}, \tag{2.17}$$

where $\mu[Pa \cdot s]$ is the viscosity, $Q[m^3/s]$ is the volumetric flow rate, $A[m^2]$ is the cross-sectional area of the sample, $L[m]$ is the thickness of the sample in the flow direction where the pressure drop $dp[Pa]$ is applied. Note, we consider $\mu = 1[Pa \cdot s]$ since it does not influence the permeability. The flow over the unit area is computed as follows:

$$\frac{Q}{A} \approx \langle u_{\mathrm{z}} \rangle. \tag{2.18}$$

Here, the Darcy's velocity $\langle u_{\rm z} \rangle$ is calculated by averaging the respective velocity component over the entire volume of the porous sample [43]:

$$\langle u_{\rm z} \rangle = \frac{1}{|\Omega_h|} \int_{\Omega_h^f} u_{\rm z}. \tag{2.19}$$

In practice, for reasons of numerical stability in finite precision arithmetic, we compute the non-dimensional permeability, denoted $\hat{k}_{\rm zz}$, by taking $L = 1[m]$ in (2.8). The physical permeability can then be computed by scaling as follows:

$$k_{\rm zz} = \hat{k}_{\rm zz} L^2.$$

When the periodic boundary conditions (2.15) are imposed in the flow direction $\rm z$, the problem is actually solved for the periodic part of the pressure, while its gradient $\nabla p = \frac{dp}{L}$ goes to the volumetric force $\mathbf{f}$ in the equation (2.1). For Ω_h having the unit length $L = 1[m]$, the unit pressure drop $dp = 1[Pa]$ corresponds to the unit volume force $\mathbf{f} = (0, 0, 1)^T$ applied in the flow direction [44]. So, the permeability $k_{\rm zz}$ equals the Darcy (averaged) velocity $\langle u_{\rm z} \rangle$ in this case. Note, that in order to compute permeabilities $k_{\rm xx}$ and $k_{\rm yy}$ for the $\rm x$ and $\rm y$ flow directions, two additional computations for $\mathbf{f} = (1, 0, 0)^T$ and $\mathbf{f} = (0, 1, 0)^T$ are required. We omit here the discussion on computing the off diagonal elements of the permeability tensor.

2.3 Iterative methods and preconditioning

2.3.1 Outer iterations: CG-SIMPLE and CG-Uzawa algorithms

Iterative methods for solving the Stokes problem (2.1) can generally be classified into two categories:

- Methods in which iterations are performed for the coupled system (2.3) in its full form.
- Methods in which iterations are performed for the reduced Schur complement system (2.4).

In both categories, efficient techniques are required to solve systems involving the matrices $\mathbf{A}$ and S. Concerning the velocity Laplacian matrix $\mathbf{A}$, its inverse can be effectively applied through multigrid methods, such as the Algebraic Multigrid method (AMG) [45], which can handle intricate geometries. In the present work, our focus is on the Schur complement matrix S, so we consider the reduced formulation (2.4). Note, the relation between the reduced methods under investigation and their coupled analogues is discussed in Section 2.B in the Appendix.
Since the matrix S is positive semi-definite, following Axelsson (Section 3.1 in [27]), we employ the Conjugate Gradient (CG) method as a Krylov subspace accelerator. However, in contrast to [27], we use the preconditioned version of CG (as described in, e.g., [46], Section 9.2). The specific form of the Preconditioned Conjugate Gradient (PCG) method considered in our study is formulated in Algorithm 1. By **CG-Uzawa** and **CG-SIMPLE**, we denote the Algorithm 1 for the preconditioners $\hat{S} = \hat{S}_{\rm uzawa}$ and $\hat{S} = \hat{S}_{\rm simple}$ defined in (2.7), respectively. Obviously, when the identity is used as a preconditioner, the preconditioned and unpreconditioned CG coincide. It should be noted, that in the Stokes case, the spectrum of $\hat{S}_{\rm simple}$ is qualitatively different from the spectrum of S. Namely, the matrix $\hat{S}_{\rm simple}$ behaves essentially as the pressure Laplacian matrix $\mathbf{B}\mathbf{B}^T$, so its condition number increases

Algorithm 1 Preconditioned CG, adapted from Axelsson [27]

Require: tolerance ε_S, initial guess p_h^0
Ensure: Approximate solution p_h for the system (2.4)
1: Compute the initial residual $r_h^0 = Sp_h^0 - g_h$
2: Solve $\hat{S}z_h^0 = r_h^0$ for z_h^0
3: Set $d_h^0 = z_h^0$, $k = 0$
4: **while** not converged **do**
5: Apply $Sd_h^k = q_h^k$ for q_h^k
6: $\alpha_k = \dfrac{(r_h^k)^T z_h^k}{(d_h^k)^T q_h^k}$
7: $p_h^{k+1} = p_h^k + \alpha_k d_h^k$
8: $r_h^{k+1} = r_h^k + \alpha_k q_h^k$
9: Solve $\hat{S}z_h^{k+1} = r_h^{k+1}$ for z_h^{k+1}
10: If $\|z_h^{k+1}\|/\|z_h^0\| < \varepsilon_S$, exit loop
11: $\beta_k = \dfrac{(r_h^{k+1})^T z_h^{k+1}}{(r_h^k)^T z_h^k}$
12: $d_h^{k+1} = z_h^{k+1} + \beta_k d_h^k$, $k = k + 1$
13: **end while**
14: Return p_h^{k+1} as the approximate solution

quadratically as the grid resolution decreases. However, such spectral behavior turns out to be justified in the case of tight geometries in the presence of narrow channels where the nature of flow is predominantly diffusive.

Remark 2.3.1. *In the original work by Axelsson, the CG-Uzawa algorithm is formulated for the regularized version of the system* (2.3)*. We do not employ any regularization as there is no necessity for it. It is also worth noting, the MAC scheme is stable in LBB sense, see e.g. [47] and references therein.*

Remark 2.3.2. *As mentioned earlier, in the case of periodic boundary conditions* (2.15)*, the matrix S has a constant nullspace. However, the CG method converge to the normal solution as soon as the corresponding system is consistent [48]. Because S is singular, instead of considering its inverse, we have to actually consider its pseudo-inverse. For clarity of notation, we use $\hat{S}^{-1}$ instead of $\hat{S}^{\dagger}$ throughout the text. The term condition number here is used in the sense of effective condition number.*

Remark 2.3.3. *It should be noted, that in the exact arithmetic the Algorithm 1 can be also considered as the regular CG applied for the symmetrically preconditioned Schur complement system [49], which can be written as follows:*

$$\hat{S}^{-\frac{1}{2}} S \hat{S}^{-\frac{1}{2}} \hat{p}_h = \hat{S}^{-\frac{1}{2}} g_h,$$

where:

$$(\hat{S}^{-\frac{1}{2}})^2 = \hat{S}^{-1}, \quad \hat{p}_h = \hat{S}^{\frac{1}{2}} p_h.$$

However, it is not required to compute the Cholesky decomposition $\hat{S} = \hat{S}^{\frac{1}{2}} \hat{S}^{\frac{1}{2}}$ of the preconditioner $\hat{S}$, but only to apply $\hat{S}^{-1}$ in this case. Indeed, the spectrum of a non-symmetric matrix $\hat{S}^{-1}S$ coincides with the spectrum of $\hat{S}^{-\frac{1}{2}} S \hat{S}^{-\frac{1}{2}}$, since S and $\hat{S}$ are both symmetric.

2.3.2 Stopping criteria

According to Algorirhm 1, the preconditioned residual norm is used as the stopping criteria for the outer iterative process, see line 10 there. Specifically, given an input tolerance ε_S, the outer PCG iterations stop as soon as:

$$\|\hat{S}^{-1}r_h^{\#}\|/\|\hat{S}^{-1}r_h^0\| < \varepsilon_S, \tag{2.20}$$

where the superscript $\#$ denotes the final iteration number, and r_h^k denotes the residual on the k^{th} outer iteration, given as follows:

$$r_h^k = Sp_h^k - g_h. \tag{2.21}$$

Using the preconditioned residual norm is considered more natural because the preconditioned CG method minimizes the norm of the preconditioned residuals at each step [46, 50]. However, in our numerical experiments, for the CG-SIMPLE method, we additionally monitor the unpreconditioned residual norm. This helps us to compare with the CG-Uzawa method, where the unpreconditioned norm is monitored. Moreover, when studying synthetic 2D geometries in Section 2.5.2, we use the unpreconditioned residual norm directly as the stopping criteria for consistency of the comparison; Note, similar approach was used in [36].

2.3.3 Inner iterations

On each step of the outer iterations, applying the matrix S requires solution of an auxiliary problem to recover intermediate velocity from the intermediate pressure. We use the PCG method for solving with the velocity Laplacian matrix $\mathbf{A}$, so formally we deal with inexact version of the outer PCG method [25, 24]. Thus, we have a two-level inner-outer iterative process: at each step of the outer PCG iteration for S, inner PCG iterations for $\mathbf{A}$ are performed. In our numerical experiments, we use the preconditioned relative residual norm as the stopping criteria for the inner iterations with the matrix $\mathbf{A}$. Namely, given an input tolerance $\varepsilon_{\mathbf{A}}$, the inner PCG iteration for computing $\mathbf{u}_h = \mathbf{A}^{-1}\mathbf{f}^h$ stops as soon as:

$$\|(\hat{\mathbf{A}})^{-1}r_{\mathbf{A}}^{\#}\|/\|(\hat{\mathbf{A}})^{-1}r_{\mathbf{A}}^0\| < \varepsilon_{\mathbf{A}}, \tag{2.22}$$

where $r_{\mathbf{A}}^k$ denotes the residual on the k^{th} inner iteration, given as follows:

$$r_{\mathbf{A}}^k = \mathbf{A}\mathbf{u}_h^k - \mathbf{f}_h. \tag{2.23}$$

Additionally, at each step of the outer CG-SIMPLE iteration, we have to solve the system with the preconditioner matrix $\hat{S}_{\text{simple}}$. Again, we use the Preconditioned Conjugate Gradient for solving with the preconditioner $\hat{S}$. It should be noted that, as well as the pressure Schur complement matrix S, the pressure Neumann Laplacian $\hat{S}_{\text{simple}}$ has constant in the nullspace, so a special care is required. As for the stopping criteria, we use the preconditioned relative residual norm for inner iterations with the matrix $\hat{S}_{\text{simple}}$. For example, given an input tolerance $\varepsilon_{\hat{S}}$, the inner CG iteration for computing $p_h = (\hat{S}_{\text{simple}})^{-1}g^h$ stops as soon as:

$$\|(\hat{\hat{S}}_{\text{simple}})^{-1}r_{\hat{S}}^{\#}\|/\|(\hat{\hat{S}}_{\text{simple}})^{-1}r_{\hat{S}}^0\| < \varepsilon_{\hat{S}}, \tag{2.24}$$

where $r^k_{\hat{S}}$ denotes the residual on the k^{th} inner iteration, given as follows:

$$r^k_{\hat{S}} = (\hat{S}_{\text{simple}})p^k_h - g_h. \tag{2.25}$$

For building preconditioners $\hat{\mathbf{A}}$ and $\hat{\hat{S}}_{\text{simple}}$ for the Laplacian matrices $\mathbf{A}$ and $\hat{S}_{\text{simple}}$, we use the implementation BoomerAMG [51] from HYPRE library.

Algebraic Multigrid Method

When we solve system (2.4) using Algorithm 1, most of the time is spent on inverting matrices A and $\hat{S}$ at each step of the outer iterations. The overall efficiency of the method is determined by the efficiency of the constructed preconditioners for A and $\hat{S}$. Multigrid methods are proven to work well for elliptic problems. However, the classical Geometric Multigrid method is not easy to apply in the case of complex pore-scale geometry. The Algebraic Multigrid approach generalizes the principles of the Geometric Multigrid method for complex geometries and discontinuous coefficients [45]. Recent advances in monolithic multigrid methods for the Stokes problem are noteworthy here, as detailed in references such as [52, 53, 54, 55, 56, 57]. However, the application of monolithic multigrid to the coupled matrix $\mathbb{A}$ is non-trivial and remains an active research topic. In real-world scenarios, efficient solvers for the Stokes equations usually employ multigrid only for the viscous term. This is because the Schur complement is generally believed to be well-represented by the identity operator, for instance, the weighted mass matrix in the Finite Element Method (FEM) [58, 59, 60].

Discussion on flexible Krylov subspace methods

According to Algorithm 1, we do not employ a flexible version of the Krylov subspace method for the outer iterative process, although we do not compute exact solutions during the inner iterations in our practical applications (in 3D). This approach aligns with the consensus among numerous researchers, who assert that if the inner iterations are solved with sufficient accuracy, there is no necessity for a flexible version of the outer iterative method. Moreover, we consider Axelsson's assertion [27], Section 3.1, p.615, which claims that the CG-accelerated inner-outer Uzawa algorithm, used for solving the Schur complement formulation of the Stokes problem, is not sensitive to the tolerance used for inner iterations. At least it is less sensitive then the classical, not accelerated Uzawa algorithm. Our observations also confirm that the CG-SIMPLE and CG-Uzawa algorithms converge well when a reasonable tolerance for the inner iterations is maintained.

2.4 Validation of the developed Stokes solver

The accuracy of the developed Stokes solver is validated by comparing it with data from the literature and with simulations performed with validated solvers. For validation, 3D CT images of real rock samples and synthetic 2D geometries are employed. The absolute permeability serves as an objective functional, which is of interest in our practical application. The results presented in this section are computed using the CG-SIMPLE Algorithm 1. However, the CG-Uzawa solver produces identical results, as the only variation lies in the preconditioners for these two solvers. The primary goal here is to validate the underlying discretization scheme.

2.4.1 3D rock samples from tight reservoirs

Preliminaries: samples and notations

We consider six 3D images: five ultra-tight samples $A-E$ earlier considered in [21], and one image of the classical Berea's sandstone with medium porosity [37]. The corresponding pore space images are depicted on Fig. 2.1 using the Geodict visualization tool. The samples $A-E$ are scanned with resolution 1.2 mkm and have the size $\mathrm{n}=600$, and the sample S is scanned with the resolution 4 mkm and has the size $\mathrm{n}=300$. Detailed information about the samples can be found in Table 2.1, which includes the reference permeability $k_{\mathrm{zz}}^{\mathrm{ref}}$ computed according to (2.17) solving (2.4) for the respective samples with the CG-SIMPLE with very high accuracy, the number of fluid voxels $\mathbb{V}^f=|\mathbb{I}_f^{\mathrm{n}}|$, and the porosity ϕ, defined as follow:

$$\phi=(\mathbb{V}^f/\mathbb{V})\cdot 100\%, \tag{2.26}$$

where $\mathbb{V}=|\mathbb{I}^{\mathrm{n}}|=\mathrm{n}^d$ is the total number of voxels. Additionally, for each sample we compute the surface-to-volume ratio which is defined as follows:

$$\sigma_s=(\mathbb{V}^s_{surf}/\mathbb{V}^f)\cdot 100\%, \tag{2.27}$$

where the surface area $\mathbb{V}^s_{surf}$ of the no-slip boundary is determined as the number of near-boundary solid voxels, i.e., solid voxels face-adjacent with the fluid domain:

$$\mathbb{V}^s_{surf}=|\mathbb{I}^{\mathrm{n}}_{surf}|,\quad \mathbb{I}^{\mathrm{n}}_{surf}=\{(i,j,k)\in\mathbb{I}^{\mathrm{n}}_s:\omega_{(i,j,k)}\cap\overline{\Omega}_f\neq\emptyset\}. \tag{2.28}$$

TABLE 2.1: Description of 3D samples $A-S$ depicted in Fig. 2.1 including the problem size n, reference permeabilities $k_{\mathrm{zz}}^{\mathrm{ref}}$, the number of fluid voxels $\mathbb{V}^f$, the porosity ϕ, and the surface-to-volume ratio σ_s.

	A	**B**	**C**	**D**	**E**	**S**
size n	600	600	600	600	600	300
perm. $k_{\mathrm{zz}}^{\mathrm{ref}}$, mD	0.65	0.80	0.34	11.4	0.74	$6.61\cdot 10^3$
# pores $\mathbb{V}^f$, mln.	12.5	9.4	11.4	20.9	13.9	5.7
porosity ϕ, %	5.8	4.4	5.3	9.7	6.4	21.1
s-t-v ratio σ_s, %	74	56	70	55	61	28

Validation with commercial software GeoDict

In this section, we study 3D rock samples from our practical applications discribed in the previous subsection. We compare the performance of the CG-SIMPLE algorithm with the performance of the commercial solver GeoDict (GeoDict 2023 Service Pack 4 Standard Edition) [12], which is considered the state-of-the-art solver for DRP. Inside GeoDict, we used GeoDict (LIR) flow solver [61]. Note, we have designed our computational experiments to closely match the settings for both solvers. As for the boundary consitions, for both solvers, we applied periodic boundary conditions in the flow direction and the no-slip boundary conditions in the tangential direction.
The results of the simulations are presented in Table 2.2 for different stopping criteria (convergence tolerance). First of all, the results indicate that the permeabilities calculated by GeoDict are in good agreement with those computed using the CG-SIMPLE algorithm. Additionally, we provide the computational times measured on an isolated computational node with 48 CPU kernels. It's worth noting that the GeoDict

(LIR) solver leverages adaptive grid coarsening, making it highly efficient for higher porosities. However, despite the use of adaptive grids, GeoDict (LIR) exhibits inferior performance compared to CG-SIMPLE when dealing with low porosity samples.

TABLE 2.2: Cross-validation of the developed Stokes solver with GeoDict (LIR) for binary images $A-S$ described in Table 2.1 and pictured in Fig. 2.1. Comparison of the absolute permeability computed for different stopping criteria and computational time.

A, $\phi = 5.8$, $\sigma_s = 74$					
GeoDict (LIR)			SCoPeS-S (CG-SIMPLE)		
Tol.	Perm [mDa]	CPU Time [sec]	Tol. ε_S	Perm [mDa]	CPU time [sec]
$2 \cdot 10^{-1}$	0.99	486	10^{-1}	1.61	76+166
$5 \cdot 10^{-2}$	0.81	861	10^{-2}	0.69	77+402
$1 \cdot 10^{-2}$	0.64	3092	10^{-3}	0.66	75+589
B, $\phi = 4.4$, $\sigma_s = 56$					
GeoDict (LIR)			SCoPeS-S (CG-SIMPLE)		
Tol.	Perm [mDa]	CPU Time [sec]	Tol. ε_S	Perm [mDa]	CPU time [sec]
$1 \cdot 10^{-1}$	0.79	937	10^{-2}	0.83	69+322
$1 \cdot 10^{-2}$	0.78	1801	10^{-3}	0.81	68+548
C, $\phi = 5.3$, $\sigma_s = 70$					
GeoDict (LIR)			SCoPeS-S (CG-SIMPLE)		
Tol.	Perm [mDa]	CPU Time [sec]	Tol. ε_S	Perm [mDa]	CPU time [sec]
$1 \cdot 10^{-1}$	0.37	1069	10^{-2}	0.36	75+383
$1 \cdot 10^{-2}$	0.31	3137	10^{-3}	0.35	77+719
D, $\phi = 9.7$, $\sigma_s = 55$					
GeoDict (LIR)			SCoPeS-S (CG-SIMPLE)		
Tol.	Perm [mDa]	CPU Time [sec]	Tol. ε_S	Perm [mDa]	CPU time [sec]
$1 \cdot 10^{-1}$	9.6	964	10^{-2}	11.4	157+747
$1 \cdot 10^{-2}$	10.2	1546	10^{-3}	11.4	161+1261
E, $\phi = 6.4$, $\sigma_s = 61$					
GeoDict (LIR)			SCoPeS-S (CG-SIMPLE)		
Tol.	Perm [mDa]	CPU Time [sec]	Tol. ε_S	Perm [mDa]	CPU time [sec]
$1 \cdot 10^{-1}$	0.76	2083	10^{-2}	0.82	104+650
$1 \cdot 10^{-2}$	0.74	3173	10^{-3}	0.74	108+1124
S, $\phi = 21.1$, $\sigma_s = 28$					
GeoDict (LIR)			SCoPeS-S (CG-SIMPLE)		
Tol.	Perm [mDa]	CPU Time [sec]	Tol. ε_S	Perm [mDa]	CPU time [sec]
$1 \cdot 10^{-2}$	$6.71 \cdot 10^3$	54	10^{-2}	$6.62 \cdot 10^3$	48+325
$1 \cdot 10^{-3}$	$6.66 \cdot 10^3$	125	10^{-3}	$6.61 \cdot 10^3$	49+582

2.4.2 Validation on synthetic geometry - periodic array of spheres

The developed Stokes solver is next validated on the classic example of flow around periodic arrangements of solid spheres. Imposing periodic boundary conditions in all three directions, one can consider only a single sphere. The computed permeability

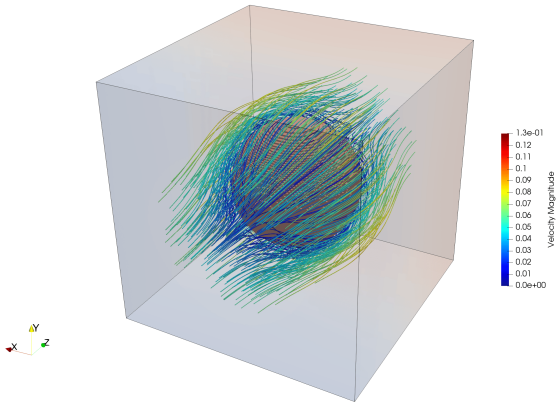


FIGURE 2.3: Velocity magnitude streamlines for periodic arrangements of spheres; size $L = 1$, sphere diameter $D = 0.5$, resolution $\mathrm{n} = 160$.

is compared with numerical and analytical results from [62] and [63]. An exemplary computation is visualized in Figure 2.3. The results are summarized in Table 2.3. The simulation results show that The developed method correctly computes the permeability. Also, convergence with respect to the grid size is clearly observed.

TABLE 2.3: Dimensionless permeability $\hat{k}_{zz}$ for periodic array of impermeable spheres. D and N denote the diameter of the spheres (with respect to the unit length of the cube) and the number of voxels in one direction, respectively.

D	N=40	N=80	N=160	J&T (ref [62])	S&A (ref [62], [63])
0.1	$9.74 \cdot 10^{-1}$	$9.01 \cdot 10^{-1}$	$9.02 \cdot 10^{-1}$	$9.15 \cdot 10^{-1}$	$9.11 \cdot 10^{-1}$
0.2	$3.77 \cdot 10^{-1}$	$3.78 \cdot 10^{-1}$	$3.80 \cdot 10^{-1}$	$3.84 \cdot 10^{-1}$	$3.82 \cdot 10^{-1}$
0.4	$1.21 \cdot 10^{-1}$	$1.22 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$1.25 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$
0.6	$4.44 \cdot 10^{-2}$	$4.43 \cdot 10^{-2}$	$4.43 \cdot 10^{-2}$	$4.58 \cdot 10^{-2}$	$4.45 \cdot 10^{-2}$
0.8	$1.29 \cdot 10^{-2}$	$1.31 \cdot 10^{-2}$	$1.31 \cdot 10^{-2}$	$1.38 \cdot 10^{-2}$	$1.32 \cdot 10^{-2}$
1.0	$2.48 \cdot 10^{-3}$	$2.51 \cdot 10^{-3}$	$2.51 \cdot 10^{-3}$	$2.67 \cdot 10^{-3}$	$2.52 \cdot 10^{-3}$

2.5 Convergence study and spectral analysis of the CG-Uzawa and CG-SIMPLE algorithms

In the present Section, computational experiments are conducted to numerically investigate:

- Possible correlation between the surface-to-volume ratio and the condition number of the preconditioned\unpreconditioned Schur complement matrix;
- Possible correlation between the number of boundary nodes where no-slip boundary conditions are imposed and the number of non-unit eigenvalues of the Schur complement matrix.

- The performance of the Uzawa and SIMPLE preconditioners in solving the Schur complement problem (2.4), especially in complex geometries with high surface-to-volume ratio;
- The performance of two preconditioners in computing the permeability of representative 3D and 2D samples according to (2.17);

Two sets of experiments are considered.
In the first set, 3D CT images of rock samples from tight reservoirs (characterized by low porosity) are analyzed. The performance of the inexact Uzawa and SIMPLE preconditioners in conjunction with the Conjugate Gradient method for the Schur complement matrix is compared. It is observed that the SIMPLE preconditioner is more efficient for this class of problems. A correlation between the surface-to-volume ratio and the condition number of the Schur complement matrix is established. Additionally, the permeability computed using both preconditioners is compared, showing that the CG-SIMPLE method offers a more robust and accurate calculation.
In the second set of experiments, a more detailed examination of the two preconditioners is conducted, using synthetic 2D geometries and exact inner iterations. Besides comparing the performance of the preconditioners, we perform an in-depth study of the spectra of the Schur complement matrix. Based on the obesravtions, it is conjectured that the no-slip boundary conditions primarily cause the non-unit eigenvalues of the Schur complement matrix. For the synthetic samples examined, it is clearly observed that an increase in the surface-to-volume ratio results in an increased condition number of the Schur complement matrix. Conversely, there is an inverse dependence when the Schur complement matrix is preconditioned with the SIMPLE.

2.5.1 3D rock samples from tight reservoirs

Performance of the preconditioners in solving the Schur complement system and in computing permeability

In this section, we study the performance of the CG-Uzawa and CG-SIMPLE algorithms for the pore space images of real rock samples from tight reservoirs (see Fig. 2.1). As the stopping criteria for the outer CG iterations, we use $\varepsilon_S = 10^{-3}$ for both the CG-Uzawa and CG-SIMPLE algorithms. The relative preconditioned residual norm, as defined in (2.20), is used here. For the inner iterations, we use a higher precision $\varepsilon_{\mathbf{A}} = 10^{-6}$ in both cases. Also, for the CG-SIMPLE algorithm we use $\varepsilon_{\hat{S}} = 10^{-6}$ for solving with the SIMPLE preconditioner $\hat{S}_{\text{simple}}$. The reference permeabilities k_{zz}^{ref} were computed using the CG-SIMPLE algorithm with higher precision $\varepsilon_S = 10^{-5}$, $\varepsilon_{\mathbf{A}} = 10^{-8}$, $\varepsilon_{\hat{S}} = 10^{-8}$. Such inexact solves for $\mathbf{A}$ and $\hat{S}_{\text{simple}}$ make it difficult to rigorously analyze the underlying numerical methods. However, the purpose of this section is to demonstrate convergence problems with the established Uzawa preconditioner that occur in practical permeability calculations when computing flows in tight porous media. So, we compare methods for the settings that we usually use in our practical calculations.
The convergence history for the selected tolerances is presented in Fig. 2.4, where the relative unpreconditioned residual norm $\|\hat{S}^{-1}r_S^k\|/\|\hat{S}^{-1}r_S^0\|$ is shown on the top graph. Furthermore, the relative permeability error, defined as:

$$e^k = |k_{zz}^k - k_{zz}^{\text{ref}}|/k_{zz}^{\text{ref}}, \tag{2.29}$$

is shown in Fig. 2.4 on the bottom graph.

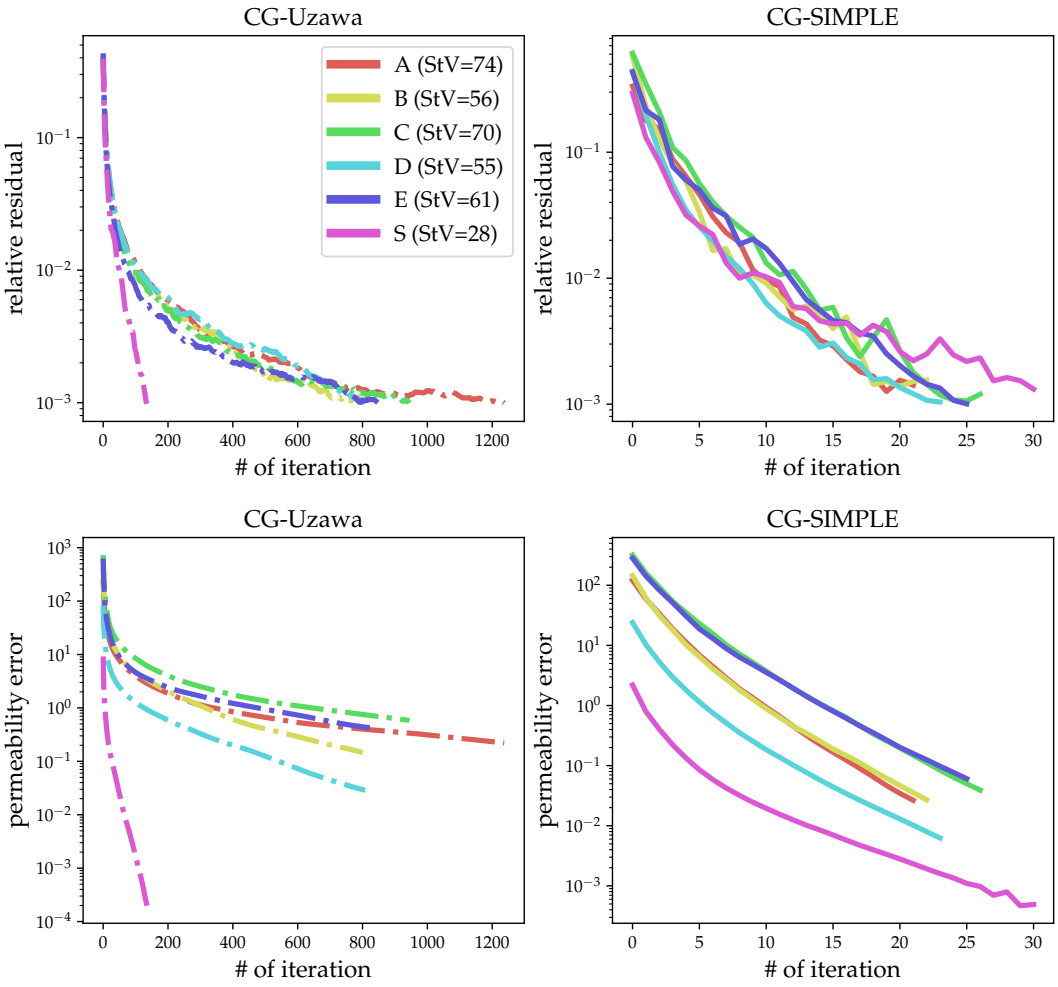


FIGURE 2.4: Convergence history of the CG-SIMPLE and CG-Uzawa algorithms for 3D samples $A - S$ described in the Table 2.1. Preconditioned relative residual norm $\|\hat{S}^{-1}r_S^k\|/\|\hat{S}^{-1}r_S^0\|$ (top) and the permeability error e^k defined in (2.29) (bottom) are shown.

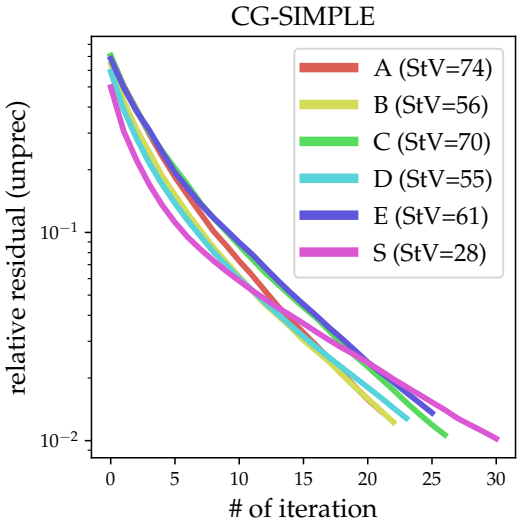


FIGURE 2.5: Convergence history of the CG-SIMPLE algorithm for 3D samples $A - S$ described in the Table 2.1. Unpreconditioned relative residual norm $\|r_S^k\|/\|r_S^0\|$ decreases monotonically compared with preconditioned residual norm $\|\hat{S}^{-1}r_S^k\|/\|\hat{S}^{-1}r_S^0\|$ (compare with Fig. 2.4).

The convergence summary can be found in Table 2.4, which includes the number of iterations, the required computational time, and the permeability error $e^{\#}$ computed on the final iteration. Additionally, we provide estimations for the condition numbers of the preconditioned and unpreconditoned Schur complement matrices, computed for free during the outer CG iterations using Lanczos algorithm.

TABLE 2.4: Summary of the results for CG-Uzawa and CG-SIMPLE algorithms for 3D samples $A - S$ described in the Table 2.1 including relative permeability error on the final iteration $e^{\#}$, the total computational time, the number of iterations, and the estimated condition numbers.

	A	**B**	**C**	**D**	**E**	**S**
CG-Uzawa:						
perm. error $e^{\#}$, %	0.221	0.146	0.585	0.029	0.403	0.0002
total comp. time, hrs	8.5	5.0	6.2	10.9	7.3	0.8
# iters	1238	802	946	809	849	136
$\approx \mathrm{cond}(S)$	$7.1 \cdot 10^5$	$2.1 \cdot 10^5$	$3.7 \cdot 10^5$	$2.2 \cdot 10^5$	$3.4 \cdot 10^5$	$3.4 \cdot 10^3$
CG-SIMPLE:						
perm. error $e^{\#}$, %	0.019	0.020	0.031	0.005	0.047	0.0005
total comp. time, hrs	0.3	0.3	0.3	0.7	0.5	0.3
# iters	22	23	27	24	26	31
$\approx \mathrm{cond}(\hat{S}^{-1}_{\mathrm{simple}}S)$	$0.9 \cdot 10^2$	$1.2 \cdot 10^2$	$1.4 \cdot 10^2$	$1.1 \cdot 10^2$	$1.8 \cdot 10^2$	$2.4 \cdot 10^2$

The following hardware was used in our numerical experiments: 48x MPI compute node (Dell PowerEdge M640), dual Intel Xeon Gold 6132 ("Skylake") @ 2.6 GHz, i.e. 28 CPU cores per node. The computational times shown in Table 2.4 were obtained using 8 CPU nodes.
Several observations can be drawn from the results presented in Table 2.4. For the

considered low porosity, high surface-to-volume ratio images, the SIMPLE preconditioner appears to perform better than the Uzawa preconditioner, as it converges robustly while the CG-Uzawa tends to stagnate. Firstly, despite the CG-SIMPLE algorithm being more expensive (approximately x1.5) per iteration, its total computational time is smaller compared to the CG-Uzawa because significantly fewer number of iterations is required. Secondly, the estimated condition number for the preconditioned Schur complement matrix is about three orders of magnitude smaller for the SIMPLE preconditioner than for the Uzawa preconditioner. For moderate porosity (sample S), the condition number for both preconditioners is comparable. Thirdly, the CG-SIMPLE computes the permeability much more accurately for the considered samples. Actually, achieving even 10% accuracy in computing permeability is not always possible with the established CG-Uzawa method.
It is worth mentioning, that for the CG-SIMPLE algorithm oscillations may appear in the permeability error (see Fig. 2.4, sample S). This is the case when inner tolerance $\varepsilon_{\hat{S}}$ for inverting preconditioner $\hat{S} = \hat{S}_{\text{simple}}$ is not small enough. However, despite these oscillations, the permeability error continues to decrease, albeit not monotonically. This confirms our conjecture that using flexible Krylov method is not necsesary in our case. Note, similar oscillations may appear if inner tolerance $\varepsilon_{\mathbf{A}}$ for inverting $\mathbf{A}$ when applying S is not small enough. It is also worth mentioning that the unpreconditioned residual for the CG-SIMPLE algorithm *always* decreases monotonically in our numerical experiments (i.e. even when oscillations appear in the permeability error). In Fig. 2.5 in the Appendix, we show the unpreconditioned residual for the CG-SIMPLE algorithm corresponding to the stopping criteria $\varepsilon_S = 10^{-3}$ in the preconditioned residual norm, as it is shown in Fig. 2.4. Note, the unpreconditioned residual norm is suppressed only up to 10^{-2} in this case. We investigate this issue in the subsequent section for synthetic 2d geometries, where we use the unpreconditioned residual norm directly as the stopping criteria for the preconditioned CG for both methods. Detailed comparison of preconditioned/unpreconditioned residuals for CG-SIMPLE and CG-Uzawa for different thresholds $\varepsilon_S = 10^{-1}$, $\varepsilon_S = 10^{-2}$, $\varepsilon_S = 10^{-3}$ is shown in Tables 2.8, 2.9, 2.10 in the Appendix, respectively.

Correlation between the surface-to-volume ratio and the condition number of the Schur complement matrix

In Fig. 2.6, we show that for the considered samples $A - S$ there is a strong correlation between the surface-to-volume ratio σ_s (2.27) and the estimated condition number of the unpreconditioned Schur complement matrix. This is also confirmed by the number of iterations for the CG-Uzawa algorithm. However, the dependence in the case of the Schur complement matrix preconditioned with the SIMPLE is not so distinctive here, so we perform a more rigorous study in the subsequent Section 2.5.2.

2.5.2 2D synthetic geometries - periodic array of randomly shifted squares

To identify consistent patterns that affect the performance of the methods under consideration in complex pore space domains, we consider synthetic 2D geometries with a transparent generation process and directly available geometric information (such as porosity, no-slip surface area, etc). As in the 3D case, we present and discuss the convergence of the two algorithms, as well as the accuracy with which the permeability is computed. Additionally, for the considered synthetic samples we compute

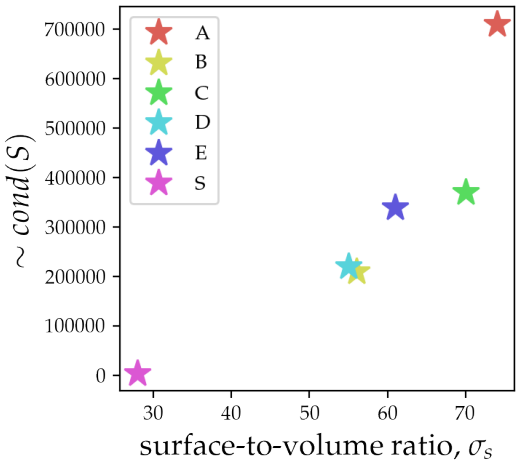
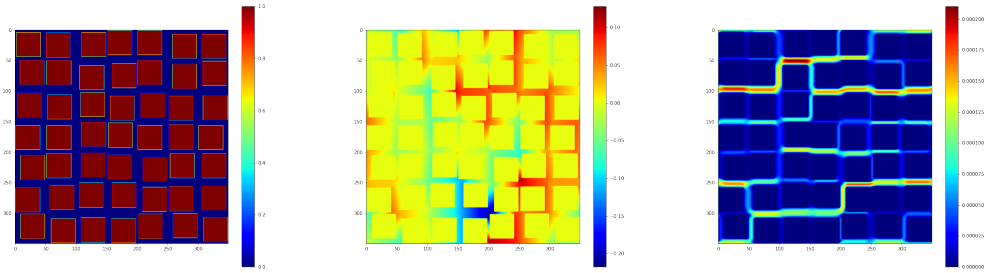


FIGURE 2.6: Correlation between surface-to-volume ratio and estimated condition number for the samples $A - S$ described in the Table 2.1.



(A) geometry (B) pressure (C) velocity magnitude

FIGURE 2.7: Example of synthetic 2D geometry: array of randomly shifted square obstacles for $N = 7$, $n_{\mathrm{s}} = 40$, $n_{\mathrm{avg}} = 10$, $n_{\mathrm{min}} = 2$.

and analyse the full spectra of the preconditioned and unpreconditioned Schur complement matrices.

Generation of synthetic 2D geometries.

We study flows passing around arrays of solid square obstacles randomly placed in a fluid bed. First of all, a uniform voxel (pixel in 2D) grid is generated in Ω as described in Section 1. For ease of generation, we consider square obstacles of the same size, and each obstacle is located in the center of the square cell, or is slightly shifted, so that a cell contains the obstacle and a part of the flow domain around it. The obstacles do not touch the boundary of the cell. Each generated geometry is defined by four integer parameters $(N, n_{\mathrm{c}}, n_{\mathrm{avg}}, n_{\mathrm{min}})$, where N and n_{c} determine the number of cells in one direction and their size measured in voxels, while the parameters n_{avg} and n_{min} control the average and minimal thicknesses (in voxels) of the fluid channels between two adjacent solid squares. Note, that the cells and the obstacles in all cases are adjusted to the introduced computational grid, so that each voxel is fully occupied either by fluid or by solid. An example for $N = 7$, $n_{\mathrm{c}} = 50$, $n_{\mathrm{avg}} = 10$, $n_{\mathrm{min}} = 2$ is presented in Fig. 2.7a. Formally, we have a domain of the total size $\mathrm{n} \times \mathrm{n}$, $\mathrm{n} = Nn_{\mathrm{c}}$

which represents an $N \times N$ array of cells of the size $n_c \times n_c$; each cell contains a solid square of the size $(n_c - n_{avg}) \times (n_c - n_{avg})$ with the origin $(n_c/2 + r_1, n_c/2 + r_2)$, where $r_1, r_2 \in [-(n_{avg} - n_{min})/2, (n_{avg} - n_{min})/2]$ are (integer) random shifts. It should be noted, that randomness is necessary to observe non-trivial solutions which take place in the case of fully periodic arrays.

Performance of the preconditioners in solving the Schur complement system and in computing permeability

In the present section, we investigate the effect of surface-to-volume ratio on the convergence of the algorithms by varying the average thickness of the channels of the synthetic 2D geometries. Namely, we randomly generated five geometries according to the procedure described in the previous subsection for $N = 7$, $n_c = 50$, $n_{min} = 2$, and $n_{avg} = \{4, 6, 8, 10, 12\}$. Detailed information about the samples can be found in Table 2.5, which includes the reference permeability k_{xx}^{ref}, the number of fluid voxels $\mathbb{V}^f$, the porosity ϕ (2.26), and the surface-to-volume ratio σ_s (2.27). It should be noted, that the average channel thickness for synthetic 2D geometries is directly related to the surface-to-volume ratio for general porous media.

TABLE 2.5: Description of the synthetic 2D geometries with variable channel thicknesses n_{avg} including the problem size n, reference permeabilities k_{xx}^{ref}, the number of fluid voxels $\mathbb{V}^f$, the porosity ϕ, and the surface-to-volume ratio σ_s.

	$n_{avg} = 4$	$n_{avg} = 6$	$n_{avg} = 8$	$n_{avg} = 10$	$n_{avg} = 12$
size n	350	350	350	350	350
perm. k_{xx}^{ref}	$1.0 \cdot 10^{-6}$	$3.2 \cdot 10^{-6}$	$7.3 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$
# pores $\mathbb{V}^f$, thsnd.	18.8	27.6	36.0	44.1	51.8
porosity ϕ, %	15.4	22.6	29.4	36.0	42.3
s-t-v ratio σ_s, %	46.9	30.5	22.3	17.3	14.0

As the stopping criteria for the outer CG iterations, we use $\varepsilon_S = 10^{-3}$ for both the CG-SIMPLE and CG-Uzawa algorithms. In contrast to the previous subsection, we consider the unpreconditioned relative residual here. Considering residuals in the same norm allows for straightforward comparison between the two algorithms. A detailed comparison of preconditioned/unpreconditioned residuals for the CG-SIMPLE and CG-Uzawa for different thresholds $\varepsilon_S = 10^{-2}$, $\varepsilon_S = 5 \cdot 10^{-3}$, $\varepsilon_S = 10^{-3}$ is shown in Tables 2.11, 2.12, 2.13 in the Appendix, respectively. As for the inner iterations, in this experiment we use the machine epsilon $\varepsilon_{\mathbf{A}} = \varepsilon_{\hat{S}} = 10^{-13}$. In particular, this means that the exact Uzawa is used here. The convergence history for selected tolerances is presented in Fig. 2.8, where the relative unpreconditioned residual norm $\|r_h^k\|/\|r_h^0\|$ is shown on the left and the permeability error e^k defined in (2.29) is shown on the right.

The convergence summary can be found in Table 2.6, which includes the number of iterations until convergence, the required computational time, and the permeability error $e^{\#}$ computed on the final iteration. In addition, we show the effective condition numbers of the preconditioned and unpreconditoned Schur complement matrices, which are computed using a direct method. Similar observations to those made for the 3D simulations can be made from the results presented in Table 2.6. Again, the CG-SIMPLE algorithm requires less iterations and less total computational time compared to the CG-Uzawa for samples with larger surface-to-volume ratio. When decreasing surface-to-volume ratio, the CG-Uzawa tends to show better performance.

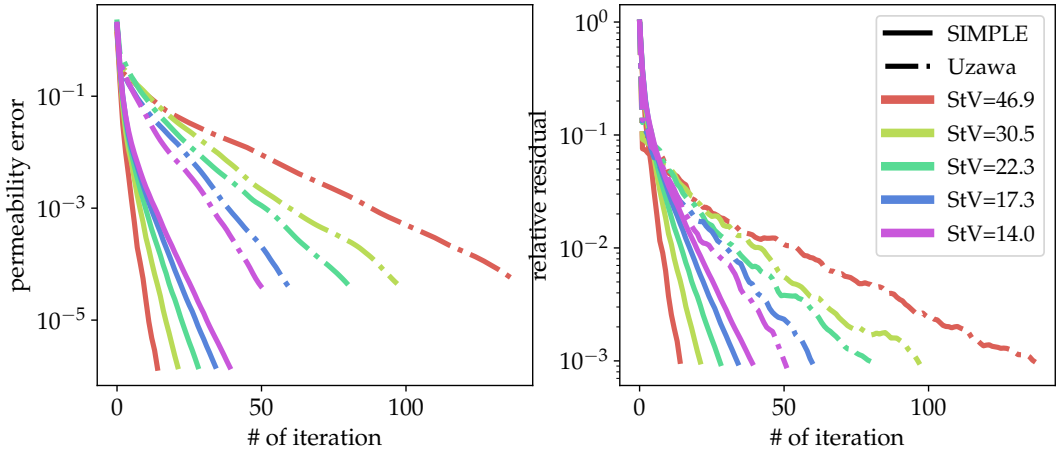


FIGURE 2.8: Convergence history of the CG-SIMPLE and CG-Uzawa algorithms for the synthetic 2D geometries with variable channel thicknesses n_{avg} described in the Table 2.5. Unpreconditioned relative residual norm $\|r_h^k\|/\|h_S^0\|$ (right) and relative permeability error e^k defined in (2.29) (left) are shown.

As in the 3D case, the CG-SIMPLE more accurately computes the permeability, which is clearly seen from the left graph in Fig. 2.8. This may seem counterintuitive since the residuals have the same norm for both algorithms, as illustrated in the right graph of Fig. 2.8. This phenomenon requires a deeper theoretical investigation, which is beyond the scope of this paper. We currently hypothesize that the primary reason is that the iterative solutions computed by the two algorithms belong to different Krylov subspaces. Furthermore, it is evident from Tables 2.11, 2.12, 2.13 that as the tolerance ε_S for the outer iterations decreases, the difference in permeability calculations decreases as well.

TABLE 2.6: Convergence summary of the CG-Uzawa and CG-SIMPLE algorithms for the synthetic 2D geometries with variable channel thicknesses n_{avg} described in the Table 2.5 including the permeability error on the final iteration $e^{\#}$, the total computational time, the number of iterations, and the condition numbers.

	$n_{\text{avg}} = 4$ (StV=46.9)	$n_{\text{avg}} = 6$ (StV=30.5)	$n_{\text{avg}} = 8$ (StV=22.3)	$n_{\text{avg}} = 10$ (StV=17.3)	$n_{\text{avg}} = 12$ (StV=14.0)
CG-Uzawa:					
perm. error $e^{\#}$, %	$5.4 \cdot 10^{-5}$	$4.4 \cdot 10^{-5}$	$4.3 \cdot 10^{-5}$	$3.6 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$
total comp. time, s	6.5	6.0	5.8	5.0	4.7
# iters	138	98	81	61	52
$\text{cond}(S)$	$4.7 \cdot 10^{3}$	$2.3 \cdot 10^{3}$	$1.3 \cdot 10^{3}$	$7.7 \cdot 10^{2}$	$5.3 \cdot 10^{2}$
CG-SIMPLE:					
perm. error $e^{\#}$, %	$1.4 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$
total comp. time, s	1.4	2.4	3.5	4.6	5.7
# iters	15	22	29	35	40
$\text{cond}(\hat{S}^{-1}_{\text{simple}} S)$	$3.4 \cdot 10^{1}$	$8.6 \cdot 10^{1}$	$1.6 \cdot 10^{2}$	$2.6 \cdot 10^{2}$	$3.4 \cdot 10^{2}$

It is important to note that for the 2D geometries under consideration, the convergence of the CG-Uzawa algorithm closely follows a linear asymptote. Specifically, the

convergence factor is determined by the effective condition number:

$$\frac{\sqrt{\mathrm{cond}(S)}-1}{\sqrt{\mathrm{cond}(S)}+1} \approx 1 - \frac{2}{\sqrt{\mathrm{cond}(S)}}, \quad \text{for} \quad \mathrm{cond}(S) \gg 1. \tag{2.30}$$

Taking into account $\lambda_{\max}(S) = 1$, (2.30) becomes:

$$\frac{1-\sqrt{\lambda_{\min}(S)}}{1+\sqrt{\lambda_{\min}(S)}}, \tag{2.31}$$

where $\lambda_{\min}(S) > 0$ is the smallest non-zero eigenvalue. Recall, the linear convergence factor for the CG-Uzawa formulated in operator setting (see, e.g., [27]) is given as follows:

$$\frac{1-\gamma}{1+\gamma}, \tag{2.32}$$

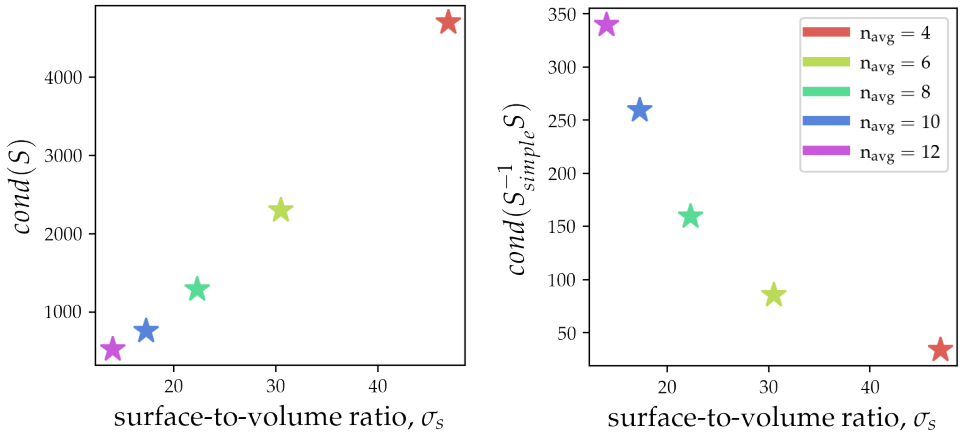
where γ denotes the inf-sup constant. Comparing (2.31) with (2.32), we observe the parallel between $\lambda_{\min}(S)$ and γ^2.
Notably, for the CG-SIMPLE, the convergence is super-linear, which means some clustering of eigenvalues. Similar spectral clustering is reported in [36].

Correlation between the surface-to-volume ratio and the condition number of the Schur complement matrix

In Fig. 2.9, for the considered 2D geometries with variable channel thicknesses, we show the dependence between the surface-to-volume ratio σ_s (2.27) and the condition number of the unpreconditioned/preconditioned Schur complement matrix. The main conclusion that we draw is that the condition number of the Schur complement matrix increases linearly when increasing the surface-to-volume ratio. However, unlike the 3D case, for the considered here synthetic 2D geometries, the inverse dependence is also clearly observed for the Schur complement matrix preconditioned with the SIMPLE. Specifically, the condition number of the preconditioned matrix decreases super-linearly with increasing surface-to-volume ratio. Moreover, we conjecture that the condition number of the Schur complement matrix is influenced by the number of non-unit eigenvalues in its spectrum. In the next subsection, we show that the large ratio of non-unit eigenvalues in the spectrum of the Schur complement matrix is directly related to the large surface-to-volume ratio.

Number of non-unit eigenvalues of the Schur complement matrix and no-slip surface area

In the present section, for the synthetic 2D geometries, we show the relation between surface-to-volume ratio and the number of non-unit eigenvalues of the Schur complement matrix. We consider various configurations of synthetic 2D geometries. First, we vary the number of squares N, which determines the degree of connectivity of the flow domain Ω_h^f. Second, for each N we vary the surface area $\mathbb{V}^s_{surf}$ defined in (2.28). For the selected configurations, we compute the full spectrum of the Schur complement matrix S and calculate the number of its eigenvalues, denoted N_{ev}, which are not equal to one (including the zero eigenvalue). The results are presented in Figure 2.10, where we show the dependence between the surface area $\mathbb{V}^s_{surf}$ and the number of non-unit eigenvalues N_{ev}. For the considered here class of geometries, we observe that the following empirical formula for the number of non-unit eigenvalues



(A) Uzawa (B) SIMPLE

FIGURE 2.9: Correlation between surface-to-volume ratio and condition number for the synthetic 2D samples with various channel thicknesses n_{avg} described in the Table 2.5.

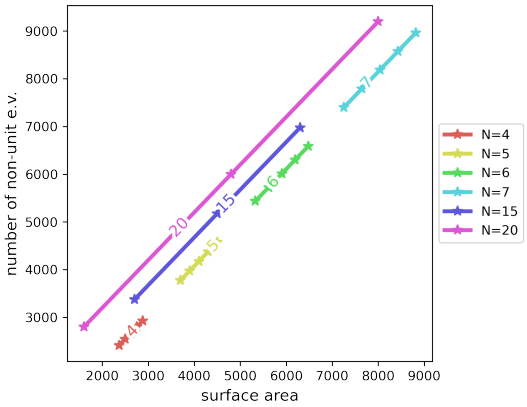


FIGURE 2.10: Dependence of the number of non-unit eigenvalues of the Schur complement matrix N_{ev} on the connectivity of the flow domain and the surface area of the no-slip boundary $\mathbb{V}^s_{surf}$ defined in (2.28). The number of square obstacles N in one dimension is shown by color, total N^2 obstacles.

holds:

$$N_{ev} = \mathbb{V}^s_{surf} + 3N^2 - 1. \tag{2.33}$$

Note, the surface area $\mathbb{V}^s_{surf}$ coincides with the number of boundary nodes (i.e., the nodes lying on Γ_0), where the Dirichlet b.c. on the tangential velocity component is imposed (see Section 2.6 for rigorous explanation). In particular, the formula (2.33) reveals that for a simply-connected domain, the surface-to-volume ratio equals the ratio of non-unit to unit eigenvalues of the Schur complement matrix. Thus, the greater is the ratio, the further the Schur complement is from the identity, and the worse is the performance of the Uzawa preconditioner.

2.6 Theoretical justification in the case of simplest geometry

In the previous Section 2.5, it was shown numerically that there is a correlation between the surface-to-volume ratio and 1) the condition number of the Schur complement matrix S, 2) the ratio of non-unit to unit eigenvalues of S. The primary aim of the present Section is to provide theoretical justification of these results. Investigating these results theoretically in complex domains is difficult, therefore we investigate them in the case of simplest geometry - the square in 2D. We describe the fully-staggered finite-difference discretization of the Stokes equations in subsection 2.6.2. In the case of square domain, the tensor-structured grids can be considered for the discretization which allow for the underlying discrete operators to be assembled in a simple way using the Kronecker matrix product. Note, the present Section is accompanied with the 50-lines Python code provided in the Appendix 2.C.

2.6.1 Primary and auxiliary Stokes BVPs under considiration

Let us recall the Stokes equations formulated in a bounded open domain $\Omega \subset \mathbb{R}^2$:

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \mathbf{0} \quad \text{in } \Omega, \\ -\nabla \cdot \mathbf{u} &= 0 \quad \text{in } \Omega, \end{aligned} \tag{2.34}$$

where $\mathbf{u} = (u, v)^T$ denotes the fluid velocity and p is the fluid pressure. Let us further denote $\mathbf{n}$ the outward unit normal of the boundary $\partial\Omega$. Then, one can uniqely write the following decomposition for the velocity vector field $\mathbf{u}$:

$$\mathbf{u} = \mathbf{u}_\perp + \mathbf{u}_\parallel,$$

where $\mathbf{u}_\perp = (\mathbf{n} \cdot \mathbf{u})\mathbf{n}$ and $\mathbf{u}_\parallel = \mathbf{u} - \mathbf{u}_\perp$ denote the normal and tangential components, respectively. First, we consider the no-penetration boundary condition imposed on the entire boundary $\partial\Omega$, which is given as follows:

$$\mathbf{u}_\perp \ = \mathbf{0} \quad \text{on } \partial\Omega. \tag{2.35}$$

Second, we prescribe the tangential component of the velocity as follows:

$$\mathbf{u}_\parallel = \mathbf{u}^{\mathbf{D}} \quad \text{on } \partial\Omega. \tag{2.36}$$

In the present Section, the BVP (2.34)-(2.36) is considered as the **primary** problem of interest. It is the Dirichlet problem for the Stokes equation, and the no-penetration condition (2.35) is sufficient for its well-posedness. Thus, in contrast to the flow experiment described in Section 2.2.2, the inflow and outflow boundaries are not considered here. This simplification is motivated by our aim to study the impact of the no-slip boundary conditions on the spectrum of the Schur complement matrix, which is not influenced by the periodic boundary conditions. It should be noted, that the boundary condition (2.35) when imposed on the entire boundary $\partial\Omega$ characterizes a special class of flow problems, called the *enclosed* flow problems (terminology from [64], Ch. 5, P. 215). For the enclosed flows, the pressure is determined only up to a constant nullspace.

Instead of (2.36), we may consider the Neumann boundary condition imposed on the tangential velocity, which is given as follows:

$$\partial \mathbf{u}_{\parallel}/\partial \hat{\mathbf{n}} = \mathbf{g}^{\mathbf{N}} \quad \text{on } \partial\Omega. \tag{2.37}$$

In the present Section, the BVP (2.34),(2.35),(2.37) is denoted as the **auxiliary** problem.

Remark 2.6.1. *Note, that for the case* $\mathbf{u}^{\mathbf{D}} = \mathbf{0}$ *the boundary conditions* (2.35),(2.36) *are the no-slip conditions, and for the case* $\mathbf{g}^{\mathbf{N}} = \mathbf{0}$*, boundary conditiones* (2.35),(2.37) *are the free-slip conditions described in Section 2.A.*

Remark 2.6.2. *Also, the well-known lid-driven cavity problem is a special case of the just described primary BVP with* $\mathbf{u}^{\mathbf{D}} = (0,1)^T$ *on the top boundary (i.e.* $y = 0$*) and* $\mathbf{u}^{\mathbf{D}} = \mathbf{0}$ *on the other boundaries. The lid driven cavity solution for* $\mathrm{n} = 128$ *is pictured in Fig. 2.12, which is computed using Python code provided in Section 2.C.*

Corresponding Laplacian BVPs

Along with the primary and auxiliary BVPs for the Stokes equation, we also consider the corresponding BVPs for the Laplacian equation, obtained by discarding the divergence-free constraint in (2.34). The Laplacian BVP corresponding to the primary Stokes problem is given as follows:

$$\begin{cases} -\Delta \mathbf{u} &= \mathbf{0} \quad \text{in } \Omega, \\ \mathbf{u}_{\perp} &= \mathbf{0} \quad \text{on } \partial\Omega, \\ \mathbf{u}_{\parallel} &= \mathbf{u}^{\mathbf{D}} \quad \text{on } \partial\Omega, \end{cases} \tag{2.38}$$

and the Laplacian BVP corresponding to the auxiliary problem is given as follows:

$$\begin{cases} -\Delta \mathbf{u} &= \mathbf{0} \quad \text{in } \Omega, \\ \mathbf{u}_{\perp} &= \mathbf{0} \quad \text{on } \partial\Omega, \\ \partial \mathbf{u}_{\parallel}/\partial \hat{\mathbf{n}} &= \mathbf{g}^{\mathbf{N}} \quad \text{on } \partial\Omega. \end{cases} \tag{2.39}$$

Section outline

Discretization of the primary (enclosed Dirichlet) BVP is considered in a specific way. Namely, it is considered as a perturbation of the auxiliary (enclosed Neumann) problem. This is motivated by the fact that, in the auxiliary Neumann case, several neat properties hold. Firstly, as described in Section 2.6.3, the Discrete Helmholtz-Hodge orthogonal Decomposition (DHHD) holds true for the discrete curl and discrete divergence operators. Secondly, the corresponding velocity vector Laplacian matrix is decomposed into the direct sum under this decomposition (see Corollary 2.6.10.1). These properties imply that the Schur complement matrix for the auxiliary problem, as well as its inverse, up to a constant factor in the nullspace, equals the identity matrix acting on the discrete pressure space (see Section 2.6.3 for details).
The results regarding structure of the Schur complement matrix are described in Section 2.6.3. Specifically, we demonstrate that the Schur complement matrix assembled for the primary problem, as well as its inverse, can be considered as a rank-r correction of the Schur complement matrix assembled for the auxiliary problem. Moreover, the rank r is determined by the number of the tangential velocity nodes affected by the Dirichlet boundary condition. The main result is formulated in Theorem 2.6.12;

it is related to the case when all the velocity nodes are affected by the Dirichlet condition. In this limiting case, a particularly simple structure of the Schur complement matrix is explicitly derived, which implies several practical outcomes (see Section 2.6.3).

2.6.2 Finite difference discretization on fully-staggered grids

In what follows, we consider Ω from (2.34) to be a unit two-dimensional domain with the following tensor-product structure:

$$\Omega = \omega \times \omega \subset \mathbb{R}^2, \tag{2.40}$$

where ω denotes a unit one-dimensional interval:

$$\omega = (0, 1) \subset \mathbb{R}. \tag{2.41}$$

Given the problem size n and the corresponding grid size $h = 1/\mathrm{n}$, we consider the classical fully-staggered finite difference scheme [39, 40, 38], for which the discretized velocities $\mathbf{u}_h = (u_h, v_h)^T$, the discretized pressure p_h, and the discretized velocity curl, denoted q_h, live on different grids. Namely, we discretize Ω from (2.40) with four different tensor-product grids:

$$\begin{aligned} \Omega_h^u = \overline{\omega}_h \times \omega_h, \quad \Omega_h^v = \omega_h \times \overline{\omega}_h, \\ \Omega_h^p = \overline{\omega}_h \times \overline{\omega}_h, \quad \Omega_h^q = \omega_h \times \omega_h, \\ \Omega_h^* \subset \mathbb{Q}^2, \quad \text{for } * = u, v, p, q. \end{aligned} \tag{2.42}$$

where ω_h and $\overline{\omega}_h$ denote the aligned grid and the shifted (by $h/2$) grid, respectively, which are discretizations of the interval ω from (2.41), given as follows:

$$\omega_h = \frac{1}{\mathrm{n}}(1, \ldots, \mathrm{n}-1), \quad \overline{\omega}_h = \frac{1}{\mathrm{n}}(0, \ldots, \mathrm{n}-1) + \frac{1}{2\mathrm{n}}, \quad \omega_h, \overline{\omega}_h \subset \mathbb{Q}. \tag{2.43}$$

For illustration of the fully-staggered grids (2.42) for $\mathrm{n} = 4$, see Fig. 2.11. It is worth noting, that such fully-staggered discretization is known to be *structure-preserving* in the sense that many fundamental structures of the continuous model, e.g. mass and momentum conservation laws, are preserved at the discrete level. The term structure-preserving is typically used in the context of mimetic discretizations of the divergence and gradient operators, see, e.g., [65, 66, 67]. For earlier introduction of the mimetic approach see, e.g., [68].
Formally, we have the discrete functions belonging to different discrete spaces:

$$\begin{aligned} \mathcal{U}_h = \{u_h : \Omega_h^u \to \mathbb{R}\}, \quad \mathcal{V}_h = \{v_h : \Omega_h^v \to \mathbb{R}\}, \\ \mathcal{P}_h = \{p_h : \Omega_h^p \to \mathbb{R}\}, \quad \mathcal{Q}_h = \{q_h : \Omega_h^q \to \mathbb{R}\}. \end{aligned} \tag{2.44}$$

We identify these discrete spaces $\mathcal{U}_h$, $\mathcal{V}_h$, $\mathcal{P}_h$, and $\mathcal{Q}_h$ with the vector spaces $\mathbb{R}^{\mathrm{n}(\mathrm{n}-1)}$, $\mathbb{R}^{(\mathrm{n}-1)\mathrm{n}}$, $\mathbb{R}^{\mathrm{nn}}$, and $\mathbb{R}^{(\mathrm{n}-1)(\mathrm{n}-1)}$, respectively. Then, the underlying discrete operators can be assembled using the Kronecker matrix product (see, e.g., [69]). For example, the discrete identity operators $I^u : \mathcal{U}_h \to \mathcal{U}_h$, $I^v : \mathcal{V}_h \to \mathcal{V}_h$, $I^p : \mathcal{P}_h \to \mathcal{P}_h$, $I^q : \mathcal{Q}_h \to$

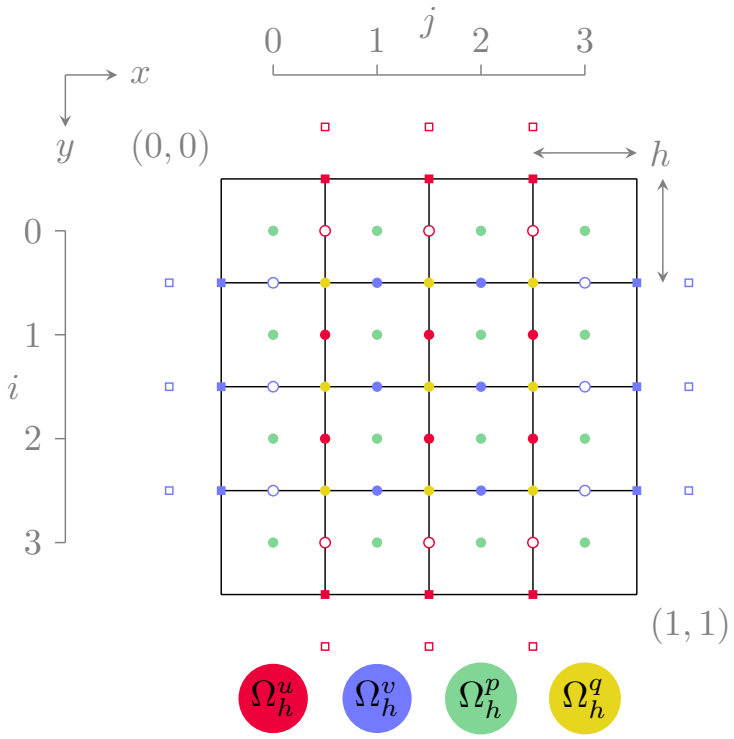


FIGURE 2.11: Fully-staggered grids for $\mathrm{n} = 4$. The systems from (2.48) are written for the interior nodes marked by circles. Bold circles denote the fully interior nodes; empty circles denote the near-boundary interior nodes with perturbed stencil. Solid squares denote the nodes lying on the boundary $\partial\Omega$ where the functions $\mathbf{u}_h^{\mathbf{D}}$ and $\mathbf{g}_h^{\mathbf{N}}$ are prescribed; empty squares denote the phantom nodes, which are used for discretization but eliminated using linear interpolation.

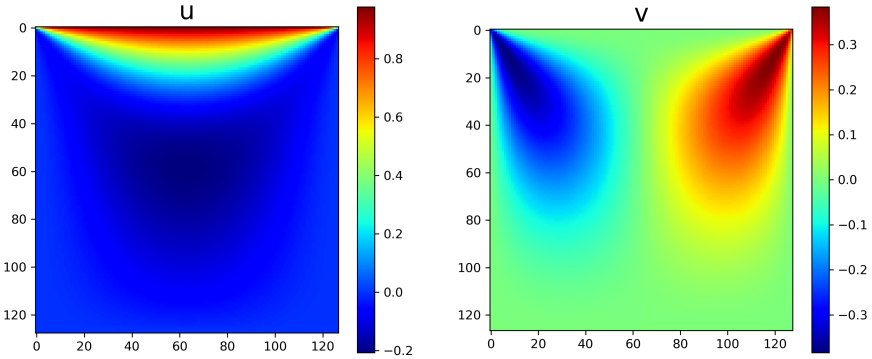


FIGURE 2.12: Lid-driven cavity problem. Velocity solution computed for $\mathrm{n} = 128$. The 50-lines Python code is provided in Section 2.C.

$\mathcal{Q}_h$ are constructed using the Kronecker product as follows:

$$\begin{aligned} I^u &= I^{\overline{\omega}_h} \otimes I^{\omega_h}, \quad I^u \in \mathbb{R}^{\mathrm{n}(\mathrm{n}-1)\times \mathrm{n}(\mathrm{n}-1)}, \\ I^v &= I^{\omega_h} \otimes I^{\overline{\omega}_h}, \quad I^v \in \mathbb{R}^{(\mathrm{n}-1)\mathrm{n}\times(\mathrm{n}-1)\mathrm{n}}, \\ I^p &= I^{\overline{\omega}_h} \otimes I^{\overline{\omega}_h}, \quad I^p \in \mathbb{R}^{\mathrm{nn}\times\mathrm{nn}}, \\ I^q &= I^{\omega_h} \otimes I^{\omega_h}, \quad I^q \in \mathbb{R}^{(\mathrm{n}-1)(\mathrm{n}-1)\times(\mathrm{n}-1)(\mathrm{n}-1)}, \end{aligned} \tag{2.45}$$

where I^{ω_h} and $I^{\overline{\omega}_h}$ are the identity matrices given as follows:

$$I^{\omega_h} = I^{\mathrm{n}-1} \in \mathbb{R}^{(\mathrm{n}-1)\times(\mathrm{n}-1)}, \quad I^{\overline{\omega}_h} = I^{\mathrm{n}} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}. \tag{2.46}$$

Note, the matrices I^{ω_h} and $I^{\overline{\omega}_h}$ correspond to the discrete identity operators acting on the one-dimensional functions discretized on the aligned grid ω_h and the shifted grid $\overline{\omega}_h$ from (2.43), respectively.
Finally, the velocity vector identity operator is composed as follows:

$$\mathbf{I}^{\mathbf{u}} = \begin{bmatrix} I^u & \\ & I^v \end{bmatrix}, \quad \mathbf{I}^{\mathbf{u}} : (\mathcal{U}_h, \mathcal{V}_h)^T \to (\mathcal{U}_h, \mathcal{V}_h)^T. \tag{2.47}$$

Remark 2.6.3. *Formally, in two dimensions, the discrete functions from* (2.44) *are represented as 2-dimensional tensors, and the corresponding operators are represented as 4-dimensional tensors. However, we work with matrices and vectors, assuming flattering of the 2-dimensional grid functions in the row-major order, which is a default order for python and C languages.*

Remark 2.6.4. *Note, the sparse Kronecker product is assumed in* (2.45) *and later in the text. See Section 2.C for python code example.*

Discretization of the primary Stokes BVP

After finite-difference discretization of the primary and auxiliary BVPs for the Stokes equation (2.34) using fully-staggered grids (2.42), we obtain the following block systems:

$$\mathbb{A}_{\mathbf{D}} \begin{bmatrix} \mathbf{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{u}}_h^{\mathbf{D}} \\ 0 \end{bmatrix}, \quad \mathbb{A}_{\mathbf{N}} = \begin{bmatrix} \mathbf{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{g}}_h^{\mathbf{N}} \\ 0 \end{bmatrix}, \tag{2.48}$$

where $\tilde{\mathbf{u}}_h^{\mathbf{D}}, \tilde{\mathbf{g}}_h^{\mathbf{N}} \in (\mathcal{U}_h, \mathcal{V}_h)^T$ are discretizations of the boundary terms $\mathbf{u}^{\mathbf{D}}, \mathbf{g}^{\mathbf{N}}$ from (2.36),(2.37), respectively. Note, the tilde sign here is used to emphasize that $\tilde{\mathbf{u}}_h^{\mathbf{D}}$ and $\tilde{\mathbf{g}}_h^{\mathbf{N}}$ are defined in the interior nodes, but not on the boundary nodes. Moreover, they are nonzero only on the near-boundary interior nodes, as we explain later in the text. The matrices $\mathbb{A}_{\mathbf{N}}$, $\mathbb{A}_{\mathbf{D}}$ have the following saddle-point structure:

$$\mathbb{A}_{\mathbf{N}} = \begin{bmatrix} \mathbf{A_N} & \mathbf{B}^T \\ \mathbf{B} & \end{bmatrix}, \quad \mathbb{A}_{\mathbf{D}} = \begin{bmatrix} \mathbf{A_D} & \mathbf{B}^T \\ \mathbf{B} & \end{bmatrix},$$

where the matrices $\mathbf{B} : (\mathcal{U}_h, \mathcal{V}_h)^T \to \mathcal{P}_h$ and $\mathbf{B}^T : \mathcal{P}_h \to (\mathcal{U}_h, \mathcal{V}_h)^T$ represent the discrete counterparts of the velocity (negative) divergence and the pressure gradient operator. The matrices $\mathbf{A_D}, \mathbf{A_N} : (\mathcal{U}_h, \mathcal{V}_h)^T \to (\mathcal{U}_h, \mathcal{V}_h)^T$ are the discrete counterparts of the velocity vector Laplacian operators corresponding to the Laplacian BVPs (2.38) and (2.39), respectively.
It should be noted, that the matrix $\mathbf{B}$ depends solely on the normal boundary condition (2.35), which makes it identical for both the primary and auxiliary BVPs. However, for the discrete Laplacian operators, we have $\mathbf{A} = \mathbf{A_D}$ or $\mathbf{A} = \mathbf{A_N}$ depending

on whether the condition (2.36) or (2.37) is imposed on the tangential velocity component.
Next, we describe how the boundary conditions (2.35), (2.36), and (2.37) are incorporated into the matrices. Lets consider the grids represented in Fig. 2.11. The systems for both primary and auxiliary BVPs from (2.48) are written for the interior nodes, marked by circles (both solid and empty), hence the matrices $\mathbb{A}_{\mathbf{D}}$ and $\mathbb{A}_{\mathbf{N}}$ have equal sizes. All the interior nodes are subdivided into two parts: the fully interior nodes, marked by solid circles, and the near-boundary interior nodes, marked by empty circles. Also, the solid squares denote the nodes lying precisely on the boundary $\partial\Omega$ where the functions $\mathbf{u}^{\mathbf{D}}$ and $\mathbf{g}^{\mathbf{N}}$ are prescribed. Finally, the empty squares denote the phantom nodes, which are used for discretization but eliminated using linear interpolation as discussed in the end of the Section.
Let us further unfold the discrete vector Laplacians as follows:

$$\mathbf{A_D} = \begin{bmatrix} A_D^u & \\ & A_D^v \end{bmatrix}, \quad A_D^u : \mathcal{U}_h \to \mathcal{U}_h, \quad A_D^v : \mathcal{V}_h \to \mathcal{V}_h,$$

and:

$$\mathbf{A_N} = \begin{bmatrix} A_N^u & \\ & A_N^v \end{bmatrix}, \quad A_N^u : \mathcal{U}_h \to \mathcal{U}_h, \quad A_N^v : \mathcal{V}_h \to \mathcal{V}_h.$$

For the fully interior nodes, all four discrete Laplacian operators $A_D^u, A_D^v, A_N^u, A_N^v$ have the full 5-point stencil, given as follows:

$$\frac{1}{h^2}\begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix} \tag{2.49}$$

However, for the near-boundary interior nodes, the stencil is perturbed. Namely, for the matrices A_D^u, A_N^u, the stencils for the interior nodes near to the top boundary (i.e. $y = 0$) become:

$$\frac{1}{h^2}\begin{pmatrix} -1 & 3 & -1 \\ & -1 & \end{pmatrix}, \quad \frac{1}{h^2}\begin{pmatrix} -1 & 5 & -1 \\ & -1 & \end{pmatrix},$$

respectively, and the stencils for the interior nodes near to the bottom boundary ($y = 1$) become:

$$\frac{1}{h^2}\begin{pmatrix} & -1 & \\ -1 & 3 & -1 \end{pmatrix}, \quad \frac{1}{h^2}\begin{pmatrix} & -1 & \\ -1 & 5 & -1 \end{pmatrix}.$$

Similarly, for the matrices A_D^v, A_N^v, the stencils for the interior nodes near to the left boundary ($x = 0$) become:

$$\frac{1}{h^2}\begin{pmatrix} -1 & \\ 3 & -1 \\ -1 & \end{pmatrix}, \quad \frac{1}{h^2}\begin{pmatrix} -1 & \\ 5 & -1 \\ -1 & \end{pmatrix},$$

respectively, and the stencils for the interior nodes near to the right boundary ($x = 1$) become:

$$\frac{1}{h^2}\begin{pmatrix} & -1 \\ -1 & 3 \\ & -1 \end{pmatrix}, \quad \frac{1}{h^2}\begin{pmatrix} & -1 \\ -1 & 5 \\ & -1 \end{pmatrix}.$$

According to these observations, the discrete Dirichlet velocity Laplacian operator $\mathbf{A_D}$ can be assembled as a diagonal perturbation of the discrete Neumann velocity Laplacian operator $\mathbf{A_N}$. Namely, we have:

Proposition 2.6.5.

$$\mathbf{A_D} = \mathbf{A_N} + \frac{2}{h^2}\mathbf{I}^{\mathbf{u}}_{\sim}, \tag{2.50}$$

where the diagonal perturbation matrix is given as follows:

$$\mathbf{I}^{\mathbf{u}}_{\sim} = \begin{bmatrix} I^{u}_{\sim} & \\ & I^{v}_{\sim} \end{bmatrix} = \begin{bmatrix} I^{\overline{\omega}_h}_{\sim} \otimes I^{\omega_h} & \\ & I^{\omega_h} \otimes I^{\overline{\omega}_h}_{\sim} \end{bmatrix}, \quad \mathbf{I}^{\mathbf{u}}_{\sim} : (\mathcal{U}_h, \mathcal{V}_h)^T \to (\mathcal{U}_h, \mathcal{V}_h)^T, \tag{2.51}$$

where I^{ω_h} is defined in (2.46)*, and $I^{\overline{\omega}_h}_{\sim}$ is defined as follows:*

$$I^{\overline{\omega}_h}_{\sim} = \begin{pmatrix} 1 & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & 1 \end{pmatrix}, \quad I^{\overline{\omega}_h}_{\sim} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}.$$

It should be noted, that the perturbation matrix $\mathbf{I}^{\mathbf{u}}_{\sim}$ has only $r = \mathrm{rank}(\mathbf{I}^{\mathbf{u}}_{\sim})$ non-zeros on the diagonal which corresponds to the number of near-boundary interior nodes (i.e. to the number of velocity nodes affected by the tangential boundary condition (2.36) imposed on $\partial\Omega$):

$$r = 4(\mathrm{n}-1) = \mathcal{O}(\mathrm{n}). \tag{2.52}$$

Note, the number of near-boundary interior nodes (empty circles), the number of boundary nodes (solid squares), and the number of phantom nodes (empty squares) are all equal to r. Then, the corresponding discrete space for the boundary data can be written as follows:

$$\Gamma_h = \mathbb{R}^r.$$

Let us further consider the following decomposition of the velocity perturbation matrix $\mathbf{I}^{\mathbf{u}}_{\sim}$:

$$\mathbf{I}^{\mathbf{u}}_{\sim} = \mathbf{U}^T\mathbf{U}, \tag{2.53}$$

where $\mathbf{U} \in \mathbb{R}^{r\times(\mathrm{n}(\mathrm{n}-1)+(\mathrm{n}-1)\mathrm{n})} : (\mathcal{U}_h, \mathcal{V}_h)^T \to \Gamma_h$ is the operator which extracts the near-boundary interior nodes, and $\mathbf{U}^T \in \mathbb{R}^{(\mathrm{n}(\mathrm{n}-1)+(\mathrm{n}-1)\mathrm{n})\times r} : \Gamma_h \to (\mathcal{U}_h, \mathcal{V}_h)^T$ is its (pseudo-)inverse, i.e. the operator which prolongates with zeros from the near-boundary interior nodes to all interior nodes.
Finally, let $\mathbf{u}^{\mathbf{D}}_h$, $\mathbf{g}^{\mathbf{N}}_h \in \Gamma_h$ denote the vectors obtained after discretization of the functions $\mathbf{u}^{\mathbf{D}}$, $\mathbf{g}^{\mathbf{N}}$ from (2.36),(2.37) on the boundary nodes (solid squares). Then, the right-hand-side vectors from (2.48) are given as follows:

$$\tilde{\mathbf{u}}^{\mathbf{D}}_h = \frac{2}{h^2}\mathbf{U}^T\mathbf{u}^{\mathbf{D}}_h, \quad \tilde{\mathbf{g}}^{\mathbf{N}}_h = \frac{1}{h}\mathbf{U}^T\mathbf{g}^{\mathbf{N}}_h.$$

The above relations can be obtained using full stencil (2.49) for the near-boundary interior nodes with subsequent elimination of the phantom variables. Namely, let $\hat{\mathbf{u}}^{\Gamma}_h \in \Gamma_h$ denotes the velocities defined on the phantom nodes (empty squares), and $\mathbf{u}^{\Gamma}_h \in \Gamma_h$ denotes the velocities defined on the near-boundary interior nodes (which are included into the systems). Then, in the case of the Dirichlet boundary condition (2.36), the phantom variables can be eliminated using linear interpolation as follows:

$$\frac{\tilde{\mathbf{u}}^{\Gamma}_h + \hat{\mathbf{u}}^{\Gamma}_h}{2} = \mathbf{u}^{\mathbf{D}}_h,$$

and in the case of the Neumann boundary condition (2.37), they can be eliminated as follows:

$$\frac{\tilde{\mathbf{u}}_h^\Gamma - \hat{\mathbf{u}}_h^\Gamma}{h} = \mathbf{g}_h^{\mathbf{N}},$$

Assembling operators $\mathbf{A_N}$ and B for the auxiliary Stokes BVP

Most of the Lemmas in the present subsection are well-known for mimetic discretizations in the case of infinite domains. We formulate their analogues for the auxiliary (enclosed Neumann) BVP. Note, the semi-staggered discretization using Kronecker product was written in [70, 71].
The discrete operators for the auxiliary problem have a simple structure such that the block matrices $\mathbf{A_N}$ and $\mathbf{B}$ from (2.3) can be assembled using the Kronecker matrix product from a single matrix B:

$$\mathrm{B} = \frac{1}{h}\begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{pmatrix}, \quad \mathrm{B} \in \mathbb{R}^{\mathrm{n}\times(\mathrm{n}-1)}. \tag{2.54}$$

Note, the matrix B can be considered as a discrete one-dimensional first-order derivative operator (from aligned grid to shifted) with zero Dirichlet boundary conditions.
Lets consider the first-order continuous differential operators (assuming infinite domain):

$$\begin{aligned} &\mathrm{div}(\mathbf{u}) = \nabla\cdot\mathbf{u} = u_x + v_y, \quad \mathrm{curl}(\mathbf{u}) = \nabla\times\mathbf{u} = u_y - v_x, \\ &\mathrm{grad}(p) = \nabla p = (p_x, p_y)^T, \quad \mathrm{curl}(q) = \nabla\times q = (-q_y, q_x)^T. \end{aligned} \tag{2.55}$$

In what follows, we describe how the discrete analogues of these continuous operators are assembled, including boundary conditions for the auxiliary BVP.
Firstly, the discrete co-directional derivatives of the velocity, denoted $\mathrm{B}_\mathrm{x}^u : \mathcal{U}_h \to \mathcal{P}_h$ and $\mathrm{B}_\mathrm{y}^v : \mathcal{V}_h \to \mathcal{P}_h$, are assembled as follows:

$$\begin{aligned} \mathrm{B}_\mathrm{x}^u &= I^{\overline{\omega}_h} \otimes \mathrm{B}, \quad \mathrm{B}_\mathrm{x}^u \in \mathbb{R}^{\mathrm{nn}\times\mathrm{n}(\mathrm{n}-1)}, \\ \mathrm{B}_\mathrm{y}^v &= \mathrm{B} \otimes I^{\overline{\omega}_h}, \quad \mathrm{B}_\mathrm{y}^v \in \mathbb{R}^{\mathrm{nn}\times(\mathrm{n}-1)\mathrm{n}}, \end{aligned}$$

and the discrete derivatives of the velocity curl, denoted $\mathrm{B}_\mathrm{x}^q : \mathcal{Q}_h \to \mathcal{V}_h$ and $\mathrm{B}_\mathrm{y}^q : \mathcal{Q}_h \to \mathcal{U}_h$, are assembled as follows:

$$\begin{aligned} \mathrm{B}_\mathrm{x}^q &= I^{\omega_h} \otimes \mathrm{B}, \quad \mathrm{B}_\mathrm{x}^q \in \mathbb{R}^{(\mathrm{n}-1)\mathrm{n}\times(\mathrm{n}-1)(\mathrm{n}-1)}, \\ \mathrm{B}_\mathrm{y}^q &= \mathrm{B} \otimes I^{\omega_h}, \quad \mathrm{B}_\mathrm{y}^q \in \mathbb{R}^{\mathrm{n}(\mathrm{n}-1)\times(\mathrm{n}-1)(\mathrm{n}-1)}. \end{aligned}$$

Then, the discrete (negative) divergence of the velocity is composed as follows:

$$\mathbf{B} = \begin{bmatrix} -\mathrm{B}_\mathrm{x}^u & -\mathrm{B}_\mathrm{y}^v \end{bmatrix}, \tag{2.56}$$

and the discrete curl of the velocity curl is composed as follows:

$$\mathbf{C}^T = \begin{bmatrix} -\mathrm{B}_\mathrm{y}^q \\ \mathrm{B}_\mathrm{x}^q \end{bmatrix}, \quad \mathbf{C}^T : \mathcal{Q}_h \to (\mathcal{U}_h, \mathcal{V}_h)^T. \tag{2.57}$$

Note, the boundary condition $\mathrm{curl}(\mathbf{u}) = 0$ on $\partial\Omega$ is included in $\mathbf{C}^T$, which is naturally satisfied for the enclosed Neumann problem. Next, for the discrete gradient of the pressure, we have:

$$\begin{bmatrix} \mathrm{B}_{\mathrm{x}}^{p} \\ \mathrm{B}_{\mathrm{y}}^{p} \end{bmatrix} = \begin{bmatrix} -(\mathrm{B}_{\mathrm{x}}^{u})^T \\ -(\mathrm{B}_{\mathrm{y}}^{v})^T \end{bmatrix} = \mathbf{B}^T, \tag{2.58}$$

and for the discrete curl of the velocity, we have:

$$\begin{bmatrix} \mathrm{B}_{\mathrm{y}}^{u} & -\mathrm{B}_{\mathrm{x}}^{v} \end{bmatrix} = \begin{bmatrix} -(\mathrm{B}_{\mathrm{y}}^{q})^T & (\mathrm{B}_{\mathrm{x}}^{q})^T \end{bmatrix} = \mathbf{C}^{T^T} = \mathbf{C}, \quad \mathbf{C} : (\mathcal{U}_h, \mathcal{V}_h)^T \to \mathcal{Q}_h. \tag{2.59}$$

It is worth noting, that the relation (2.58) reflects the fact that the pressure gradient operator, by definition, is conjugate to the negative divergence velocity operator. Thus, the following property is always satisfied:

Lemma 2.6.6.

$$\begin{bmatrix} \mathrm{B}_{\mathrm{x}}^{u} & \mathrm{B}_{\mathrm{y}}^{v} \end{bmatrix} \begin{bmatrix} -\mathrm{B}_{\mathrm{y}}^{q} \\ \mathrm{B}_{\mathrm{x}}^{q} \end{bmatrix} = 0, \tag{2.60}$$

which mimics the continuous property for the infinite domain:

$$\nabla \cdot (\nabla \times q) = 0.$$

Proof.

$$\begin{aligned} \mathbf{B}\mathbf{C}^T = &- \mathrm{B}_{\mathrm{x}}^{u}\mathrm{B}_{\mathrm{y}}^{q} + \mathrm{B}_{\mathrm{y}}^{v}\mathrm{B}_{\mathrm{x}}^{q} = \\ &- (I^{\overline{\omega}_h} \otimes \mathrm{B})(\mathrm{B} \otimes I^{\omega_h}) + (\mathrm{B} \otimes I^{\overline{\omega}_h})(I^{\omega_h} \otimes \mathrm{B}) = \\ &- \mathrm{B} \otimes \mathrm{B} + \mathrm{B} \otimes \mathrm{B} = 0. \end{aligned}$$

□

However, the relation (2.59) is not valid in general, but only for certain "energy-conserving" formulations. In fact, it is the fulfillment of the relation (2.59) that entails the discrete Helmholtz-Hodge decomposition for the enclosed Neumann problem, which is discussed in Section 2.6.3. So, the following relation is satisfied:

Lemma 2.6.7.

$$\begin{bmatrix} \mathrm{B}_{\mathrm{y}}^{u} & -\mathrm{B}_{\mathrm{x}}^{v} \end{bmatrix} \begin{bmatrix} \mathrm{B}_{\mathrm{x}}^{p} \\ \mathrm{B}_{\mathrm{y}}^{p} \end{bmatrix} = 0, \tag{2.61}$$

which mimics the continuous property for the infinite domain:

$$\nabla \times (\nabla p) = 0.$$

Proof.

$$\mathbf{C}\mathbf{B}^T = (\mathbf{B}\mathbf{C}^T)^T = 0^T = 0.$$

□

We also state the following equalities:

Lemma 2.6.8.

$$\mathrm{B}_{\mathrm{y}}^{p}\mathrm{B}_{\mathrm{x}}^{u} = \mathrm{B}_{\mathrm{x}}^{q}\mathrm{B}_{\mathrm{y}}^{u}, \quad \mathrm{B}_{\mathrm{x}}^{p}\mathrm{B}_{\mathrm{y}}^{v} = \mathrm{B}_{\mathrm{y}}^{q}\mathrm{B}_{\mathrm{x}}^{v}.$$

which mimics the continuous relations for the infinite domain:

$$u_{xy} = u_{yx}, \quad v_{yx} = v_{xy}.$$

Proof.

$$-\mathrm{B}_{\mathrm{y}}^{p}\mathrm{B}_{\mathrm{x}}^{u} = \mathrm{B}_{\mathrm{y}}^{v\,T}\mathrm{B}_{\mathrm{x}}^{u} = (\mathrm{B}\otimes I^{\overline{\omega}_h})^T(I^{\overline{\omega}_h}\otimes\mathrm{B}) =$$
$$= (\mathrm{B}^T\otimes I^{\overline{\omega}_h})(I^{\overline{\omega}_h}\otimes\mathrm{B}) = \mathrm{B}^T\otimes\mathrm{B} = (I^{\omega_h}\otimes\mathrm{B})(\mathrm{B}^T\otimes I^{\omega_h}) =$$
$$= (I^{\omega_h}\otimes\mathrm{B})(\mathrm{B}\otimes I^{\omega_h})^T = \mathrm{B}_{\mathrm{x}}^{q}\mathrm{B}_{\mathrm{y}}^{q\,T} = -\mathrm{B}_{\mathrm{x}}^{q}\mathrm{B}_{\mathrm{y}}^{u}.$$

□

The discrete velocity Laplacian operator $\mathbf{A_N}$ is assembled as follows:

Proposition 2.6.9.

$$\mathbf{A_N} \;=\; \mathbf{B}^T\mathbf{B} \;+\; \mathbf{C}^T\mathbf{C}. \tag{2.62}$$

which mimics the continuous identity for the infinite domain:

$$-\Delta \;=\; -\nabla\nabla\cdot \;+\; \nabla\times\nabla\times .$$

Proof.

$$\mathbf{B}^T\mathbf{B}+\mathbf{C}^T\mathbf{C} =$$
$$= \begin{bmatrix} -\mathrm{B}_{\mathrm{x}}^{p}\mathrm{B}_{\mathrm{x}}^{u} & -\mathrm{B}_{\mathrm{x}}^{p}\mathrm{B}_{\mathrm{y}}^{v} \\ -\mathrm{B}_{\mathrm{y}}^{p}\mathrm{B}_{\mathrm{x}}^{u} & -\mathrm{B}_{\mathrm{y}}^{p}\mathrm{B}_{\mathrm{y}}^{v} \end{bmatrix} + \begin{bmatrix} -\mathrm{B}_{\mathrm{y}}^{q}\mathrm{B}_{\mathrm{y}}^{u} & \mathrm{B}_{\mathrm{y}}^{q}\mathrm{B}_{\mathrm{x}}^{v} \\ \mathrm{B}_{\mathrm{x}}^{q}\mathrm{B}_{\mathrm{y}}^{u} & -\mathrm{B}_{\mathrm{x}}^{q}\mathrm{B}_{\mathrm{x}}^{v} \end{bmatrix} =$$
$$= \begin{bmatrix} -\mathrm{B}_{\mathrm{x}}^{p}\mathrm{B}_{\mathrm{x}}^{u}-\mathrm{B}_{\mathrm{y}}^{q}\mathrm{B}_{\mathrm{y}}^{u} & \\ & -\mathrm{B}_{\mathrm{x}}^{q}\mathrm{B}_{\mathrm{x}}^{v}-\mathrm{B}_{\mathrm{y}}^{p}\mathrm{B}_{\mathrm{y}}^{v} \end{bmatrix} = \begin{bmatrix} \mathrm{A}_{N}^{u} & \\ & \mathrm{A}_{N}^{v} \end{bmatrix} = \mathbf{A_N}.$$

□

2.6.3 Structure of the Schur complement matrix

In the present Section, we investigate the structure of the Schur complement matrices:

$$S_N = \mathbf{B}\mathbf{A}_{\mathbf{N}}^{-1}\mathbf{B}^T, \quad S_D = \mathbf{B}\mathbf{A}_{\mathbf{D}}^{-1}\mathbf{B}^T, \tag{2.63}$$

corresponding to the velocity Laplacian matrices $\mathbf{A_N}$, $\mathbf{A_D}$ defined in (2.62), (2.50) for the enclosed Dirichlet and Neumann Laplacian BVPs formulated in Section 2.6.1, respectively.

Discrete Helmoltz-Hodge Decomposition (DHHD)

The Helmholz-Hodge decomposition states that any vector field can be uniquely represented as the sum of a non-divergent field and a non-rotating field. It turns out, that for the discrete divergence matrix $\mathbf{B}$ from (2.56) and the discrete curl matrix $\mathbf{C}$ from (2.57), the discrete analogue of the Helmholtz-Hodge decomposition is preserved, which can be formulated as follows:

Theorem 2.6.10.

$$(\mathcal{U}_h,\mathcal{V}_h)^T = \mathrm{Ker}\mathbf{B}\oplus\mathrm{Ker}\mathbf{C}, \tag{2.64}$$

and

$$r_B = \dim(\mathrm{Ker}\mathbf{B}) = \mathrm{n}^2-1, \quad r_C = \dim(\mathrm{Ker}\mathbf{C}) = (\mathrm{n}-1)^2. \tag{2.65}$$

Proof. Firstly, from Lemma 2.6.6 or 2.6.7, we have:

$$\mathrm{Im}\mathbf{C}^T \subset \mathrm{Ker}\mathbf{B},$$

which implies:

$$(\mathrm{Ker}\mathbf{C})^{\perp} \subset \mathrm{Ker}\mathbf{B}, \tag{2.66}$$

since $(\mathrm{Ker}\mathbf{C})^{\perp} = \mathrm{Im}\mathbf{C}^T$. Secondly, let us take $\mathbf{x} \in (\mathcal{U}_h, \mathcal{V}_h)^T$ such that $\mathbf{x} \in \mathrm{Ker}\mathbf{B} \cap \mathrm{Ker}\mathbf{C}$, then we have $\mathbf{x} \in \mathrm{Ker}(\mathbf{B}^T\mathbf{B}) \cap \mathrm{Ker}(\mathbf{C}^T\mathbf{C})$. Thus, using the representation (2.62), we have:

$$(\mathbf{B}^T\mathbf{B})\mathbf{x} + (\mathbf{C}^T\mathbf{C})\mathbf{x} = \mathbf{A_N}\mathbf{x} = \mathbf{0},$$

which implies that $\mathbf{x} = \mathbf{0}$, since the matrix $\mathbf{A_N}$ has full rank. Therefore, we have shown that $\mathrm{Ker}\mathbf{B} \cap \mathrm{Ker}\mathbf{C} = \mathbf{0}$, which completes the proof of the decomposition (2.64), taking into account the relation (2.66). Next, in order to prove the dimension relations (2.65), we note that the fully-staggered finite-difference discretization of the pressure gradient operator $\mathbf{B}^T$ defined in (2.58) has a one-dimensional nullspace spanned by the constant vector $\mathbf{1}$, which is defined as follows:

$$\mathbf{1} = h(1, \ldots, 1)^T \in \mathcal{P}_h, \quad |\mathbf{1}|_2 = 1. \tag{2.67}$$

It is also worth mentioning that $r_B = \mathrm{rank}(\mathbf{B}^T\mathbf{B})$ and $r_C = \mathrm{rank}(\mathbf{C}^T\mathbf{C})$. □

Several important conclusions can be drawn from Theorem 2.6.10. The most important one (see Corollary 2.6.10.1) is that the discrete Neumann velocity Laplacian operator $\mathbf{A_N}$ expands into the direct sum with respect to the discrete Helmholtz-Hodge decomposition (2.64). In what follows, we consider the Singular Value Decomposition (SVD) of the underlying operators.
Lets consider the reduced SVD decomposition of the matrix $\mathbf{B}$:

$$\mathbf{B} = \mathrm{V}_B \Sigma_B \mathbf{U}_B^T,$$

with matrices with orthogonal columns:

$$\mathrm{V}_B = \begin{bmatrix} p_B^1 & p_B^2 & \ldots & p_B^{r_B} \end{bmatrix}, \quad \mathbf{U}_B = \begin{bmatrix} \mathbf{u}_B^1 & \mathbf{u}_B^2 & \ldots & \mathbf{u}_B^{r_B} \end{bmatrix},$$

and the reduced SVD decomposition of the matrix $\mathbf{C}$:

$$\mathbf{C} = \mathrm{V}_C \Sigma_C \mathbf{U}_C^T,$$

with matrices with orthogonal columns:

$$\mathrm{V}_C = \begin{bmatrix} q_C^1 & q_C^2 & \ldots & q_C^{r_C} \end{bmatrix}, \quad \mathbf{U}_C = \begin{bmatrix} \mathbf{u}_C^1 & \mathbf{u}_C^2 & \ldots & \mathbf{u}_C^{r_C} \end{bmatrix},$$

where r_B and r_C are the ranks given in (2.65). Then, we have:

$$(\mathrm{Ker}\mathbf{B})^{\perp} = \mathrm{Im}\mathbf{B}^T = \mathcal{L}(\mathbf{u}_B^1, \mathbf{u}_B^2, \ldots, \mathbf{u}_B^{r_B}),$$

$$(\mathrm{Ker}\mathbf{C})^{\perp} = \mathrm{Im}\mathbf{C}^T = \mathcal{L}(\mathbf{u}_C^1, \mathbf{u}_C^2, \ldots, \mathbf{u}_C^{r_C}).$$

From the equation (2.64), we also have $(\mathrm{Ker}\mathbf{C})^{\perp} = \mathrm{Ker}\mathbf{B}$ and $(\mathrm{Ker}\mathbf{B})^{\perp} = \mathrm{Ker}\mathbf{C}$, so it can be rewritten as follows:

$$(\mathcal{U}^h, \mathcal{V}^h) = \mathcal{L}(\mathbf{u}_C^1, \mathbf{u}_C^2, \ldots, \mathbf{u}_C^{r_C}) \oplus \mathcal{L}(\mathbf{u}_B^1, \mathbf{u}_B^2, \ldots, \mathbf{u}_B^{r_B}).$$

The Moore-Penrose pseudo-inverses of $\mathbf{B}$ and $\mathbf{C}$ are defined as follows using SVD decompositions:

$$\mathbf{C}^{\dagger} = \mathbf{U}_C \Sigma_C^{-1} \mathrm{V}_C^T,$$

$$\mathbf{B}^{\dagger} = \mathbf{U}_B \Sigma_B^{-1} \mathrm{V}_B^T.$$

We also define:

$$\mathcal{P}_{\mathrm{Ker}\mathbf{B}} = \mathbf{C}^{\dagger}\mathbf{C} = \mathbf{U}_C(\Sigma_C^{-1}\mathrm{V}_C^T\mathrm{V}_C\Sigma_C)\mathbf{U}_C^T = \mathbf{U}_C\mathbf{U}_C^T, \tag{2.68}$$

$$\mathcal{P}_{\mathrm{Ker}\mathbf{C}} = \mathbf{B}^{\dagger}\mathbf{B} = \mathbf{U}_B(\Sigma_B^{-1}\mathrm{V}_B^T\mathrm{V}_B\Sigma_B)\mathbf{U}_B^T = \mathbf{U}_B\mathbf{U}_B^T, \tag{2.69}$$

which are orthogonal projectors:

$$\mathcal{P}^2_{\mathrm{Ker}\mathbf{B}} = \mathcal{P}_{\mathrm{Ker}\mathbf{B}} = \mathcal{P}^T_{\mathrm{Ker}\mathbf{B}},$$
$$\mathcal{P}^2_{\mathrm{Ker}\mathbf{C}} = \mathcal{P}_{\mathrm{Ker}\mathbf{C}} = \mathcal{P}^T_{\mathrm{Ker}\mathbf{C}}.$$

From (2.64), we have:

$$\begin{gathered}\mathcal{P}_{\mathrm{Ker}\mathbf{B}}\mathcal{P}_{\mathrm{Ker}\mathbf{C}} = \mathcal{P}_{\mathrm{Ker}\mathbf{C}}\mathcal{P}_{\mathrm{Ker}\mathbf{B}} = 0,\\ \mathcal{P}_{\mathrm{Ker}\mathbf{B}} + \mathcal{P}_{\mathrm{Ker}\mathbf{C}} = \mathbf{I}^{\mathbf{u}}.\end{gathered} \tag{2.70}$$

Also, swapping the matrices and their pseudo-inverses, we have:

$$\mathbf{C}\mathbf{C}^{\dagger} = I^q, \quad \mathbf{B}\mathbf{B}^{\dagger} = I^p - (\mathbf{1}\cdot\mathbf{1}^T). \tag{2.71}$$

Corollary 2.6.10.1. *The discrete Neumann velocity Laplacian operator* $\mathbf{A_N}$ *defined in* (2.62) *expands into direct sum with respect to the Helmholtz-Hodge decomposition* (2.64)*, namely:*

$$\mathbf{A_N} = (\mathcal{P}_{\mathrm{Ker}\mathbf{C}}\mathbf{A_N}\mathcal{P}_{\mathrm{Ker}\mathbf{C}}) + (\mathcal{P}_{\mathrm{Ker}\mathbf{B}}\mathbf{A_N}\mathcal{P}_{\mathrm{Ker}\mathbf{B}}), \tag{2.72}$$

and for the inverse, we have:

$$\mathbf{A}_{\mathbf{N}}^{-1} = (\mathcal{P}_{\mathrm{Ker}\mathbf{C}}\mathbf{A_N}\mathcal{P}_{\mathrm{Ker}\mathbf{C}})^{\dagger} + (\mathcal{P}_{\mathrm{Ker}\mathbf{B}}\mathbf{A_N}\mathcal{P}_{\mathrm{Ker}\mathbf{B}})^{\dagger}. \tag{2.73}$$

Proof. First, for the matrices $\mathbf{B}^T\mathbf{B}$ and $\mathbf{C}^T\mathbf{C}$, we have the following SVD decompositions:

$$\mathbf{B}^T\mathbf{B} = \mathbf{U}_B\Sigma_B(\mathrm{V}_B^T\mathrm{V}_B)\Sigma_B\mathbf{U}_B^T = \mathbf{U}_B\Sigma_B^2\mathbf{U}_B^T,$$
$$\mathbf{C}^T\mathbf{C} = \mathbf{U}_C\Sigma_C(\mathrm{V}_C^T\mathrm{V}_C)\Sigma_C\mathbf{U}_C^T = \mathbf{U}_C\Sigma_C^2\mathbf{U}_C^T.$$

For the second term of (2.72), we have:

$$\begin{gathered}\mathcal{P}_{\mathrm{Ker}\mathbf{B}}\mathbf{A_N}\mathcal{P}_{\mathrm{Ker}\mathbf{B}} =\\ = \mathbf{U}_C\mathbf{U}_C^T(\mathbf{U}_B\Sigma_B^2\mathbf{U}_B^T + \mathbf{U}_C\Sigma_C^2\mathbf{U}_C^T)\mathbf{U}_C\mathbf{U}_C^T =\\ = \mathbf{U}_C(\mathbf{U}_C^T\mathbf{U}_B)\Sigma_B^2(\mathbf{U}_B^T\mathbf{U}_C)\mathbf{U}_C^T + \mathbf{U}_C(\mathbf{U}_C^T\mathbf{U}_C)\Sigma_C^2(\mathbf{U}_C^T\mathbf{U}_C)\mathbf{U}_C^T =\\ = \mathbf{U}_C(0)\Sigma_B^2(0)\mathbf{U}_C^T + \mathbf{U}_C(I)\Sigma_C^2(I)\mathbf{U}_C^T =\\ = \mathbf{C}^T\mathbf{C}.\end{gathered}$$

Similarly, we have for the first term of (2.72):

$$\mathcal{P}_{\mathrm{Ker}\mathbf{C}}\mathbf{A_N}\mathcal{P}_{\mathrm{Ker}\mathbf{C}} = \mathbf{B}^T\mathbf{B},$$

which proves the first part of theorem.

For the inverse, we have:

$$\begin{aligned}
\mathbf{A}_\mathbf{N}^{-1} &= (\mathbf{B}^T\mathbf{B} + \mathbf{C}^T\mathbf{C})^{-1} = \\
&= (\mathbf{U}_B\Sigma_B^2\mathbf{U}_B^T + \mathbf{U}_C\Sigma_C^2\mathbf{U}_C^T)^{-1} = \\
= (\begin{bmatrix}\mathbf{U}_B & \mathbf{U}_C\end{bmatrix}\begin{bmatrix}\Sigma_B^2 & \\ & 0\end{bmatrix}\begin{bmatrix}\mathbf{U}_B^T \\ \mathbf{U}_C^T\end{bmatrix} &+ \begin{bmatrix}\mathbf{U}_B & \mathbf{U}_C\end{bmatrix}\begin{bmatrix}0 & \\ & \Sigma_C^2\end{bmatrix}\begin{bmatrix}\mathbf{U}_B^T \\ \mathbf{U}_C^T\end{bmatrix})^{-1} = \\
&= (\begin{bmatrix}\mathbf{U}_B & \mathbf{U}_C\end{bmatrix}\begin{bmatrix}\Sigma_B^2 & \\ & \Sigma_C^2\end{bmatrix}\begin{bmatrix}\mathbf{U}_B^T \\ \mathbf{U}_C^T\end{bmatrix})^{-1} = \\
&= \begin{bmatrix}\mathbf{U}_B & \mathbf{U}_C\end{bmatrix}\begin{bmatrix}\Sigma_B^{-2} & \\ & \Sigma_C^{-2}\end{bmatrix}\begin{bmatrix}\mathbf{U}_B^T \\ \mathbf{U}_C^T\end{bmatrix} = \\
= \begin{bmatrix}\mathbf{U}_B & \mathbf{U}_C\end{bmatrix}\begin{bmatrix}\Sigma_B^{-2} & \\ & 0\end{bmatrix}\begin{bmatrix}\mathbf{U}_B^T \\ \mathbf{U}_C^T\end{bmatrix} &+ \begin{bmatrix}\mathbf{U}_B & \mathbf{U}_C\end{bmatrix}\begin{bmatrix}0 & \\ & \Sigma_C^{-2}\end{bmatrix}\begin{bmatrix}\mathbf{U}_B^T \\ \mathbf{U}_C^T\end{bmatrix} = \\
&= \mathbf{U}_B\Sigma_B^{-2}\mathbf{U}_B^T + \mathbf{U}_C\Sigma_C^{-2}\mathbf{U}_C^T = \\
&= (\mathbf{B}^T\mathbf{B})^\dagger + (\mathbf{C}^T\mathbf{C})^\dagger,
\end{aligned}$$

which completes the proof. □

The following block matrix is of special interest:

$$\mathbf{Q} = \begin{bmatrix}\mathbf{B} \\ \mathbf{C}\end{bmatrix}, \quad \mathbf{Q} : (\mathcal{U}_h, \mathcal{V}_h)^T \to (\mathcal{P}_h, \mathcal{Q}_h)^T. \tag{2.74}$$

The matrix $\mathbf{Q} \in \mathbb{R}^{\mathrm{N}+1\times\mathrm{N}}$ is nearly square matrix of the size N if we denote $\mathrm{N} = (\mathrm{n}-1)\mathrm{n}+\mathrm{n}(\mathrm{n}-1) = 2\mathrm{n}(\mathrm{n}-1)$ the number of velocity nodes. Indeed, for the dimension of the divergency-vorticity space $(\mathcal{P}_h, \mathcal{Q}_h)^T$, we have $n^2+(n-1)^2 = 2n(n-1)+1 = \mathrm{N}+1$. Moreover, from Theorem 2.6.10, the matrix $\mathbf{Q}$ is non-singular in the sense that it has full column rank. Using this matrix $\mathbf{Q}$, for the Dirichlet and Neumann Laplacians, defined in (2.50) and (2.62), we have the following decomposition:

$$\mathbf{A_N} = \mathbf{Q}^T\mathbf{Q}, \quad \mathbf{A_D} = \mathbf{Q}^T\mathbf{Q} + \frac{2}{h^2}\mathbf{I}^\mathbf{u}_\sim. \tag{2.75}$$

Note, the decomposition (2.75) leads to a simple way to show positive definitness of the discrete vector Laplacians. Namely, for all $\mathbf{u}_h \in (\mathcal{U}_h, \mathcal{V}_h)^T$, $\mathbf{u}_h \neq 0$, we have:

$$\begin{aligned}
(\mathbf{A_N}\mathbf{u}_h, \mathbf{u}_h) &= (\mathbf{Q}^T\mathbf{Q}\mathbf{u}_h, \mathbf{u}_h) = (\mathbf{Q}\mathbf{u}_h, \mathbf{Q}\mathbf{u}_h) > 0, \\
(\mathbf{A_D}\mathbf{u}_h, \mathbf{u}_h) &= (\mathbf{Q}^T\mathbf{Q}\mathbf{u}_h, \mathbf{u}_h) + \frac{2}{h^2}(\mathbf{I}^\mathbf{u}_\sim\mathbf{u}_h, \mathbf{u}_h) = \\
&= (\mathbf{Q}\mathbf{u}_h, \mathbf{Q}\mathbf{u}_h) + \frac{2}{h^2}(\mathbf{I}^\mathbf{u}_\sim\mathbf{u}_h, \mathbf{u}_h) > 0.
\end{aligned}$$

Structure of S_N

In this subsection, we demonstrate that the Neumann Schur complement matrix S_N from (2.63) is reduced to the pressure identity operator I^p defined in (2.45), up to a one-dimensional constant nullspace.

Corollary 2.6.10.2. *For the Neumann Schur complement matrix, we have the following explicit representation:*

$$S_N = I^p - (\mathbf{1}\cdot\mathbf{1}^T), \tag{2.76}$$

where $\mathbf{1} \in \mathcal{P}^h$ is the pressure constant defined in (2.67).

Proof. Using decomposition (2.73), we have:

$$\begin{aligned} S_N = \mathbf{B}\mathbf{A}_{\mathbf{N}}^{-1}\mathbf{B}^T = \mathbf{B}((\mathbf{B}^T\mathbf{B})^\dagger + (\mathbf{C}^T\mathbf{C})^\dagger)\mathbf{B}^T = \\ = \mathbf{B}(\mathbf{B}^T\mathbf{B})^\dagger\mathbf{B}^T = \mathbf{B}\mathbf{B}^\dagger(\mathbf{B}\mathbf{B}^\dagger)^T = \\ = (\mathbf{B}\mathbf{B}^\dagger)^2 = \mathbf{B}\mathbf{B}^\dagger, \end{aligned}$$

where the resulting matrix $\mathbf{B}\mathbf{B}^\dagger$ is, by definition (2.71), an orthogonal projector onto the kernel of the discrete gradient operator $\mathbf{B}^T$, which completes the proof. Alternative proof using SVD can be written as follows:

$$\begin{aligned} \mathbf{B}\mathbf{A}_{\mathbf{N}}^{-1}\mathbf{B}^T = \mathbf{B}((\mathbf{B}^T\mathbf{B})^\dagger + (\mathbf{C}^T\mathbf{C})^\dagger)\mathbf{B}^T = \\ = \mathrm{V}_B\Sigma_B\mathbf{U}_B^T(\mathbf{U}_B\Sigma_B^{-2}\mathbf{U}_B^T + \mathbf{U}_C\Sigma_C^{-2}\mathbf{U}_C^T)\mathbf{U}_B\Sigma_B\mathbf{V}_B^T = \\ = \mathrm{V}_B\Sigma_B(\mathbf{U}_B^T\mathbf{U}_B)\Sigma_B^{-2}(\mathbf{U}_B^T\mathbf{U}_B)\Sigma_B\mathbf{V}_B^T + \\ + \mathrm{V}_B\Sigma_B(\mathbf{U}_B^T\mathbf{U}_C)\Sigma_B^{-2}(\mathbf{U}_C^T\mathbf{U}_B)\Sigma_B\mathbf{V}_B^T = \\ = \mathrm{V}_B\mathrm{V}_B^T = I^p - (\mathbf{1}\cdot\mathbf{1}^T). \end{aligned}$$

□

Finally, for the inverse of S_N, by definition, we have:

Corollary 2.6.10.3.

$$S_N^\dagger = S_N, \quad S_N^\dagger : \mathcal{P}_h \to \mathcal{P}_h. \tag{2.77}$$

Thus, it has been shown that the Neumann Schur complement matrix S_N, as well as its inverse $S_N^\dagger$, equals the identity operator I^p acting on the discrete pressure space $\mathcal{P}_h$, up to a one-dimensional kernel of the discrete pressure gradient operator $\mathbf{B}^T$.

Structure of S_D

The matrix $S_D = \mathbf{B}\mathbf{A}_{\mathbf{D}}^{-1}\mathbf{B}^T$, as well as its inverse $S_D^\dagger$, can be written as rank-r perturbations of the identity matrices S_N and $S_N^\dagger$ defined in (2.76) and (2.77), respectively:

Theorem 2.6.11.

$$S_D = S_N - (\mathbf{U}\mathbf{B}^\dagger)^T(K_1)^{-1}(\mathbf{U}\mathbf{B}^\dagger), \tag{2.78a}$$

$$S_D^\dagger = S_N^\dagger + (\mathbf{U}\mathbf{B}^\dagger)^T(K_2)^{-1}(\mathbf{U}\mathbf{B}^\dagger), \tag{2.78b}$$

where r-dimensional kernels $K_1, K_2 \in \mathbb{R}^{r\times r}$ are defined as follows:

$$\begin{aligned} K_1 &= \mathbf{U}(\frac{h^2}{2}\mathbf{I}^{\mathbf{u}} + (\mathbf{B}^T\mathbf{B})^\dagger + (\mathbf{C}^T\mathbf{C})^\dagger)\mathbf{U}^T, \\ K_2 &= \mathbf{U}(\frac{h^2}{2}\mathbf{I}^{\mathbf{u}} + (\mathbf{C}^T\mathbf{C})^\dagger)\mathbf{U}^T. \end{aligned}$$

Proof. For the proof we use Sherman–Morrison–Woodbury formula [72], which states that the inverse of a rank-r correction of some matrix can be computed by doing a rank-r correction to the inverse of the original matrix. Let $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{r\times r}$, $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{r\times n}$, then for the inverse of A, we have:

$$(A + U^TCU)^{-1} = A^{-1} - A^{-1}U^T(C^{-1} + UA^{-1}U^T)^{-1}UA^{-1}. \tag{2.79}$$

Taking into account the representation formula (2.50) and the decomposition (2.53), the Woodbury formula for the inverse of $\mathbf{A_D}$ gives:

$$\begin{aligned}\mathbf{A_D^{-1}} &= \mathbf{A_N^{-1}} - \mathbf{A_N^{-1}}\mathbf{U}^T(\frac{h^2}{2}I^r + \mathbf{U}\mathbf{A_N}\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{A_N^{-1}} = \\ &= \mathbf{A_N^{-1}} - (\mathbf{U}\mathbf{A_N^{-1}})^T(K_1)^{-1}(\mathbf{U}\mathbf{A_N^{-1}}),\end{aligned}$$

where I^r is identity matrix of size $r \times r$. Then, for the Dirichlet Schur complement, we have:

$$S_D = \mathbf{B}\mathbf{A_D^{-1}}\mathbf{B}^T = \mathbf{B}\mathbf{A_N^{-1}}\mathbf{B}^T - (\mathbf{U}\mathbf{A_N^{-1}}\mathbf{B}^T)^T(K_1)^{-1}(\mathbf{U}\mathbf{A_N^{-1}}\mathbf{B}^T),$$

which completes the proof of (2.78a), taking into account that:

$$\mathbf{A_N^{-1}}\mathbf{B}^T = ((\mathbf{B}^T\mathbf{B})^\dagger + (\mathbf{C}^T\mathbf{C})^\dagger)\mathbf{B}^T = (\mathbf{B}^T\mathbf{B})^\dagger\mathbf{B}^T = \mathbf{B}^\dagger.$$

In order to proof (2.78b), we exploit the Woodbury formula once again for the inverse of S_D under the representation (2.78a):

$$\begin{aligned}&S_D^\dagger - S_N^\dagger = \\ &= (S_N^\dagger)(\mathbf{U}\mathbf{B}^\dagger)^T[((K_1)^{-1})^{-1} + (\mathbf{U}\mathbf{B}^\dagger)(S_N^\dagger)(\mathbf{U}\mathbf{B}^\dagger)^T]^{-1}(\mathbf{U}\mathbf{B}^\dagger)(S_N^\dagger) \\ &= (\mathbf{U}\mathbf{B}^\dagger)^T[\mathbf{U}(\frac{h^2}{2}\mathbf{I^u} + (\mathbf{B}^T\mathbf{B})^\dagger + (\mathbf{C}^T\mathbf{C})^\dagger - (\mathbf{B}^T\mathbf{B})^\dagger)\mathbf{U}^T]^{-1}(\mathbf{U}\mathbf{B}^\dagger) \\ &= (\mathbf{U}\mathbf{B}^\dagger)^T(K_2)^{-1}(\mathbf{U}\mathbf{B}^\dagger),\end{aligned}$$

which completes the proof. □

From Theorem 2.6.11, it can be seen that the number of non-unit eigenvalues of the Schur complement matrix corresponds to the rank r, i.e. the number of the near-boundary interior nodes with perturbed stencil. In Table 2.7, we also provide comparison with numerical experiment (recall the relation (2.33)).

size, n	n^2	N_{ev}	$r = 4(\mathrm{n}-1)$
32	1024	124	124
64	4096	252	252
128	16386	508	508
256	65536	1020	1020

TABLE 2.7: n is size of the problem, n^2 is the total number of eigenvalues of S, N_{ev} is the number of non-unit eigenvalues computed numerically using Python code provided in Section 2.C, and r is the number of the near-boundary interior nodes defined in 2.52.

All spectral bounds for S_D and S_D^{-1} can be derived from Theorem 2.6.11.

Limiting case: surface-to-volume ratio $\to$ 1

Let us consider the limiting case, when the perturbation matrix $\mathbf{I^u_\sim}$ defined in (2.51) has full rank and coincides with the vector velocity identity matrix $\mathbf{I^u}$ defined in (2.47):

$$\mathbf{I^u_\sim} = \mathbf{I^u}.$$

For examples, this corresponds to the case $\mathrm{n} = 2$ for the square domain. Then, the inverse of the Dirichlet Schur complement matrix defined in (2.78b) is reduced as follows:

Theorem 2.6.12.

$$S_{lim}^{\dagger} = (\mathbf{B}(\mathbf{A_N} + \frac{2}{h^2}\mathbf{I^u})^{-1}\mathbf{B}^T)^{\dagger} = S_N^{\dagger} + \frac{2}{h^2}(\mathbf{B}\mathbf{B}^T)^{\dagger}, \tag{2.80}$$

where the matrix $(\mathbf{B}\mathbf{B}^T) : \mathcal{P}^h \to \mathcal{P}^h$ *is the pressure Laplacian with the Neumann boundary conditions and the constant nullspace* (2.67).

Proof. First, the velocity identity matrix can be decomposed under the DHHD using (2.70) as follows:

$$\mathbf{I^u} = \mathbf{B}^{\dagger}\mathbf{B} + \mathbf{C}^{\dagger}\mathbf{C}.$$

Next, for $\mathbf{U} = \mathbf{U}^T = \mathbf{I^u}$, we have:

$$K_2^{-1} = (\frac{h^2}{2}(\mathbf{B}^{\dagger}\mathbf{B} + \mathbf{C}^{\dagger}\mathbf{C}) + (\mathbf{C}^T\mathbf{C})^{\dagger})^{-1},$$

then, using orthogonality of $\mathbf{B}$ and $\mathbf{C}$, we have for S_{lim}:

$$S_{lim}^{\dagger} - S_N^{\dagger} = (\mathbf{B}^{\dagger})^T(\frac{h^2}{2}\mathbf{B}^{\dagger}\mathbf{B})^{\dagger}(\mathbf{B}^{\dagger}) = \frac{2}{h^2}(\mathbf{B}^{\dagger})^T\mathbf{B}^{\dagger} = \frac{2}{h^2}(\mathbf{B}\mathbf{B}^T)^{\dagger}.$$

This explains why the SIMPLE preconditioner and the preconditioner presented in [36] works well for geometries with high s-t-v ratio. Indeed, for $\mathbf{I}_{\sim}^{\mathbf{u}} \to \mathbf{I^u}$, we have $S_D^{\dagger} \to S_{lim}^{\dagger}$. The result is straightforward to generalize for voxel-based geometries. □

Appendix for Chapter 2

2.A Exterior boundary conditions for DRP flow experiment

In the main text in Section 2.2.2 the periodic boundary conditions were described for all three direction according to homogenization theory. In this section we formulate additional setting of the flow experiments which are used in our computational experiments and are implemented in the code. Similarly, we assume that z direction is the flow direction, and x, y directions are the tangential ones. We further subdivide the exterior boundary Γ_{ext} defined in (2.14) into three parts:

$$\Gamma_{\mathrm{ext}} = \Gamma_{\mathrm{in}} \cup \Gamma_{\mathrm{out}} \cup \Gamma_{\mathrm{t}}, \tag{2.81}$$

where Γ_{t} denotes the tangential exterior boundary:

$$\Gamma_{\mathrm{t}} = (\Gamma_{\mathrm{x}=0} \cup \Gamma_{\mathrm{x}=1} \cup \Gamma_{\mathrm{y}=0} \cup \Gamma_{\mathrm{y}=1}) \cap \overline{\Omega}_h^f, \tag{2.82}$$

and Γ_{in}, Γ_{out} denote the inflow, outflow boundaries, respectively:

$$\Gamma_{\mathrm{in}} = \Gamma_{\mathrm{z}=0} \cap \overline{\Omega}_h^f, \quad \Gamma_{\mathrm{out}} = \Gamma_{\mathrm{z}=1} \cap \overline{\Omega}_h^f. \tag{2.83}$$

Let us further denote $\hat{\mathbf{n}}$ the outward unit normal of the boundary Γ_{ext}. Then, one can uniqely write the following decomposition for the velocity vector field $\mathbf{u}$:

$$\mathbf{u} = \mathbf{u}_\perp + \mathbf{u}_\parallel, \tag{2.84}$$

where $\mathbf{u}_\perp = u_\perp \hat{\mathbf{u}}_\perp = (\hat{\mathbf{n}} \cdot \mathbf{u})\hat{\mathbf{n}}$ and $\mathbf{u}_\parallel = \mathbf{u} - \mathbf{u}_\perp = u_\parallel \hat{\mathbf{u}}_\parallel = (\hat{\mathbf{n}} \times \mathbf{u}) \times \hat{\mathbf{n}}$ are the normal and tangential components, respectively.

2.A.1 Boundary conditions in tangential directions

Except of the periodic boundary conditions, several options can be considered for the tangential boundary Γ_{t}:

1. The first option is to impose the *no-slip* plus *no-penetration* boundary condition, which is the zero Dirichlet b.c. imposed on the velocity:

   $$\mathbf{u} = \mathbf{0} \text{ on } \Gamma_{\mathrm{t}}. \tag{2.85}$$

2. The second option is to impose the *free-slip* boundary condition, which again includes the *no-penetration* boundary condition imposed on the normal component of the velocity:

   $$u_\perp = (\mathbf{u} \cdot \hat{\mathbf{n}}) = 0 \text{ on } \Gamma_{\mathrm{t}}. \tag{2.86}$$

   and for the tangential part of the normal traction force we prescribe zero value as follows:

   $$(\nabla \mathbf{u} \cdot \hat{\mathbf{n}} - p\hat{\mathbf{n}})_\parallel = \partial u_\parallel / \partial \hat{\mathbf{n}} = 0 \text{ on } \Gamma_{\mathrm{t}}. \tag{2.87}$$

2.A.2 Boundary conditions in flow direction

In order to define other then periodic boundary conditions in the direction of flow, we assume that there always exist fluid transient regions at the inlet $\Gamma_{\rm in}$ (inflow region) and at the outlet $\Gamma_{\rm out}$ (outflow region). Formally, we have $\Gamma_{\rm in} = \Gamma_{\rm z=0} \subset \overline{\Omega}_f^h$ and $\Gamma_{\rm out} = \Gamma_{\rm z=1} \subset \overline{\Omega}_f^h$. These transient regions are assumed to be large enough to ensure stationary behaviour of the flow at the exterior boundaries of the computational domain.
On the boundary $\Gamma_{\rm out}$, we consider the *outflow* boundary conditions, given as follows:

$$(\nabla \mathbf{u} \cdot \hat{\mathbf{n}} - p\hat{\mathbf{n}})_\perp = \partial u_\perp / \partial \hat{\mathbf{n}} - p = p_{\rm out}, \quad u_\parallel = 0 \text{ on } \Gamma_{\rm out}, \tag{2.88}$$

where $p_{\rm out}$ is a given outflow pressure, which is typically assumed zero.
On the boundary $\Gamma_{\rm in}$, we consider two boundary conditions. Firstly, the inflow Dirichlet boundary condition, which is given as follows:

$$u_\perp = u_{\rm in}, \quad u_\parallel = 0 \text{ on } \Gamma_{\rm in}, \tag{2.89}$$

where $u_{\rm in}$ is a given normal velocity. Secondly, similarly to the outflow boundary conditions (2.88), we define the *inflow* boundary conditions as follows:

$$(\nabla \mathbf{u} \cdot \hat{\mathbf{n}} - p\hat{\mathbf{n}})_\perp = \partial u_\perp / \partial \hat{\mathbf{n}} + p = p_{\rm in}, \quad u_\parallel = 0 \text{ on } \Gamma_{\rm in}, \tag{2.90}$$

where $p_{\rm in}$ is a given inflow pressure which drives the flow.
In the DRP flow experiment, except of the periodic boundary conditions (2.15), we consider several other combinations for the boundary conditions imposed in the flow direction:

1. The first option is to combine the Dirichlet boundary condition (2.89) on $\Gamma_{\rm in}$ and the outflow BC (2.88) on $\Gamma_{\rm out}$. This combination is also sometimes called as Velocity-Inlet-Pressure-Outlet (VIPO) in DRP.

2. The second option is to combine the inflow boundary condition (2.90) on $\Gamma_{\rm in}$ and the outflow boundary condition (2.88) on $\Gamma_{\rm out}$. Similarly to VIPO, this combination can be called as Pressure-Inlet-Pressure-Outlet (PIPO).

2.B On the coupled analogues of the CG-SIMPLE and CG-Uzawa algorithms

A wide class of preconditioners for the stationary Stokes equations (see, e.g., [23, 29, 73, 74, 33]), including the classical SIMPLE and Uzawa algorithms in their coupled forms, can be constructed by discarding one or more factors in the following approximation of the matrix $\mathbb{A}$ from (2.3):

$$\hat{\mathbb{A}} = \hat{\mathbb{L}}\hat{\mathbb{D}}\hat{\mathbb{U}} = \begin{bmatrix} \mathbf{I^u} & \\ \mathbf{B}\hat{\mathbf{A}}_1^{-1} & I^p \end{bmatrix} \begin{bmatrix} \hat{\mathbf{A}}_2 & \\ & -\hat{S} \end{bmatrix} \begin{bmatrix} \mathbf{I^u} & \hat{\mathbf{A}}_3^{-1}\mathbf{B}^T \\ & I^p \end{bmatrix}, \tag{2.91}$$

where $\hat{\mathbf{A}}_1$, $\hat{\mathbf{A}}_2$, $\hat{\mathbf{A}}_3$ are (possibly different) approximations of $\mathbf{A}$, and $\hat{S}$ is an approximation of the Schur complement matrix S from (2.5). Popular options are block lower triangular, block diagonal, and block upper triangular preconditioners:

$$\mathbb{L}\hat{\mathbb{D}} = \begin{bmatrix} \hat{\mathbf{A}} & \\ \mathbf{B} & -\hat{S} \end{bmatrix}, \quad \hat{\mathbb{D}} = \begin{bmatrix} \hat{\mathbf{A}} & \\ & -\hat{S} \end{bmatrix}, \quad \hat{\mathbb{D}}\mathbb{U} = \begin{bmatrix} \hat{\mathbf{A}} & \mathbf{B}^T \\ & -\hat{S} \end{bmatrix}, \tag{2.92}$$

where $\hat{\mathbf{A}}$ is an approximations of $\mathbf{A}$.
Let us consider the stationary iteration corresponding to an operator splitting $\mathbb{A} = \hat{\mathbb{A}} - (\hat{\mathbb{A}} - \mathbb{A})$, given as follows:

$$\begin{bmatrix} \mathbf{u}_h^{k+1} \\ p_h^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_h^{k} \\ p_h^{k} \end{bmatrix} + \hat{\mathbb{A}}^{-1}(\begin{bmatrix} \mathbf{f}_h \\ g_h \end{bmatrix} - \mathbb{A} \begin{bmatrix} \mathbf{u}_h^{k} \\ p_h^{k} \end{bmatrix}). \tag{2.93}$$

For certain choices of the preconditioner $\hat{\mathbb{A}}$, the coupled iteration (2.93) can be equivalently rewritten as the preconditioned Richardson iteration for the reduced system (2.4):

$$p_h^{k+1} = p_h^k + \alpha \hat{S}^{-1}(g_h^S - S p_h^k), \tag{2.94}$$

where $\alpha > 0$ is a parameter of the Richardson iteration.
In subsection 2.B.1 and 2.B.2, we show how the classical non-Krylov-accelerated versions of the Uzawa and SIMPLE algorithms can be equivalently written as stationary iterations in either coupled or reduced form.

2.B.1 Classical Uzawa algorithm

We consider the classical Uzawa algorithm in the following form (see., e.g., [23]):

$$\begin{cases} \mathbf{A}\mathbf{u}_h^{k+1} = \mathbf{f}_h - \mathbf{B}^T p_h^k, \\ \hat{S}_{\text{uzawa}} \delta p_h^k = \alpha_{\text{uzawa}} (\mathbf{B}\mathbf{u}_h^{k+1} - g_h), \\ p_h^{k+1} = p_h^k + \delta p_h^k, \end{cases} \tag{2.95}$$

where α_{uzawa} is a relaxation parameter of the algorithm, and $\hat{S}_{\text{uzawa}}$ is given in (2.7).

Proposition 2.B.1. *Firstly, as it is shown in [75], the Uzawa iteration* (2.95) *can be regarded as a stationary iteration of the form* (2.93) *with a preconditioner* $\hat{\mathbb{A}}$ *given as follows:*

$$\hat{\mathbb{A}} = \hat{\mathbb{A}}_{\text{uzawa}} = \begin{bmatrix} \mathbf{A} & \\ \mathbf{B} & -(1/\alpha_{\text{uzawa}}) \hat{S}_{\text{uzawa}} \end{bmatrix}, \tag{2.96}$$

which corresponds to a block lower triangular preconditioner $\hat{\mathbb{LD}}$ *defined in* (2.92) *if we take:*

$$\hat{S} = (1/\alpha_{\text{uzawa}}) \hat{S}_{\text{uzawa}}, \quad \hat{\mathbf{A}} = \mathbf{A}. \tag{2.97}$$

Secondly, the Uzawa iteration (2.95) *is equivalent to the Richardson iteration* (2.94) *(see, e.g. [23]), if we take:*

$$\hat{S} = \hat{S}_{\text{uzawa}}, \quad \alpha = \alpha_{\text{uzawa}}. \tag{2.98}$$

2.B.2 Classical SIMPLE algorithm

Let us now consider the classical SIMPLE algorithm in the following form [30, 33]:

$$\begin{cases} \mathbf{A}\mathbf{u}_h^{k+\frac{1}{2}} = \mathbf{f}_h - \mathbf{B}^T p_h^k, \\ \hat{S}_{\text{simple}} \delta p_h^k = \alpha_{\text{simple}} (\mathbf{B}\mathbf{u}_h^{k+\frac{1}{2}} - g_h), \\ \hat{\mathbf{A}}_{\text{simple}} \delta \mathbf{u}_h^k = -\mathbf{B}^T \delta p_h^k, \\ p_h^{k+1} = p_h^k + \delta p_h^k, \\ \mathbf{u}_h^{k+1} = \mathbf{u}_h^{k+\frac{1}{2}} + \delta \mathbf{u}_h^k, \end{cases} \tag{2.99}$$

where α_{simple} is a pressure damping parameter of the algorithm and $\hat{S}_{\text{simple}}$ is defined in (2.7).

Proposition 2.B.2. *Firstly, as it is shown in [33], the iterative process* (2.99) *can be regarded as a stationary iteration of the form* (2.93) *with a preconditioner* $\hat{\mathbb{A}}$ *given as follows:*

$$\hat{\mathbb{A}} = \hat{\mathbb{A}}_{\text{simple}} = \begin{bmatrix} \mathbf{A} & \\ \mathbf{B} & -(1/\alpha_{\text{simple}})\hat{S}_{\text{simple}} \end{bmatrix} \begin{bmatrix} \mathbf{I^u} & \hat{\mathbf{A}}^{-1}_{\text{simple}}\mathbf{B}^T \\ & I^p \end{bmatrix}, \tag{2.100}$$

which corresponds to a preconditioner $\hat{\mathbb{A}}$ *of the block structure* (2.91) *if we take:*

$$\hat{\mathbf{A}}_1 = \hat{\mathbf{A}}_2 = \hat{\mathbf{A}}, \quad \hat{\mathbf{A}}_3 = \hat{\mathbf{A}}_{\text{simple}}, \quad \hat{S} = (1/\alpha_{\text{simple}})\hat{S}_{\text{simple}}. \tag{2.101}$$

Secondly, the SIMPLE iteration (2.99) *is equivalent to the Richardson iteration* (2.94) *(see, e.g. [20]), if we take:*

$$\hat{S} = \hat{S}_{\text{simple}}, \quad \alpha = \alpha_{\text{simple}}. \tag{2.102}$$

It should be noted, that different splittings of $\mathbb{A}$ in the coupled iterations (2.93) may result in the same reduced iteration (2.94). For example, replacing $\hat{S}_{\text{uzawa}}$, α_{uzawa} by $\hat{S}_{\text{simple}}$, α_{simple} in (2.96) results in the following preconditioner $\hat{\mathbb{A}}$:

$$\hat{\mathbb{A}} = \hat{\mathbb{A}}^*_{\text{simple}} := \begin{bmatrix} \mathbf{A} & \\ \mathbf{B} & -(1/\alpha_{\text{simple}})\hat{S}_{\text{simple}} \end{bmatrix}, \tag{2.103}$$

which corresponds to a block lower triangular preconditioner $\hat{\mathbb{L}}\hat{\mathbb{D}}$ defined in (2.92) if we take:

$$\hat{S} = (1/\alpha_{\text{simple}})\hat{S}_{\text{simple}}, \quad \hat{\mathbf{A}} = \mathbf{A}. \tag{2.104}$$

Then, the coupled iteration (2.93) with the preconditioner $\hat{\mathbb{A}} = \hat{\mathbb{A}}^*_{\text{simple}}$ determines the following iterative process:

$$\begin{cases} \mathbf{A}u_h^{k+1} = \mathbf{f}_h - \mathbf{B}^T p_h^k, \\ \hat{S}_{\text{simple}}\delta p_h^k = \alpha_{\text{simple}}(\mathbf{B}\mathbf{u}_h^{k+1} - g_h), \\ p_h^{k+1} = p_h^k + \delta p_h^k. \end{cases} \tag{2.105}$$

In the same time, the iterative process (2.105) is also equivalent to the Richardson iteration (2.94) for the choice (2.102), hence it is also equivalent to the classical SIMPLE algorithm (2.99). So, according to the terminology from [25, 24], the CG-SIMPLE algorithm can be also identified as the preconditioned with SIMPLE CG-accelerated Uzawa algorithm.

2.B.3 Reduced analogue for the block diagonal preconditioner

Instead of the block lower triangular preconditioners (2.96) and (2.103), in practice the block diagonal preconditioner is often used:

$$\hat{\mathbb{A}}_{\text{diag}} = \begin{bmatrix} \mathbf{A} & \\ & -(1/\alpha)\hat{S} \end{bmatrix}.$$

Proposition 2.B.3. *The stationary iteration corresponding to the operator splitting* $\mathbb{A} = \hat{\mathbb{A}} - (\hat{\mathbb{A}}_{\text{diag}} - \mathbb{A})$*:*

$$\hat{\mathbb{A}}_{\text{diag}} \begin{bmatrix} \mathbf{u}_h^{k+1} \\ p_h^{k+1} \end{bmatrix} = \hat{\mathbb{A}}_{\text{diag}} \begin{bmatrix} \mathbf{u}_h^{k} \\ p_h^{k} \end{bmatrix} - \left(\mathbb{A} \begin{bmatrix} \mathbf{u}_h^{k} \\ p_h^{k} \end{bmatrix} - \begin{bmatrix} \mathbf{f}_h \\ g_h \end{bmatrix} \right)$$

is equivalent to the two-step Richardson iteration for the reduced system, given as follows:

$$p_h^{k+1} = p_h^k + \alpha \hat{S}^{-1}(g_h^S - S p_h^{k-1}).$$

2.B.4 Note on the Krylov subspace acceleration

For the coupled form of the Uzawa and SIMPLE algorithms, usually the GMRES (or BiCG) Krylov subspace method is used to accelerate stationary iterations (2.93) (see, e.g., [73, 29]), since the corresponding block lower triangular preconditioners are not symmetric. Alernatively, MINRES can be used with block diagonal preconditioner. We use CG to accelerate the reduced stationary iterations (2.94). Despite non-accelerated Uzawa and SIMPLE can be written equivalently in both reduced and coupled forms, in fact, different Krylov accelerators of the SIMPLE or Uzawa algorithms may result in non-equivalent iterative methods.

2.C Implementations details

Two implementations of the presented solvers are done. First, in the framework SCoPeS (stands for Skoltech Computing Permeability Solver). The SCoPeS solver is built on top of the PETSc open-source library [76, 77], which provides a linear algebra backend for the scalable (MPI parallel) solution of partial differential equations, including data structures for sparse matrix computations, a context for Krylov subspace methods (PETSc.KSP module), and a variety of preconditioners (PETSc.PC module). Particularly, the PCG method that used for inverting S, $\hat{S}$, and A is implemented within PETSc.KSP.KSPCG routine. We use BoomerAMG [51], which has demonstrated excellent performance in solving DRP problems. The second implementation is a part of the PoreChem package [78], where Intel MKL is used for sparse linear algebra, and SAMG [79] is used for algebraic multigrid.

```python
import numpy as np # dense linear algebra
import scipy.sparse as sp # sparse linear algebra

l = 6
n = 2**l # problem size
h = 2**(-l) # mesh size

I_shifted = sp.eye((n)) # 1d identity for shifted grid
I_aligned = sp.eye((n-1)) # 1d identity for aligned grid
B_1d = sp.diags([-1, 1], [-1, 0], shape=(n, n - 1))/h # 1d derivative

# co-directional derivatives
B_u_x = sp.kron(I_shifted, B_1d) # 2d derivative du/dx
B_v_y = sp.kron(B_1d, I_shifted) # 2d derivative dv/dy
B = sp.bmat([[-B_u_x, -B_v_y]]) # negative divergence

#  skew-directional derivatives
B_u_y = sp.kron(-B_1d.T, I_aligned) # 2d derivative du/dy
B_v_x = sp.kron(I_aligned, -B_1d.T) # 2d derivative dv/dx
C = sp.bmat([[-B_u_y, B_v_x]]) # velocity curl operator

A_N = B.T@B + C.T@C # Neumann velocity Laplacian

# perturbation matrix
diag = np.zeros(n); diag[0] = 1; diag[-1] = 1
I_shifted_tilde = sp.diags(diag, 0)
Iu_tilde = sp.kron(I_shifted_tilde, I_aligned)
Iv_tilde = sp.kron(I_aligned, I_shifted_tilde)
I_uv_tilde = sp.bmat([[Iu_tilde, None],[None, Iv_tilde]])

A_D = A_N + 2/(h**2)*I_uv_tilde # Dirichlet velocity Laplacian

stokes_aux = sp.bmat([[A_N, B.T],[B, None]]) # Stokes auxiliary
stokes_prim = sp.bmat([[A_D, B.T],[B, None]]) # Stokes primary

# construct RHS for the lid-driven cavity problem
source = np.zeros(n)
source[0] = 1
fu = (2/h**2)*np.kron(source, np.ones(n-1))
fv = np.zeros(n*(n-1))
fp = np.zeros(n*n)
f = np.concatenate((fu,fv,fp))

sol = sp.linalg.spsolve(stokes_prim, f) # solve
res = np.linalg.norm(stokes_prim@sol-f)/np.linalg.norm(f)
print("residual: " + str(res)) # check the residual is small

# extract solutions
sol_u = sol[:n*(n-1)]
sol_v = sol[n*(n-1):-n*n]
sol_p = sol[-n*n:]
sol_p -= np.sum(sol_p)/n**2 # substruct constant nullspace
```

2.D Convergence in preconditioned vs unprecinditioned norm

The Appendix provides additional information for numerical experiments from Sections 2.5.1 and 2.5.2. Tables 2.8, 2.9, 2.10 correspond to the convergence plot presented in Fig. 2.4. The tables include relative permeability error e^k and the corresponding number of iteration k for different residual thresholds $\varepsilon_S = 10^{-1}$, $\varepsilon_S = 10^{-2}$, $\varepsilon_S = 10^{-3}$, respectively. Note, for the CG-SIMPLE algorithm, the tables include thresholds for both preconditioned and unpreconditioned residual norms for the comparison purposes. Similar results corresponding to the convergence plot presented in Fig. 2.8 for thresholds $\varepsilon_S = 10^{-2}$, $\varepsilon_S = 5 \cdot 10^{-3}$, $\varepsilon_S = 10^{-3}$ are provided in Tables 2.11, 2.12, and 2.13, respectively.
Additionally, in Fig. 2.5, we plot the unpreconditioned residual norm for the CG-SIMPLE algorithm corresponding to the convergence history from Fig. 2.4, where the preconditioned residual is plotted. Note, the unpreconditioned residual always decreases monotonically.

TABLE 2.8: Permeability error e^k (iteration k) for residual threshold $\varepsilon_S = 0.1$ corresponding to the convergence history from Fig. 2.4.

Sample	Uzawa	SIMPLE (prec)	SIMPLE (unprec)
A	33.24 (9)	16.86 (3)	1.829 (8)
B	51.87 (8)	19.28 (3)	1.955 (8)
C	76.97 (9)	39.10 (4)	4.161(10)
D	11.20 (8)	3.320 (3)	0.571 (7)
E	53.79 (8)	51.42 (3)	3.565(10)
S	0.7281(6)	0.4033(2)	0.059 (6)

TABLE 2.9: Permeability error e^k (iteration k) for residual threshold $\varepsilon_S = 0.01$ corresponding to the convergence history from Fig. 2.4.

Sample	Uzawa	SIMPLE (prec)	SIMPLE (unprec)
A	3.412(114)	0.840 (10)	0.0128 (23)
B	5.013 (92)	0.678 (11)	0.0162 (24)
C	9.430 (88)	1.613 (13)	0.0357 (27)
D	1.119(112)	0.278 (9)	0.0044 (25)
E	6.097 (74)	1.427 (13)	0.0960(29)
S	0.022 (51)	0.016 (11)	0.0005 (31)

TABLE 2.10: Permeability error e^k (iteration k) for residual threshold $\varepsilon_S = 0.001$ corresponding to the convergence history from Fig. 2.4.

Sample	Uzawa	SIMPLE (prec)	SIMPLE (unprec)
A	0.2209(1237)	0.0169 (22)	0.00120 (39)
B	0.0146 (800)	0.0221 (23)	0.00260 (44)
C	0.5852 (944)	0.0557 (25)	0.00260 (48)
D	0.0286 (807)	0.0084 (22)	0.00006 (47)
E	0.4030 (837)	0.0936 (26)	0.01230 (55)
S	0.0002 (134)	0.0005 (31)	0.00000 (58)

TABLE 2.11: Permeability error e^k (iteration k) for residual threshold $\varepsilon_S = 0.01$ corresponding to the convergence history from Fig. 2.8.

	Uzawa	SIMPLE (prec)	SIMPLE (unprec)
s-t-v=46.9	$6.6e-3$ (55)	$1.1e-4$ (8)	$1.1e-4$ (8)
s-t-v=30.5	$4.7e-3$ (42)	$4.8e-4$ (9)	$1.1e-4$ (12)
s-t-v=22.3	$5.6e-3$ (33)	$7.7e-4$ (11)	$1.4e-4$ (15)
s-t-v=17.3	$3.8e-3$ (30)	$11.7e-4$ (11)	$1.5e-4$ (18)
s-t-v=14.0	$3.8e-3$ (25)	$9.5e-4$ (13)	$1.6e-4$ (20)

TABLE 2.12: Permeability error e^k (iteration k) for residual threshold $\varepsilon_S = 0.005$ corresponding to the convergence history from Fig. 2.8.

	Uzawa	SIMPLE (prec)	SIMPLE (unprec)
s-t-v=46.9	$2.1e-3$ (76)	$1.1e-4$ (8)	$2.24e-5$ (10)
s-t-v=30.5	$1.7e-3$ (53)	$1.6e-4$ (11)	$2.67e-5$ (15)
s-t-v=22.3	$1.4e-3$ (47)	$2.5e-4$ (14)	$3.17e-5$ (19)
s-t-v=17.3	$1.1e-3$ (38)	$1.9e-4$ (17)	$3.25e-5$ (23)
s-t-v=14.0	$0.7e-3$ (35)	$2.5e-4$ (18)	$3.64e-5$ (26)

TABLE 2.13: Permeability error e^k (iteration k) for residual threshold $\varepsilon_S = 0.001$ corresponding to the convergence history from Fig. 2.8.

	Uzawa	SIMPLE (prec)	SIMPLE (unprec)
s-t-v=46.9	$5.39e-5$ (137)	$9.6e-6$ (12)	$2.24e-6$ (14)
s-t-v=30.5	$4.43e-5$ (97)	$5.9e-6$ (17)	$1.47e-6$ (21)
s-t-v=22.3	$4.26e-5$ (80)	$1.4e-5$ (22)	$1.47e-6$ (28)
s-t-v=17.3	$3.62e-5$ (60)	$1.24e-5$ (27)	$1.49e-6$ (34)
s-t-v=14.0	$3.40e-5$ (51)	$1.43e-5$ (29)	$1.46e-6$ (39)

Chapter 3

Stokes-Brinkman equations in tight porous media

3.1 Problem statement

3.1.1 Representing pore-space geometries: grey-scale images

In the case of grey-scale images, the entire cube $\overline{\Omega}_h = [0, L]^3$ (defined in (2.9) for binary images), consist of three parts:

$$\overline{\Omega}_h = \overline{\Omega}_h^f \cup \overline{\Omega}_h^p \cup \overline{\Omega}_h^s, \tag{3.1}$$

where Ω_h^s and Ω_h^f are the solid part and the pure fluid part (resolved porosity) as in the binary case, but additionally Ω_h^p appears, which represents the region with unresolved porosity. Similarly to (2.11), the domain partition (3.1) corresponds to a disjoint decomposition of the index set $\mathbb{I}^{\mathrm{n}}$:

$$\mathbb{I}^{\mathrm{n}} = \mathbb{I}_f^{\mathrm{n}} \sqcup \mathbb{I}_p^{\mathrm{n}} \sqcup \mathbb{I}_s^{\mathrm{n}},$$

such that:

$$\overline{\Omega}_h^* = \bigcup_{(i,j,k)\in\mathbb{I}_*^{\mathrm{n}}} \omega_{(i,j,k)}, \quad * = f, p, s.$$

As in the binary case, the solid part Ω_h^s is considered impenetrable, hence the computational domain becomes:

$$\Omega_h^{fp} = \Omega_h^f \cup \Omega_h^p.$$

The binary images correspond to the case $\Omega_h^p = \emptyset$. However, as it was mentioned in the Introduction, it is not always possible to resolve a fine micro-structure of a porous space to the binary state, especially in the case of low porosity images. In general, for each voxel $\omega_{(i,j,k)}$ we prescribe the porosity $\phi_{(i,j,k)} \in [0, 100]$ (%) which defines the partition (3.1) as follows:

$$\omega_{(i,j,k)} \subset \begin{cases} \Omega_h^f & \text{if} \quad \phi_{(i,j,k)} = 0, \\ \Omega_h^p & \text{if} \quad \phi_{(i,j,k)} \in (0, 100), \\ \Omega_h^s & \text{if} \quad \phi_{(i,j,k)} = 100. \end{cases} \tag{3.2}$$

For this, the grey part of the image is split into equal intervals and the porosity is proportionally prescribed for each interval. Note, that in (3.2) different porosity, coming from the gray image, can be considered in each porous voxel. We call such images *multiclass images*. However, most simulations in our numerical experiments are performed for *ternary images* with some constant average porosity prescribed to all porous voxels from Ω_h^p. For examples of ternary images see Fig. 3.1.

3.1.2 Governing equations

There are two basic approaches for the numerical flow simulations when porous and free-flow regions co-exist at different scales. The first approach is to use the coupled Stokes-Darcy equations (see, e.g., [80, 81, 82]), which consider different models in different subdomains coupled via proper interface conditions. In DRP, the other approach, namely the Stokes-Brinkman model, sometimes also called the single-domain approach, is valid in both the fluid and porous subregions (see, e.g., [83]). Note, that often in the literature such models are called just Brinkman models or Darcy-Brinkman models. However, we use the Stokes-Brinkman term to emphasize that pure fluid voxels exist in the domain.

Stokes-Brinkman equations. The Stokes-Brinkman equations can be considered as a combination of Stokes and Darcy equations:

$$\begin{aligned} -\Delta \mathbf{u} + \mathbf{K}^{-1}\mathbf{u} + \nabla p = \mathbf{0} \text{ in } \Omega_h^{fp}, \quad \mathbf{K}^{-1} \geq 0, \\ -\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega_h^{fp}. \end{aligned} \tag{3.3}$$

Here $\mathbf{K}$ is a spatially varying permeability tensor, and $\mathbf{K}^{-1}$ is the flow resistivity. To avoid ambiguity with the effective (macro-scale) permeability k^{eff}, we refer this permeability tensor $\mathbf{K}$ as microscopic permeability. In our study, we consider isotropic permeability tensors, i.e. $\mathbf{K} = K\mathbf{I}$. The flow regime can be administrated via permeability tensor $\mathbf{K}$. Namely, in Ω_h^f we assume $K = +\infty$, then the reaction term $\mathbf{K}^{-1}$ disappears and the Stokes-Brinkman system reduces to the Stokes (2.1) system, while for $K \ll 1$, the viscous term can be neglected leading to the Darcy equation. It is crucial to highlight, that a single equation covers two fundamentally different flow regimes in this case. Consequently, delivering a unified numerical approach for solving the Stokes-Brinkman equations presents significant challenges, often necessitating the use of different preconditioners for different flow regimes. Namely, a robust Stokes-Brinkman solver should be able at efficiently handling both limiting cases: the Darcy problem (3.5) and the Stokes problem (2.1).

3.1.3 Approximation using Brinkman and Darcy equations

Brinkman perturbation of the Stokes-Brinkman problem. In tight reservoirs, one may face a necessity to compute the permeability for images where no pure fluid percolation path exists. In such cases, the flow consequently passes fluid and porous subregions; hence the effective permeability of the image is governed by the resistance of the porous subregions, while the resistance of the fluid subdomain could be neglected. For such samples, the perturbation $\tilde{\mathbf{K}}$ of the permeability tensor $\mathbf{K}$ can be considered by introducing a fictitious permeability value K_{Stokes} in the pure fluid voxels from Ω_h^f. Then, $\tilde{\mathbf{K}}^{-1}$ does not vanish in any of the voxels, and the Stokes-Brinkman equations (3.3) are thus perturbed to the Brinkman equations:

$$\begin{aligned} -\Delta \mathbf{u} + \tilde{\mathbf{K}}^{-1}\mathbf{u} + \nabla p = \mathbf{0} \text{ in } \Omega_h^{p}, \quad \tilde{\mathbf{K}}^{-1} > 0, \\ -\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega_h^{p}. \end{aligned} \tag{3.4}$$

The Brinkman equations (see, e.g., [84]) play an important role for classes of porous media flows, e.g., describing more adequately flows in large porosity domains and allowing to account for no-slip boundary conditions. Here, however, they are not used for a more accurate description of the flow, but they are considered as a perturbation to the Stokes-Brinkman equations (3.3).

Darcy approximation of the Brinkman problem. The Brinkman problem (3.4) can be further approximated by neglecting the viscous term, which results in the Darcy equation:

$$\begin{aligned} \tilde{\mathbf{K}}^{-1}\mathbf{u} + \nabla p = \mathbf{0} \text{ in } \Omega_h^p, \quad \tilde{\mathbf{K}}^{-1} > 0, \\ -\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega_h^p. \end{aligned} \tag{3.5}$$

It will be shown in this paper that for a certain class of images, the Darcy equation (3.5) can be solved instead of the Stokes-Brinkman equations (3.3) in order to dramatically reduce the computational time while preserving the accuracy of the effective permeability computations.
Note that the idea of solving the Darcy problem instead of the Stokes-Brinkman one is not a new one; it was earlier explored, e.g., in [15]. Here this approach is a part of the workflow, which is equipped with a preprocessor module (classifier) to automatically decide if it is reasonable to solve the Darcy approximation.

3.1.4 Boundary conditions

For multiclass images, we consider the same set of boundary conditions as for the Stokes problem. If one wants to use the periodic boundary conditions formulated in Section 2.2.2 for binary images, a similar to (2.16) periodicity constraint should be imposed on the porous part Ω_p^h (2.11), as well as on the microscopic permeability tensor $\mathbf{K}$. However, most of the experiments from Section 3.4 are performed with the Pressure-Inflow-Pressure-Outflow boundary conditions imposed in the flow direction (recall Section 2.A for details). On the interior boundaries, we impose the no-slip boundary conditions for the Stokes-Brinkman equations. Similarly to (2.12), the interior boundary for multiclass images is given as follows:

$$\Gamma_0 = (\overline{\Omega}_h^f \cup \overline{\Omega}_h^p) \cap \overline{\Omega}_h^s. \tag{3.6}$$

It is also worth noting, that no special conditions are imposed on the interface $\overline{\Omega}_h^f \cap \overline{\Omega}_h^p$, except for the continuity conditions inherited from the discretization, as described in Section 3.2.1.

Computing permeability

When computing permeability, the velocity is now integrated over Ω_h^{fp} (compare with 2.19):

$$\langle u_z \rangle = \frac{1}{|\Omega_h|} \int_{\Omega_h^{fp}} u_z.$$

3.2 Adaptation of the CG-SIMPLE algorithm

For solving the Stokes-Brinkman equations, we use the CG-SIMPLE Algorithm 1 described for the Stokes problem in Chapter 2, adapted to account for additional reaction (Brinkman) term. After finite-difference discretization of the Stokes-Brinkman equations using fully-staggered grids, we obtain the following system:

$$\mathbb{A} \begin{bmatrix} u_h \\ p_h \end{bmatrix} = \begin{bmatrix} f_h \\ g_h \end{bmatrix}, \quad \mathbb{A} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & 0 \end{bmatrix}, \tag{3.7}$$

where discretization of the momentum equation is:

$$\mathbf{F} = \mathbf{A} + \overline{\mathbf{K}}^{-1}, \tag{3.8}$$

where $\overline{\mathbf{K}}$ is the discretization of the permeability tensor $\mathbf{K}$, assembled as discussed in subsection 3.2.1. The Schur complement matrix (2.5) becomes:

$$S = \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T = \mathbf{B}(\mathbf{A} + \overline{\mathbf{K}}^{-1})^{-1}\mathbf{B}^T, \tag{3.9}$$

and the SIMPLE preconditioner becomes:

$$\hat{S}_{\text{simple}} = \mathbf{B}\hat{\mathbf{F}}^{-1}_{\text{simple}}\mathbf{B}^T, \quad \hat{\mathbf{F}}_{\text{simple}} = \text{diag}(\mathbf{A} + \overline{\mathbf{K}}^{-1}). \tag{3.10}$$

When Algorithm 1 is applied for the Stokes-Brinkman problem instead of the Stokes problem, the two major differences are:

1. When applying S on each outer iterations, we need to solve the momentum equation with the reaction-diffusion matrix $(\mathbf{A} + \overline{\mathbf{K}}^{-1})$, instead of the diffusion matrix $\mathbf{A}$.

2. When applying the SIMPLE preconditioner $\hat{S}_{\text{simple}}$, we need to solve the diffusion equation with strongly varying diffusion coefficient.

For both problems, the Algebraic Multigrid method works well.
In the limiting Darcy case (i.e. $\mathbf{A} = 0$), the matrix $\mathbf{F}$ is diagonal, such that $S = \hat{S}_{\text{simple}}$ and the Algorithm 1 degenerates to one outer iteration. When inverting $S = \hat{S}_{\text{simple}}$, the elliptic nature allows for efficient solutions using established multigrid techniques, even despite possibly intricate geometries and large variations in the microscopic permeability coefficient. In the Brinkman case (i.e. $\overline{\mathbf{K}}^{-1} > 0$), the viscous term is negligible, hence the SIMPLE preconditioner is efficient and typically converges in a few iterations in practice. For the limiting Stokes case (i.e. $\overline{\mathbf{K}}^{-1} = 0$), in Chapter 2 the CG-SIMPLE was systematically studied and justified for tight geometries.

3.2.1 Discretization of the reaction (Brinkman) term

For the discretization of the Stokes-Brinkman equations (3.3), we need to discretize the flow resistivity tensor $\mathbf{K}^{-1}$, which appears as a reaction term in these equations. We consider the scalar permeability coefficient K to be a piece-wise constant function of the porosity $\phi_{(i,j,k)}$ from (3.2), i.e. for each unresolved voxel $\omega_{(i,j,k)}, (i,j,k) \in \mathbb{I}^{\text{n}}_p$, we have $K_{(i,j,k)} = K(\phi_{(i,j,k)})$. The specific correlation formula between the porosity $\phi_{(i,j,k)} \in (0, 100)$ and microscopic permeability value $K_{(i,j,k)}$ used in our computational experiments is given in Section 3.4 (see eq. (3.11)). Thus, K can be identified with a function K^p_h from the discrete pressure space $\mathcal{P}_h$, for which the degrees of freedom are located at the centers of the voxels. However, the degrees of freedom for the velocity components on staggered grids are located on the voxel faces, necessitating some interpolation. Therefore, we should average either the porosity or the microscopic permeability over the neighboring voxels using the arithmetic mean [85], and the discretized permeability tensor, denoted as $\overline{\mathbf{K}} : (\mathcal{U}_h, \mathcal{V}_h)^T \to (\mathcal{U}_h, \mathcal{V}_h)^T$, should be constructed respectively. For instance, in 2D square domain, it is given as follows:

$$\overline{\mathbf{K}} = \begin{bmatrix} \text{diag}(A^p_x K^p_h) & \\ & \text{diag}(A^p_y K^p_h) \end{bmatrix},$$

where A_x^p and A_y^p are the averaging operators from the grid $\mathcal{P}_h$ to the grids $\mathcal{U}_h$ and $\mathcal{V}_h$, respectively. A formal definition of these averaging operators is given in Section 4.5, where discretization of the convection term for the Navier-Stokes equations is discussed.

3.3 Computational workflow

The ultimate objective of this Chapter is to propose a workflow for solving single-phase pore-scale flow problems in tight sandstones characterized by low porosity and connectivity in the case of the appearance of unresolved porosity. The workflow has essentially three components.
(i) The first stage, as usually, is the image processing. The multiclass models of real rock samples from tight reservoirs considered here are built according to the approach from [86], with the help of the double μCT scanning technique (see Section 3.3.1 for details).
(ii) At the second stage, the 3D images (multiclass models) are classified with respect to the connectivity of the resolved pore space Ω_h^f.
(iii) At the third stage, either the Stokes-Brinkman or Darcy solver is used to calculate the effective permeability for different classes of images.

3.3.1 Multiclass model preparation.

For tight reservoir rocks, the usage of DRP is not as straightforward as for higher porosity sandstones. This is due to the inherent trade-off between the spatial resolution of data and the representativeness of the size of the model. For this regime, a new approach has been developed to consider in a single 3D digital core model porosity from different scales (micro and sub-micro). To build a multiclass model of a core sample, double X-ray μCT scanning is performed. Before scanning, the sample is seated in the coreholder. The first scan is done for the sample fully saturated with air. Then, after air evacuation, the sample in the coreholder is saturated with Xenon (Xe) gas. Xe is a non-reactive and highly mobile gas that also has a lower radiolucency than grains. Comparison of two registered and calibrated 3D μCT data allows to map the distribution of Xe molecules and their amount in the volume of the digital core model. Assuming linear dependence between Xe intensity on the μCT data difference (μCT in air - μCT in Xe) and porosity of the 3D model voxels, it is possible to create a multiclass model. A specific porosity value characterizes each model class with a discreteness level equal to 1%. Multiclass model creation includes a few steps:

- Data acquisition for the same sample position, the same resolution, and the same X-ray parameters (electrical voltage and current, distance between X-ray source and sample centre, detector exposition and step of sample rotation).
- Image preprocessing (denoising, artefact removing).
- Registration of 3D images (matching images in space) .
- Intensity equilibration (to be sure that minerals without Xe have the same intensity on both 3D images).
- 3D images subtraction (to map only Xe distribution in model volume).
- Parametrization of dependence between Xe intensity and submicron porosity assuming that on subtracted images, minimum (zero) intensity corresponds

to 0% porosity (solid) and maximum intensity corresponds to 100% porosity (void).

- Applying determined "intensity-porosity" correlation to all 3D datasets of subtracted images.

It is also possible to construct a multiclass model by combining information about pore space structure from various sources of data: μCT, scanning microscopy (SEM), and focused ion beam with scanning microscopy (FIB-SEM). One can train a deep neural network to obtain a porosity map directly from SEM or FIB-SEM images. Another opportunity is to create a multiclass model by coarsening a high-resolution binary model to reduce the size of a computational domain but remain the same physical scale of the model.

3.3.2 Preprocessing stage. Sample classification.

A preprocessor is implemented as a separate module in C++ language; it relies on the Disjoint Set Union data structure [87]. First of all, as it is usual in DRP, it identifies isolated fluid (pure fluid or/and porous) subdomains and removes them. Next, the image is classified. The existence of percolation patch(es) in the computational domain $\Omega_h^p \cup \Omega_h^f$ is checked. If $\Omega_h^p \cup \Omega_h^f$ is connected, the Stokes-Brinkman problem (3.3) is well-posed, and the image is classified as an image of Category A. Furthermore, the connectivity of the pure fluid region Ω_h^f is checked. If pure fluid percolation exists, the image is moved from Category A to Category B - images with Stokes connectivity. Note that after such a preprocessing, Category A contains images that do not have Stokes connectivity but have Stokes-Brinkman connectivity via sequences of pure fluid and porous voxels.

3.4 Validation and performance study of the developed solvers

The computer simulations have been carried out to evaluate the performance of the developed Stokes-Brinkman solver for pore-scale simulation on 3D CT images of tight sandstones in the case of unresolved porosity.
In this Section, the main simulation results are presented. The performance of the developed Stokes-Brinkman solver, denoted SCoPeS-SB, is systematically investigated for samples coming from tight reservoirs. The performance of SCoPeS-SB and GeoDict Stokes-Brinkman solvers is compared for the considered class of problems. Further on, as discussed above, just Darcy's problem is solved for images with no Stokes connectivity using Darcy solver, denoted SCoPeS-D.

3.4.1 Samples database: ternary and multi-class images.

We start from three different multiclass samples of size 300^3, named *S1,S2,S3*, which are sub-samples of a large multiclass sample. Samples *S1,S2,S3* have 3.79, 3.25, 3.78 mln. non-solid voxels, correspondingly. Recall that *ternary images* are the images composed of solid voxels, fluid voxels, and identical porous voxels, the latter being responsible for all unresolved porosity. Stokes-Brinkman equations have to be solved to compute the permeability for ternary images. For each of these three multiclass samples, the following procedure for increasing porosity and segmenting into ternary images is applied. First, given a threshold $T \in (0, 100]$, we replace all porous voxels $\omega_{i,j,k} \in \Omega_p^h$ having porosity $\phi_{i,j,k} \geq T$ by pure fluid voxels. Next, the remaining

porosity is arithmetically averaged with resulting value $\tilde{\phi} \in (0, 100)$. In this way, nine ternary samples are created, which are encoded by N_T_$\tilde{\phi}$.raw, where N=S1,S2,S3, T = 100,90,80, and $\tilde{\phi}$ is resulting averaged porosity. Note, $T = 100$ corresponds to the case when the porosity is just averaged. Finally, constant permeability value is calculated for the averaged porosity $\tilde{\phi}$ according to (3.11) and assigned to all remaining porous voxels. Table 3.1 summarizes the connectivity for each of the nine samples. Six of them are from Category A, and three of them are from Category B (marked in **bold**).
The following correlation formula between porosity $\phi_{(i,j,k)} \in (0, 100)$ and permeability $K_{(i,j,k)}$ [mkDa] was used:

$$K_{(i,j,k)} = 7.251 \cdot 10^{-2} \exp(0.147 \cdot \phi_{(i,j,k)}). \tag{3.11}$$

This correlation formula (3.11) was derived using empirical petrophysical correlation from FIB-SEM data analysis [88, 86].

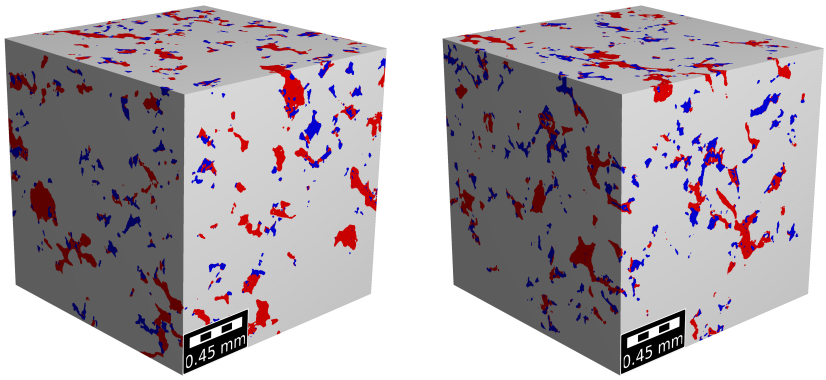
TABLE 3.1: Porosity (Resolved/Unresolved) and Connectivity of the S samples.

Samples	ϕ (Res, Unres)	Stokes Connectivity	Stokes-Brinkman Connectivity	File name
S1_100_61	0.14 (0.044, 0.096)	No	Yes	S1_100_61.raw
S1_90_56	0.14 (0.057, 0.083)	No	Yes	S1_90_56.raw
S1_80_50	0.14 (0.070, 0.070)	No	Yes	S1_80_50.raw
S2_100_60	0.14 (0.050, 0.090)	No	Yes	S2_100_60.raw
S2_90_55	0.14 (0.062, 0.078)	**Yes**	Yes	S2_90_55.raw
S2_80_49	0.14 (0.075, 0.065)	**Yes**	Yes	S2_80_49.raw
S3_100_58	0.12 (0.050, 0.070)	No	Yes	S3_100_58.raw
S3_90_53	0.12 (0.057, 0.063)	No	Yes	S3_90_53.raw
S3_80_48	0.12 (0.066, 0.054)	**Yes**	Yes	S3_80_48.raw

The same segmentation procedure was also applied for two multiclass images of size 600^3, named *T1,T2*, and having 30.5, 30.7 mln. non-solid voxels, respectively. Additionally, one large multiclass sample, *U1*, of size 1350^3 with 350.8 mln. non-solid voxels was studied. Porosities and connectivities of these samples are summarized in Table 3.2.

TABLE 3.2: Porosity (Resolved/Unresolved) and Connectivity of the T, U samples.

Samples	ϕ (Res, Unres)	Stokes Connectivity	Stokes-Brinkman Connectivity	File name
T1_100_59	0.141 (0.051, 0.089)	No	Yes	T1_100_59.raw
T1_90_54	0.141 (0.062, 0.079)	No	Yes	T1_90_54.raw
T1_80_49	0.141 (0.073, 0.067)	**Yes**	Yes	T1_80_49.raw
T2_100_58	0.142 (0.051, 0.091)	No	Yes	T2_100_58.raw
T2_90_54	0.142 (0.061, 0.081)	No	Yes	T2_90_54.raw
T2_80_49	0.142 (0.072, 0.070)	**Yes**	Yes	T2_80_49.raw
U1_100_59	0.143 (0.052, 0.091)	No	Yes	U1_100_59.raw



(A) sample T1_100_59 (B) sample T2_100_58

FIGURE 3.1: Ternary samples T1_100_59 and T2_100_58. Colors: grey - solid region Ω_h^s, red - unresolved porosity region Ω_h^p, blue - free pores Ω_h^f. GeoDict visualization.

This collection of images gives us a possibility to perform detailed testing of the performance of the Stokes-Brinkman solver for images with different porosity and different fraction of unresolved regions. The effective permeabilities computed with SCoPeS-SB solver is compared to the results obtained with GeoDict. Figure 3.1 presents explanatory pore-space visualization for two ternary samples: $T1\_100\_59$ and $T2\_100\_58$. Also, the result of Stokes-Brinkman simulation on ternary sample S2_80_49 is shown on Figure 3.2.
Not, the Raw files describing the used CT images are provided by the following link to enable other researchers to use them .

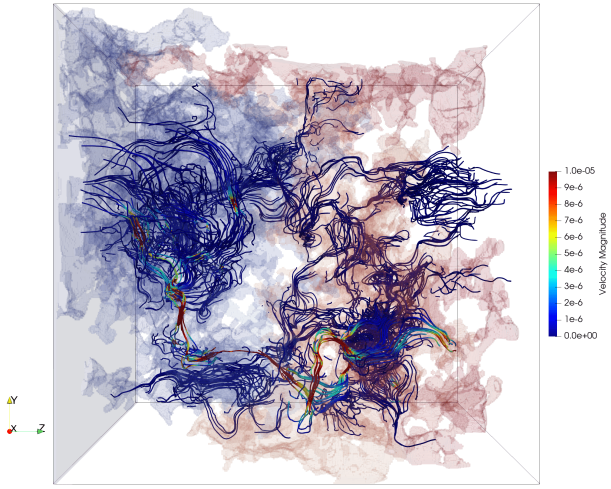


FIGURE 3.2: Velocity magnitude streamlines. Paraview visualization for ternary sample S2_80_49, Stokes-Brinkman problem.

3.4.2 Simulations on ternary images of size 300^3.

Solving Stokes-Brinkman equations for ternary images of Category A (no Stokes connectivity). Firstly, we consider three Category A images obtained from samples *S2* and *S3*. The simulation results of comparing the SCoPeS-SB solver and the SimpleFFT solver from GeoDict are presented in Table 3.3. Also, the simulations results with the

LIR solver from GeoDict are shown. However, LIR demonstrated poor performance for low-porosity images, therefore very few results were presented. It can be seen that SimpleFFT and LIR solvers converge very slowly but have the correct tendency, and the computed values are not far from what was expected.

TABLE 3.3: Ternary samples S2 and S3 of Category A. Permeability k_{zz}^{eff} in $mkDa$ computed with GeoDict (solvers SimpleFFT and LIR with periodic bc), and with SCoPeS-SB with pressure drop bc, $L = 0.0009$ m, nproc=8.

Sample S2_100_60, Perm of porous voxels 493.0 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver	k_{zz}^{eff}, $mkDa$	CPU, s	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$1.20 \cdot 10^2$	20511	10^{-6}	$6.80 \cdot 10^1$ (1564)
$1.2 \cdot 10^{-2}$	SimpleFFT:	$7.95 \cdot 10^1$	172009	10^{-7}	$7.31 \cdot 10^1$ (2769)
$1.9 \cdot 10^{-1}$	LIR:	$1.04 \cdot 10^2$	214560	10^{-8}	$7.27 \cdot 10^1$ (3499)
				10^{-9}	$7.27 \cdot 10^1$ (4090)
Sample S3_100_58, Perm of porous voxels 367.4 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	CPU, s	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$4.07 \cdot 10^1$	5862	10^{-7}	$2.23 \cdot 10^1$ (1829)
$1.1 \cdot 10^{-2}$	SimpleFFT:	$2.86 \cdot 10^1$	81435	10^{-8}	$2.54 \cdot 10^1$ (2199)
				10^{-9}	$2.57 \cdot 10^1$ (2693)
				10^{-10}	$2.57 \cdot 10^1$ (3265)
Sample S3_90_53, Perm of porous voxels 176.0 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	CPU, s	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$6.52 \cdot 10^1$	10758	10^{-6}	$2.50 \cdot 10^1$ (1487)
$8 \cdot 10^{-2}$	SimpleFFT:	$5.18 \cdot 10^1$	104421	10^{-7}	$5.12 \cdot 10^1$ (1875)
				10^{-8}	$4.92 \cdot 10^1$ (2220)
				10^{-9}	$4.92 \cdot 10^1$ (2733)

Note, all three images produced from *S1* sample also belong to Category A, and the corresponding results are available in Table 3.9 in the Appendix. In all cases, simulations with SCoPeS-SB show very robust convergence with respect to the selected tolerance ε_S. Unlike the GeoDict solvers, the computational time increases moderately when decreasing the tolerance value.

Solving Darcy approximation for ternary images of Category A (no Stokes connectivity). As it was mentioned above, in the case of no Stokes connectivity, it makes sense to first explore approximating Stokes-Brinkman equations with the Darcy equation, and after that solve it to compute the flow, and thus the permeability. The approximation is done by adding artificial permeability in the pure fluid voxels, and after that dropping the viscous terms. Substituting the velocity (in this case with a diagonal matrix) into the continuity equation, we obtain a scalar second order elliptic equation for the pressure.

Simulation results with Darcy model for samples *S2* and *S3* are presented in Table 3.5. The results are computed using Darcy solver SCoPeS-D. For comparison, the effective permeabilities computed with SCoPeS-SB solver are presented in the last line of the Table. One can see that the computations with the Darcy approximation are about

150 times faster compared to SCoPeS-SB, and at the same time the same accuracy in the computation of the permeability of the sample can be achieved. Similar results for samples *S1* are available in Table 3.10 in the Appendix.
Additionally, simulation results from sensitivity study on how the artificial permeability in the Darcy approximation influences the accuracy of the computations are presented in Table 3.11 in the Appendix. The sample S1_100_61 is considered there.

TABLE 3.4: Hardware/Software Specification.

Samples $300^3, 600^3$	
Hardware model	Operating System
2 x Intel Xeon E5-2687W v4@3.00 GHz, RAM 528 Gb DDR4 2400 Hz	Linux 4.15.0-154-generic Ubuntu 18.04.6
Samples 1350^3	
Dell PowerEdge R640, 2 x Intel Xeon Gold 6150@2.7 GHz, RAM 1536 Gb DDR4	CentOS Linux release 7.9.2009

TABLE 3.5: SCoPeS-D results, Darcy approximation for samples S2_100_60, S3_100_58 and S3_90_53 of Category A. Permeability k_{zz}^{eff} in $mkDa$ (CPU time in s), $\varepsilon_S = 10^{-9}$, $L = 0.0009$ m, nproc=8. BC: pressure drop 1 Pa. The last line for comparison recalls permeability and CPU time when solving Stokes-Brinkman equations.

$K_{Stokes}, mkDa$	S2_100_60	S3_100_58	S3_90_53
10^5	$4.11 \cdot 10^1$ (42.8)	$2.34 \cdot 10^1$ (43.4)	$3.53 \cdot 10^1$ (42.6)
10^7	$7.43 \cdot 10^1$ (44.1)	$2.57 \cdot 10^1$ (43.1)	$4.95 \cdot 10^1$ (44.2)
10^9	$7.54 \cdot 10^1$ (44.8)	$2.58 \cdot 10^1$ (43.4)	$4.99 \cdot 10^1$ (43.5)
10^{10}	$7.54 \cdot 10^1$ (44.5)	$2.58 \cdot 10^1$ (44.0)	$4.99 \cdot 10^1$ (43.7)
SCoPeS-SB:	$7.27 \cdot 10^1$ (4090)	$2.57 \cdot 10^1$ (3265)	$4.92 \cdot 10^1$ (2733)

Solving Stokes-Brinkman equations for ternary images of Category B (Stokes connectivity). Let us now discuss the simulation results for samples from Category B (Stokes connectivity). Consider remaining images obtained from samples *S2* and *S3*. Again, the Stokes-Brinkman equations are solved for them, and the computed effective permeability and the CPU time are reported in Table 3.6. One can observe that for these low porosity samples, SCoPeS-SB demonstrates very good performance. The computed effective permeability values are close to those computed with GeoDict. The convergence with respect to decreasing the tolerance is pronounced and stable. The increasing of the computational time when decreasing the tolerance is very moderate.
Solving Darcy approximation for ternary images of Category B (Stokes connectivity). For comparison, the Darcy approximation is also used for computing the effective permeability of samples from Category B. The results are summarized in Table 3.7. As expected, the Darcy approximation is not applicable for computing the effective permeability of the considered samples, namely samples for which Stokes connectivity exists. This illustrates the importance of the image classification stage in the presented workflow.

TABLE 3.6: Ternary samples S2 and S3 of Category B. Permeability k_{zz}^{eff} in $mkDa$ computed with GeoDict (solvers SimpleFFT and LIR with periodic bc), and with SCoPeS-SB with pressure drop bc, $L = 0.0009$ m, nproc=8.

Sample S2_90_55, Perm of porous voxels 236.3 mkDa					
GeoDict				SCoPeS-SB	
Tol	Solver	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$6.40 \cdot 10^2$	13996	10^{-5}	$5.26 \cdot 10^2$ (1853)
$1.1 \cdot 10^{-2}$	SimpleFFT:	$5.95 \cdot 10^2$	227036	10^{-6}	$5.93 \cdot 10^2$ (2343)
				10^{-8}	$5.91 \cdot 10^2$ (3520)
				10^{-9}	$5.91 \cdot 10^2$ (4243)
Sample S2_80_49, Perm of porous voxels 97.8 mkDa					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$6.13 \cdot 10^3$	2103	10^{-5}	$5.95 \cdot 10^3$ (1456)
10^{-2}	SimpleFFT:	$5.88 \cdot 10^3$	7347	10^{-6}	$5.81 \cdot 10^3$ (1816)
10^{-3}	SimpleFFT:	$5.84 \cdot 10^3$	35744	10^{-7}	$5.83 \cdot 10^3$ (2319)
10^{-1}	LIR:	$6.07 \cdot 10^3$	10509	10^{-8}	$5.83 \cdot 10^3$ (2892)
10^{-2}	LIR:	$5.86 \cdot 10^3$	33216		
10^{-3}	LIR:	$5.81 \cdot 10^3$	241828		
Sample S3_80_48, Perm of porous voxels 84.4 mkDa					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$1.89 \cdot 10^4$	568	10^{-3}	$1.63 \cdot 10^4$ (703)
10^{-2}	SimpleFFT:	$1.56 \cdot 10^4$	5912	10^{-4}	$1.60 \cdot 10^4$ (955)
10^{-3}	SimpleFFT:	$1.55 \cdot 10^4$	14099	10^{-5}	$1.55 \cdot 10^4$ (1190)
				10^{-7}	$1.55 \cdot 10^4$ (1945)

TABLE 3.7: SCoPeS-D results, Darcy approximation for samples S2_90_55, S2_80_49 and S3_80_48 of Category B. Permeability k_{zz}^{eff} in $mkDa$ (CPU time in s), $\varepsilon_S = 10^{-9}$, $L = 0.0009$ m, nproc=8. BC: pressure drop 1 Pa. The last line for comparison recalls permeability and CPU time when solving Stokes-Brinkman equations.

K_{Stokes}, $mkDa$	S2_90_55	S2_80_49	S3_80_48
10^5	$6.13 \cdot 10^1$ (43.3)	$8.19 \cdot 10^1$ (45.0)	$1.39 \cdot 10^2$ (42.5)
10^7	$1.56 \cdot 10^3$ (43.4)	$5.61 \cdot 10^3$ (43.1)	$1.05 \cdot 10^4$ (42.7)
10^9	$1.45 \cdot 10^5$ (43.6)	$5.57 \cdot 10^5$ (42.9)	$1.05 \cdot 10^6$ (43.4)
10^{11}	$1.45 \cdot 10^7$ (44.5)	$5.57 \cdot 10^7$ (43.6)	$1.05 \cdot 10^8$ (43.4)
SCoPeS-SB:	$5.91 \cdot 10^2$ (4243)	$5.83 \cdot 10^3$ (3600)	$1.55 \cdot 10^4$ (1945)

3.4.3 Simulations on multiclass image of size 300^3.

The developed solver is able to work directly on multiclass images, such that they have individual permeability values in each unresolved voxel. On the one hand, this imposes higher memory requirements to the solver, on the other hand, this feature of the solver might be essential for certain classes of rocks. To illustrate this option,

simulations are performed with image *S2*, as a multiclass image having 100 classes of porosity to account for the sub-micron scale effects. The results are summarized in Table 3.8. Robust convergence of the simulations with respect to the tolerance in the case of multiclass image can be observed. The effective permeabilities computed on the three ternary images produced from the multiclass image *S2* are also provided in the table. It can be seen that in this particular case the effective permeabilities computed on the multiclass and on the three different ternary images differ significantly. This means that results obtained from simulations on ternary images might not be a good approximation of the results obtained on the true, multiclass grey image, at least if a simple averaging is used when producing ternary image from a multiclass image.

TABLE 3.8: SCoPeS-SB results, permeability k_{zz}^{eff} in $mkDa$ for multiclass sample S2, $L = 0.0009$ m, nproc = 8. BC: pressure drop 1 Pa. Permeabilities of the corresponding ternary samples are also reminded.

ε_S	k_{zz}^{eff}, $mkDa$	CPU time, s
10^{-5}	$8.63 \cdot 10^2$	1421
10^{-6}	$5.32 \cdot 10^2$	1883
10^{-7}	$5.29 \cdot 10^2$	2564
10^{-8}	$5.28 \cdot 10^2$	3444
10^{-9}	$5.28 \cdot 10^2$	4285
S2_100_60, 10^{-8}	$7.27 \cdot 10^1$	3499
S2_90_55, 10^{-8}	$5.91 \cdot 10^2$	3520
S2_80_49, 10^{-7}	$5.83 \cdot 10^3$	2319

3.4.4 Simulations on ternary images of size 600^3 and 1350^3.

The simulations with larger images show the same behaviour as those with 300^3 images. The simulation results for these images are presented in the Appendix. The simulation results for images *T1,T2* of size 600^3 are summarized in Table 3.12 for Category A and in Table 3.12 for Category B. Similarly to the simulation results for samples of size 300^3, one can observe a robust convergence of the SCoPeS-SB solver with respect to the tolerance decrease. It also can be observed that the computational time is still relatively low, especially compared to GeoDict SimpleFFT solver. GeoDict-LIR solver is not explored here, we expect similar performance as for 300^3 images. Comparing computational times reported, e.g., in Tables 3.12 and 3.9, one can see that the computational time has increased about six times, what is very good result, having in mind the increase of the size of the samples. In general, when using AMG preconditioner, one can expect the computation time to increase proportionally to the number of unknowns. Thus, eight times increase of the CPU time was expected here. The fact that the increase is only six times can be explained by the fact that the more computationally intensive task exhibits better parallel scalability. Simulations with Darcy approximation of Stokes-Brinkman model in the case of 600^3 images show similar performance as in the case of 300^3 images. Results of Darcy simulations are collected in Table 3.13 for Category A and in Table 3.15 for Category B.
Similarly, the results for *U1* sample of size 1350^3 (Category A) are summarized in Table 3.16 for the Stokes-Brinkman problem and in Table 3.17 for the Darcy problem.

Appendix for Chapter 3

3.A Simulations on S1 ternary images of size 300^3.

All three samples produced from the S1 image belong to category A, no Stokes connectivity. Thus it is expected that the results in this case will be similar to those for S2_100_60, S2_90_55 and S3_90_53, which also belong to category A. Indeed, the results from Table 3.9 below are similar to those from Table 3.1 from the main text.

TABLE 3.9: Ternary samples S1 of Category A. Permeability k_{zz}^{eff} in $mkDa$ computed with GeoDict (solvers SimpleFFT and LIR with periodic bc) and with SCoPeS-SB with pressure drop bc, $L = 0.0009$ m, nproc=8.

Sample S1_100_61, Perm of porous voxels 571.2 mkDa					
GeoDict				SCoPeS-SB	
Tol	Solver	k_{zz}^{eff}, $mkDa$	CPU, s	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$8.74 \cdot 10^1$	4471	10^{-6}	$3.42 \cdot 10^1$ (1210)
10^{-2}	SimpleFFT:	$6.16 \cdot 10^1$	61406	10^{-7}	$5.69 \cdot 10^1$ (1522)
10^{-1}	LIR	$1.24 \cdot 10^2$	33968	10^{-8}	$5.68 \cdot 10^1$ (1898)
$9.95 \cdot 10^{-2}$	LIR	$6.46 \cdot 10^1$	459561	10^{-9}	$5.68 \cdot 10^1$ (2390)
Sample S1_90_56, Perm of porous voxels 273.8 mkDa					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	CPU, s	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$1.17 \cdot 10^2$	5357	10^{-6}	$5.82 \cdot 10^1$ (1227)
$2 \cdot 10^{-2}$	SimpleFFT:	$7.11 \cdot 10^1$	70690	10^{-7}	$6.45 \cdot 10^1$ (1584)
				10^{-8}	$6.34 \cdot 10^1$ (1942)
				10^{-9}	$6.33 \cdot 10^1$ (2431)
Sample S1_80_50, Perm of porous voxels 113.3 mkDa					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	CPU, s	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$8.07 \cdot 10^1$	16532	10^{-6}	$6.42 \cdot 10^1$ (1292)
$5 \cdot 10^{-3}$	SimpleFFT:	$6.51 \cdot 10^1$	228113	10^{-7}	$6.03 \cdot 10^1$ (1628)
				10^{-8}	$7.01 \cdot 10^1$ (1991)
				10^{-9}	$6.00 \cdot 10^1$ (2393)

As mentioned above, all three samples produced from the S1 image belong to Category A. Thus, it is expected that the accuracy and the performance of the Darcy approximation in this case will be similar to those for S2_100_60, S2_90_55, and S3_90_53, which also belong to category A. Indeed, the results from Table 3.10 below are similar to those from Table 3.5 from the main text.

TABLE 3.10: SCoPeS-D results, Darcy approximation for samples S1_100_61, S1_90_56 and S1_80_50 of Category A. Permeability k_{zz}^{eff} in $mkDa$ (CPU time in s), $\varepsilon_S = 10^{-9}$, $L = 0.0009$ m, nproc=8. BC: pressure drop 1 Pa. The last line for comparison recalls permeability and CPU time when solving Stokes-Brinkman equations.

K_{Stokes}, $mkDa$	S1_100_61	S1_90_56	S1_80_50
10^5	$4.54 \cdot 10^1$ (43.2)	$4.48 \cdot 10^1$ (43.5)	$4.07 \cdot 10^1$ (42.6)
10^7	$5.70 \cdot 10^1$ (43.6)	$6.35 \cdot 10^1$ (43.6)	$6.01 \cdot 10^1$ (43.3)
10^9	$5.72 \cdot 10^1$ (44.1)	$6.39 \cdot 10^1$ (44.1)	$6.04 \cdot 10^1$ (43.6)
10^{10}	$5.72 \cdot 10^1$ (44.5)	$6.39 \cdot 10^1$ (43.7)	$6.04 \cdot 10^1$ (43.1)
SCoPeS-SB:	$5.68 \cdot 10^1$ (2390)	$6.34 \cdot 10^1$ (2431)	$6.00 \cdot 10^1$ (2393)

3.B Darcy approximation, sensitivity study

As mentioned, Darcy approximation of Stokes-Brinkman equations is used for images of Category A. Its accuracy and performance were examined before its further usage. Simulation results from sensitivity study on how the artificial permeability in the Darcy approximation influences the accuracy of the computations are presented in Table 3.11 in the Appendix. The sample S1_100_61 is considered here. This Table contains three subtables. In the first one, the fictitious permeability K_{Stokes} in the Stokes (pure fluid) voxels is fixed to a moderate value 10^{-6} m^2, and the tolerance for the iterative method is varied. In the second subtable, a relatively rough tolerance is fixed for the iterative method (10^{-4}) and the fictitious permeability K_{Stokes} is varied. Finally, in the third subtable, the tolerance 10^{-9} is fixed for the iterative method, and the fictitious permeability is varied. Comparing to the results from Table 3.11, one can see that Darcy approximation can be successfully used for fast and accurate computation of the effective permeability of samples in the case of no Stokes connectivity. The results from Table 3.11 show that relatively low value for the fictitious permeability should be set in the Stokes voxels, and the iterative method should be converged with high accuracy. The latter however, in this particular case does not influence essentially the computational time.

TABLE 3.11: SCoPeS-D results, Darcy approximation for sample S1_100_61; $L = 0.0009$ m, nproc=8. BC: pressure drop 1 Pa.

ε_S	K_{Stokes}, $mkDa$	k_{zz}^{eff}, $mkDa$	CPU time, s
10^{-4}	10^6	$5.51 \cdot 10^1$	39.2
10^{-5}	10^6	$5.51 \cdot 10^1$	40.2
10^{-6}	10^6	$5.51 \cdot 10^1$	40.6
10^{-4}	10^6	$5.51 \cdot 10^1$	39.2
10^{-4}	10^8	$5.75 \cdot 10^1$	38.8
10^{-4}	10^{10}	$2.31 \cdot 10^2$	39.3
10^{-9}	10^5	$4.54 \cdot 10^1$	43.2
10^{-9}	10^6	$5.51 \cdot 10^1$	43.4
10^{-9}	10^{11}	$5.72 \cdot 10^1$	44.4

3.C Simulations on ternary images of size 600^3 and 1350^3

TABLE 3.12: Ternary samples T1 and T2 with no Stokes connectivity. Permeability k_{zz}^{eff} in $mkDa$ computed with GeoDict (solvers SimpleFFT and LIR with periodic bc) and with SCoPeS-SB with pressure drop bc, $L = 0.0018$ m, nproc=8.

Sample T1_100_59, Perm of porous voxels 425.6 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
$2.2 \cdot 10^{-1}$	SimpleFFT	$6.49 \cdot 10^1$	170969	10^{-6}	$5.61 \cdot 10^1$ (6082)
				10^{-7}	$3.33 \cdot 10^1$ (8148)
				10^{-8}	$3.37 \cdot 10^1$ (10538)
				10^{-9}	$3.38 \cdot 10^1$ (13027)
Sample T1_90_54, Perm of porous voxels 204.0 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
$7.2 \cdot 10^{-1}$	SimpleFFT:	$1.41 \cdot 10^2$	52369	10^{-6}	$3.38 \cdot 10^1$ (6059)
				10^{-7}	$5.51 \cdot 10^1$ (8035)
				10^{-8}	$5.48 \cdot 10^1$ (10240)
Sample T2_100_58, Perm of porous voxels 367.4 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
$2 \cdot 10^{-1}$	SimpleFFT:	$4.25 \cdot 10^1$	1.2509e+6	10^{-6}	$3.61 \cdot 10^1$ (6191)
				10^{-7}	$3.35 \cdot 10^1$ (8320)
				10^{-8}	$3.36 \cdot 10^1$ (10425)
Sample T2_90_54, Perm of porous voxels 204.0 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
$2.4 \cdot 10^{-1}$	SimpleFFT:	$9.32 \cdot 10^1$	815269	10^{-6}	$5.94 \cdot 10^1$ (6333)
				10^{-7}	$6.38 \cdot 10^1$ (8371)
				10^{-8}	$6.51 \cdot 10^1$ (10456)
				10^{-9}	$6.51 \cdot 10^1$ (13191)

TABLE 3.13: SCoPeS-D results, Darcy approximation for samples T1_100_59, T1_90_54, T2_100_58 and T2_90_54. Permeability k_{zz}^{eff} in $mkDa$ (CPU time in s), $\varepsilon_S = 10^{-9}$, $L = 0.0018$ m, nproc=8. BC: pressure drop 1 Pa. The last line for comparison recalls permeability and CPU time when solving Stokes-Brinkman equations.

K_{Stokes}, $mkDa$	T1_100_59	T1_90_54	T2_100_58	T2_90_54
10^5	$2.58 \cdot 10^1$ (260)	$2.18 \cdot 10^1$ (319)	$2.17 \cdot 10^1$ (321)	$2.36 \cdot 10^1$ (322)
10^7	$3.39 \cdot 10^1$ (255)	$5.84 \cdot 10^1$ (318)	$3.36 \cdot 10^1$ (330)	$7.23 \cdot 10^1$ (327)
10^9	$3.40 \cdot 10^1$ (264)	$6.21 \cdot 10^1$ (325)	$3.39 \cdot 10^1$ (334)	$7.77 \cdot 10^1$ (332)
10^{11}	$3.41 \cdot 10^1$ (269)	$6.22 \cdot 10^1$ (320)	$3.39 \cdot 10^1$ (334)	$7.77 \cdot 10^1$ (331)
SCoPeS-SB:	$3.30 \cdot 10^1$ (9472)	$4.33 \cdot 10^1$ (10425)	$3.36 \cdot 10^1$ (10425)	$6.51 \cdot 10^1$ (10456)

TABLE 3.14: Ternary samples T1 and T2 of Category B. Permeability k_{zz}^{eff} in $mkDa$ computed with GeoDict (solvers SimpleFFT and LIR with periodic bc) and with SCoPeS-SB with pressure drop bc, $L = 0.0018$ m, nproc=8.

Sample T1_80_49, Perm of porous voxels 97.8 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT	$6.45 \cdot 10^2$	155062	10^{-5}	$5.68 \cdot 10^2$ (4849)
$1.3 \cdot 10^{-2}$	SimpleFFT	$6.01 \cdot 10^2$	834989	10^{-6}	$6.06 \cdot 10^2$ (6208)
				10^{-7}	$5.88 \cdot 10^2$ (8215)
				10^{-8}	$5.88 \cdot 10^2$ (10460)

Sample T2_80_49, Perm of porous voxels 97.8 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver:	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
10^{-1}	SimpleFFT:	$6.59 \cdot 10^3$	6087	10^{-4}	$1.94 \cdot 10^3$ (4238)
$2.6 \cdot 10^{-2}$	SimpleFFT	$1.59 \cdot 10^3$	682309	10^{-5}	$1.49 \cdot 10^3$ (5352)
				10^{-6}	$1.57 \cdot 10^3$ (6899)
				10^{-7}	$1.57 \cdot 10^3$ (8801)

TABLE 3.15: SCoPeS-D results, Darcy approximation for samples S2_90_55, S2_80_49 and S3_80_48. Permeability k_{zz}^{eff} in $mkDa$ (CPU time in s), $\varepsilon_S = 10^{-9}$, $L = 0.0018$ m, nproc=8. BC: pressure drop 1 Pa. The last line for comparison recalls permeability and CPU time when solving Stokes-Brinkman equations.

K_{Stokes}, $mkDa$	T1_80_49	T2_80_49
10^5	$2.64 \cdot 10^1$ (314)	$3.45 \cdot 10^1$ (320)
10^7	$1.04 \cdot 10^3$ (320)	$1.65 \cdot 10^3$ (321)
10^9	$1.02 \cdot 10^5$ (325)	$1.60 \cdot 10^5$ (322)
10^{11}	$1.02 \cdot 10^7$ (323)	$1.60 \cdot 10^7$ (327)
SCoPeS-SB:	$5.89 \cdot 10^2$ (8215)	$1.57 \cdot 10^3$ (6899)

TABLE 3.16: Ternary sample U1 with no Stokes connectivity. Permeability k_{zz}^{eff} in $mkDa$ computed with GeoDict (solvers SimpleFFT with periodic bc) and with SCoPeS-SB with pressure drop bc, $L = 0.0027$ m, nproc=8.

Sample U1_100_59, Perm of porous voxels 425.60 $mkDa$					
GeoDict				SCoPeS-SB	
Tol	Solver	k_{zz}^{eff}, $mkDa$	(CPU, s)	ε_S	k_{zz}^{eff}, $mkDa$ (CPU, s)
$3.92 \cdot 10^{-1}$	SimpleFFT	$9.17 \cdot 10^{1}$	1.11362e+06	10^{-6}	$3.34 \cdot 10^{1}$ (99841)
10^{-1} (nproc=16)	SimpleFFT	$6.06 \cdot 10^{1}$	865126	10^{-7}	$3.29 \cdot 10^{1}$ (129588)
10^{-1}	LIR	divergence		10^{-8}	$3.28 \cdot 10^{1}$ (165821)
				10^{-9}	$3.28 \cdot 10^{1}$ (205090)

TABLE 3.17: SCoPeS-D results, Darcy approximation for sample U1. Permeability k_{zz}^{eff} in $mkDa$ (CPU time in s), $\varepsilon_S = 10^{-9}$, $L = 0.0009$ m, nproc=8. BC: pressure drop 1 Pa. The last line for comparison recalls permeability and CPU time when solving Stokes-Brinkman equations.

K_{Stokes}^{-1}, $mkDa$	U1_100_59
10^5	$2.35 \cdot 10^{1}$ (7255)
10^7	$3.34 \cdot 10^{1}$ (7373)
10^9	$3.37 \cdot 10^{1}$ (7537)
10^{11}	$3.37 \cdot 10^{1}$ (7437)
SCoPeS-SB	$3.28 \cdot 10^{1}$ (205090)

Chapter 4

Navier-Stokes equations in tight porous media

4.1 Section outline

In Chapter 2, we have shown that the SIMPLE preconditioner should be used instead of the established Uzawa preconditioner when solving the Stokes equations in domains with high surface-to-volume ratio. In the present Chapter, we show that these results are also valid for the Navier-Stokes equations in the case of low Reynolds numbers. Indeed, it is known that the identity preconditioner (e.g. the pressure mass matrix) can be used to approximate the (non-symmetric) Schur complement matrix for the Oseen equations in the case of low Reynolds number (see, e.g. [23], page 71). However, we claim that the well-known SIMPLE preconditioner should be used instead of the identity for domains with high surface-to-volume ratio.
The outline of the Chapter is as follows:

1. In Section 4.3, we formulate the BiCG-SIMPLE method, which is an analogue of the CG-SIMPLE Algorithm 1 adapted for the non-symmetric Oseen problem for low Reynolds numbers.

2. In Section 4.4, we perform validation of the developed Picard-BiCG-SIMPLE solver for 3D binary image with ≈ 350 mln. of unknowns for low Reynolds numbers.

3. In Section 4.5, we describe the finite-difference discretization on fully-staggered grids in the case of simplest geometry.

4. Also, an analogue of the Pressure Convection-Diffusion (PCD) preconditioner for domains with high surface-to-volume ratio is proposed in Section 4.6 for higher Reynolds numbers.

4.2 Problem statement

For higher flow rates, it's essential to consider influence of the inertial forces. The flow of fast laminar incompressible fluid is governed by the Navier-Stokes equations:

$$\begin{aligned} -\mu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega_h^f, \\ -\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega_h^f, \end{aligned} \tag{4.1}$$

where Ω_h^f represents a porous space where fluid propagates (see Section 2.1 for the Stokes equations). In order to formulate the underlying BVP for the equations (4.1),

the same set of boundary conditions as for the Stokes problem discussed in Chapter 2 can be considered.

Nonlinear iteration. Solving the Navier–Stokes equations (4.1) requires nonlinear iteration with a linearized problem being solved at every step. Thus, given an "initial guess" $\mathbf{u}_0$, a sequence of iterates $\mathbf{u}_1$, $\mathbf{u}_2$, ... is computed, which is supposed to converge to the solution $\mathbf{u}$ of the nonlinear problem. In our practical computations, we consider the Picard linearization of the Navier-Stokes equations [64]. For the Picard iteration, on each outer nonlinear iteration we solve the Oseen's problem, which is given as follows:

$$\begin{aligned} -\mu\Delta\mathbf{u}_k + (\mathbf{u}_{k-1}\cdot\nabla)\mathbf{u}_k + \nabla p_k &= \mathbf{f} \quad \text{in } \Omega_h^f, \\ -\nabla\cdot\mathbf{u}_k &= 0 \quad \text{in } \Omega_h^f, \end{aligned} \tag{4.2}$$

where $\mathbf{u}_{k-1}$ denotes velocity from the previous timestep and $(\mathbf{u}_k, p_k)^T$ is the solution to be computed.

Additionally, we consider the Newton iteration for which it is required to solve the full Jacobian (also known as the exact Frechet-derivative) of the Navier-Stokes equations [20, 64]:

$$\begin{aligned} -\mu\Delta\mathbf{u}_k + (\mathbf{u}_{k-1}\cdot\nabla)\mathbf{u}_k + (\mathbf{u}_k\cdot\nabla)\mathbf{u}_{k-1} + \nabla p_k &= \mathbf{r}_{k-1} \quad \text{in } \Omega_h^f, \\ -\nabla\cdot\mathbf{u}_{k-1} &= 0 \quad \text{in } \Omega_h^f, \end{aligned} \tag{4.3}$$

where $\mathbf{r}_{k-1}$ is the residual from $k-1$ step. Note, in the Newton iteration (4.3) there is an additional term $(\mathbf{u}_{k-1}\cdot\nabla)\mathbf{u}_k$ comparing with the Picard iteration (4.2).

In what follows, for simplicity of notation, we denote $\mathbf{u}_{k-1} = \tilde{\mathbf{u}}$.

4.3 Iterative method and preconditioning: Picard-BiCG-SIMPLE algorithm

The Picard-BiCG-SIMPLE algorithm consists in firstly applying the Picard linearization and secondly applyting the BiCG-SIMPLE algorithm for solving the resulting linear Oseen problem (4.2). After finite-difference discretization of the Oseen equations (4.2) using fully-staggered grids, we obtain the following system:

$$\mathbb{A}\begin{bmatrix} u_h \\ p_h \end{bmatrix} = \begin{bmatrix} f_h \\ g_h \end{bmatrix}, \quad \mathbb{A} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & 0 \end{bmatrix}, \tag{4.4}$$

where the discretization of the momentum equation is:

$$\mathbf{F} = \mathbf{A} + \hat{\mathbf{N}}, \tag{4.5}$$

where $\hat{\mathbf{N}}$ is the upwind discretization of the convection term $(\mathbf{u}_{k-1}\cdot\nabla)\mathbf{u}_k$. We describe the discretization in the case of square geometry in Section 4.5. Note, the discretization for general voxel-based geometries can be found in [38].

The Schur complement matrix from (2.5) becomes:

$$S = \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T = \mathbf{B}(\mathbf{A} + \hat{\mathbf{N}})^{-1}\mathbf{B}^T, \tag{4.6}$$

and the SIMPLE preconditioner becomes:

$$\hat{S}_{\text{simple}} = \mathbf{B}\hat{\mathbf{F}}_{\text{simple}}^{-1}\mathbf{B}^T, \quad \hat{\mathbf{F}}_{\text{simple}} = \text{diag}(\mathbf{A} + \hat{\mathbf{N}}). \tag{4.7}$$

Note, in-spite the preconditioner $\hat{S}_{\text{simple}}$ is symmetric, it reflects the non-symmetry of the matrix S from (4.6) since for the upwind differences, we have $\text{diag}(\mathbf{A} + \hat{\mathbf{N}}) \neq \text{diag}(\mathbf{A})$ (but not for the central differences). Since the matrix S is non-symmetric, we apply the BiCG Krylov subspace method instead of the CG to solve the Schur complement system (2.4). The resulting BiCG-SIMPLE algorithm is presented in Algorithm 2. It should be noted, that when applying S on each outer iteration, we need to solve the momentum equation with the convection-diffusion matrix $(\mathbf{A} + \hat{\mathbf{N}})$, instead of the diffusion matrix $\mathbf{A}$. The Algebraic Multigrid method works well when the upwind discretization is used (see, e.g. [20]).

Algorithm 2 Preconditioned BiCG, adapted from [50] (Sec. 7.1).

Require: tolerance ε_S, initial guess p_h^0, r_h^* arbitrary vector
Ensure: Approximate solution p_h for the system (4.6)
Compute the initial residual $r_0 = g_h - Sp_h^0$
Set $d_h^0 = r_h^0$, $k = 0$
while not converged **do**
  Solve $\hat{S}\hat{d}_h^k = d_h^k$ for $\hat{d}_h^k$
  Apply $S\hat{d}_h^k = q_h^k$ for q_h^k
  $\alpha_k = \dfrac{(r_h^k)^T r_h^*}{(q_h^k)^T r_h^*}$
  $s_h = r_h^k - \alpha_k q_h^k$
  Solve $\hat{S}\hat{s}_h = s_h$ for $\hat{s}_h$
  Apply $S\hat{s}_h = t_h$ for t_h
  $\omega_k = \dfrac{(s_h)^T t_h}{(t_h)^T t_h}$
  $p_h^{k+1} = p_h^k + \alpha_k \hat{d}_h^k + \omega_k \hat{s}_h$
  $r_h^{k+1} = s_h - \omega_k t_h$
  If $\|r_h^{k+1}\| < \varepsilon_S$ (unprec), exit loop
  $\beta_k = \dfrac{(r_h^{k+1})^T r_h^*}{(r_h^k)^T r_h^*} \times \dfrac{\alpha_k}{\omega_k}$
  $d_h^{k+1} = r_h^{k+1} + \beta_k(d_h^k - \omega_k q_h^k)$, $k = k + 1$
end while
Return p_h^{k+1} as the approximate solution

4.4 Validation of the developed Navier-Stokes solver

In this Section, we validate the developed Picard-BiCG-SIMPLE algorithm for 3D binary images. Firstly, we consider the sample S of the size 300^3, porosity $\phi = 21\%$ described in Table 2.1 and pictured in Fig. 2.1. In this computational experiment, the periodic boundary conditions are imposed in the flow direction z, and the no-slip boundary conditions are imposed in the tangential directions. We consider $L = 1[m]$ and compute average velocity for the unit pressure jump $dp = 1[Pa]$ for different values of the viscosity μ. The computed average velocity is validated against the Geodict (LIR) solver for the Navier-Stokes equations. For outer iterations, we use $\varepsilon_S = 10^{-2}$ and for inner iteration we use $\varepsilon_A = \varepsilon_{\hat{S}} = 10^{-3}$. The results are presented in Table 4.1 for viscosities $\mu = \{1 \cdot 10^{-3}, 5 \cdot 10^{-4}, 1 \cdot 10^{-4}\}$; the first row in the Table corresponds to the Stokes approximation of the Navier-Stokes equations. Additionally, we provide the respective values of the Reynolds number computed with Geodict. It can be seen

from the Table, that the computed average velocities are in a good agreement with validated software Geodict.

TABLE 4.1: Validation of the Picard-BiCG-SIMPLE Algorithm 2 for the sample S described in Table 2.1, $\varepsilon_S = 10^{-2}$, $dp = 1[Pa]$, $h = 1/300[m]$. The number of outer Picard iterations is shown in brackets.

Viscosity, $\mu[Pa \cdot s]$	Re	Geodict(LIR)	Picard-BICG-SIMPLE
$1 \cdot 10^{-3}$ (Stokes)	-	0.0042	(0 it) 0.0042
$1 \cdot 10^{-3}$	0.163	0.0039	(1 it) 0.0039
$5 \cdot 10^{-4}$	0.56	0.0068	(3 it) 0.0069
$1 \cdot 10^{-4}$	5.54	0.0134	(5 it) 0.0133

TABLE 4.2: Validation of the Picard-BiCG-SIMPLE Algorithm 2 for the high-porosity filter pictured in Fig. 4.1; sample size $779 \times 801 \times 256$, porosity $\phi = 51\%$, $\varepsilon_S = 10^{-2}$, $dp = 1[Pa]$, $h = 1/256[m]$.

Viscosity, $\mu[Pa \cdot s]$	Re	Geodict(LIR)	Picard-BICG-SIMPLE
$1 \cdot 10^{-3}$ (Stokes)	-	0.040	(0 it) 0.040
$1 \cdot 10^{-3}$	2.15	0.029	(1 it) 0.029
$5 \cdot 10^{-4}$	5.81	0.039	(3 it) 0.038

Secondly, we consider the high-porosity filter picture in Fig. 4.1 (solid is shown in grey colour). The size of the filter is $779 \times 801 \times 256$, and the porosity is $\phi = 52\%$, which corresponds to the 82.5 mln. of active voxels that is nearly 350 mln. of unknowns. The periodic boundary conditions are again imposed in the flow direction z, and the no-slip boundary conditions are imposed in the tangential directions. We consider $L = 1[m]$ and similarly to the previous experiment compute average velocity for the unit pressure jump $dp = 1[Pa]$ for different values of the viscosity μ. The computed average velocity is validated against the Geodict (LIR) solver for the Navier-Stokes equations. For outer iterations, we use $\varepsilon_S = 10^{-2}$ and for inner iteration we use $\varepsilon_A = \varepsilon_{\hat{S}} = 10^{-3}$. The results are presented in Table 4.2 for viscosities $\mu = \{1 \cdot 10^{-3}, 5 \cdot 10^{-4}\}$; the first row in the Table corresponds to the Stokes approximation of the Navier-Stokes equations. It can be seen from the Table, that the computed average velocities are in a good agreement with validated software Geodict.
For both considered samples, the respective Reynolds numbers are small, which justifies using the symmetric SIMPLE preconditioner. For higher Reynolds numbers, the non-symmetric preconditioner might become a necessity.

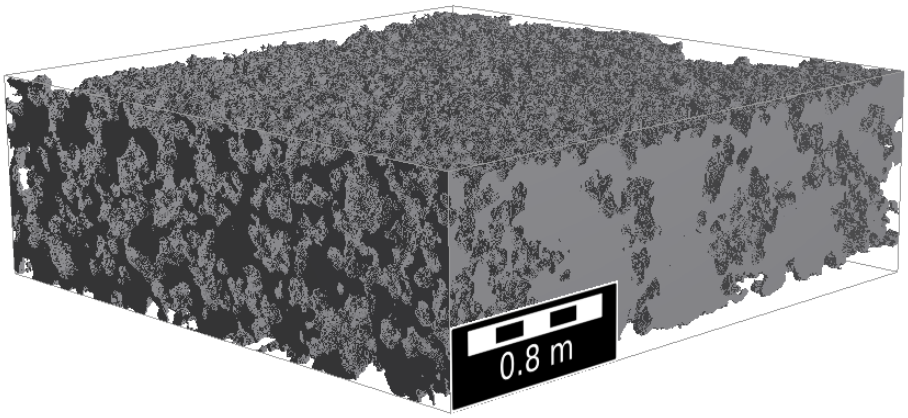


FIGURE 4.1: High-porosity filter of the size $779 \times 801 \times 256$, porosity $\phi = 51\%$, 82.5 mln. active voxels. Solid is shown in grey colour, fluid is invisible.

4.5 Discretization of the convection term

In Section 2.6, we discretized the Stokes equations on tensor-structured in the case of square domain. In the present Section, we proceed to the Navier-Stokes equations by additionally discretizing the convection term. It's worth noting, that the convection term is identical for both the primary and auxiliary BVPs described in Section 2.6.1.
As an example of the Navier-Stokes solution in the square domain, we show the velocity solution for the lid-driven cavity problem (Re=1000) in Fig. 4.2, and the corresponding stream function is shown in Fig. 4.3.
The convection term can be expressed in multiple formulations. Below we consider its gradient, divergence, mixed, and rotational forms. These are also called primitive, conservative, skew-symmetric, and rotational forms [89].

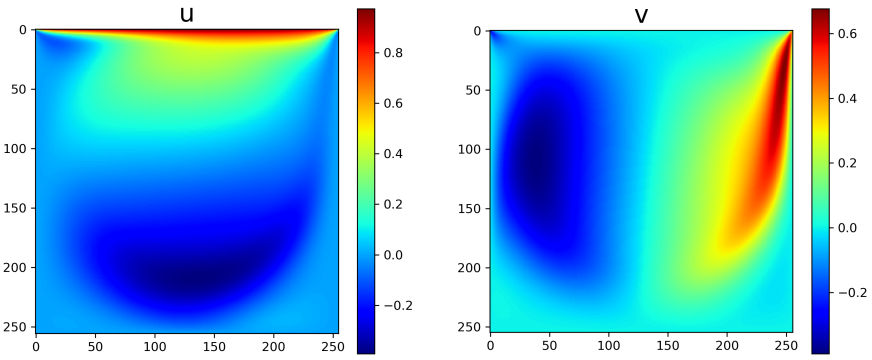


FIGURE 4.2: Lid-driven cavity problem. Velocity solution of the Navier-Stokes equations computed for $\mathrm{n} = 256$, $L = 1$, $\mu = 1/1000$.

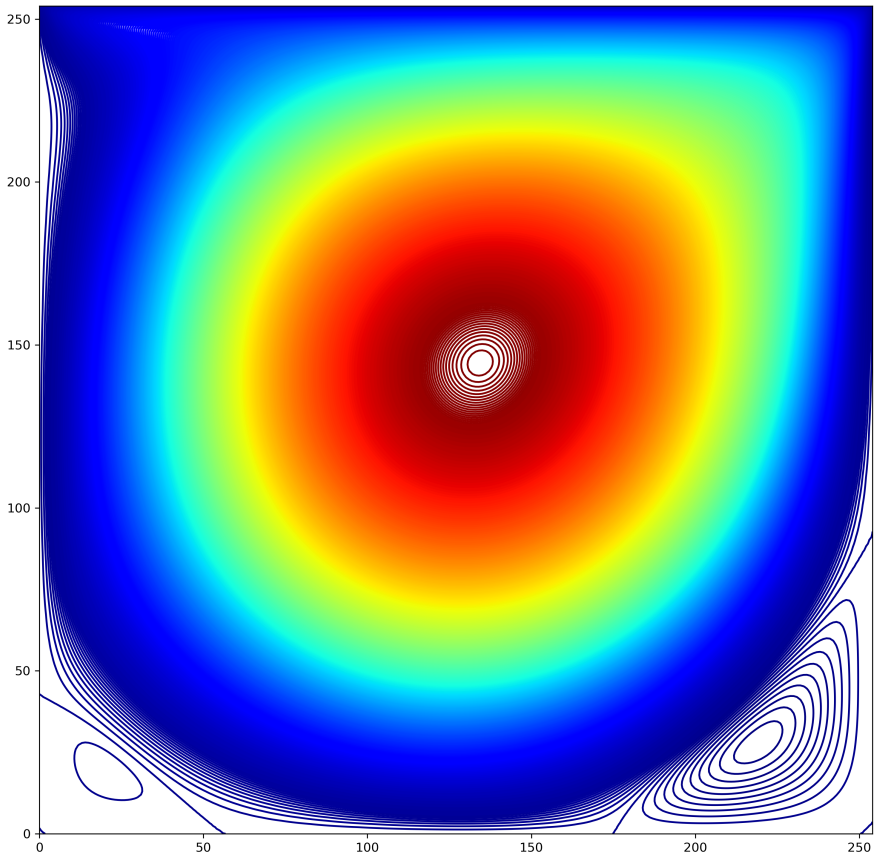


FIGURE 4.3: Lid driven cavity problem. Stream function (1200 levels) corresponding to the velocity from Fig. 4.2 computed for $\mathrm{n} = 256$, $L = 1$, $\mu = 1/1000$.

Gradient form. For a given velocity field $\tilde{\mathbf{u}} = (\tilde{u}, \tilde{v})^T$, we consider the first order convection operator $\mathcal{N}^{\text{grad}}(\tilde{\mathbf{u}})$, defined as follows:

$$\mathcal{N}^{\text{grad}}(\tilde{\mathbf{u}})\mathbf{u} = (\tilde{\mathbf{u}} \cdot \nabla)\mathbf{u} = (\tilde{\mathbf{u}} \cdot \nabla u, \tilde{\mathbf{u}} \cdot \nabla v)^T = (\tilde{u}u_x + \tilde{v}u_y, \tilde{u}v_x + \tilde{v}v_y)^T, \tag{4.8}$$

and the zeroth order reaction operator $\mathcal{R}^{\text{grad}}(\tilde{\mathbf{u}})$, defined as follows:

$$\mathcal{R}^{\text{grad}}(\tilde{\mathbf{u}})\mathbf{u} = (\mathbf{u} \cdot \nabla)\tilde{\mathbf{u}} = (\mathbf{u} \cdot \nabla \tilde{u}, \mathbf{u} \cdot \nabla \tilde{v})^T = (u\tilde{u}_x + v\tilde{u}_y, u\tilde{v}_x + v\tilde{v}_y)^T. \tag{4.9}$$

The superscript $^{\text{grad}}$ stands for the *gradient form* here.
Divergence form. Alternatively, the convective operator $\mathcal{N}^{\text{grad}}(\tilde{\mathbf{u}})$ can be equivalently written as follows:

$$\mathcal{N}^{\text{div}}(\tilde{\mathbf{u}})\mathbf{u} = (\nabla \cdot (\tilde{\mathbf{u}}u), \nabla \cdot (\tilde{\mathbf{u}}v))^T = ((\tilde{u}u)_x + (\tilde{v}u)_y, (\tilde{u}v)_x + (\tilde{v}v)_y)^T, \tag{4.10}$$

and the reaction operator $\mathcal{R}^{\text{grad}}(\tilde{\mathbf{u}})$ can be equivalently written as follows:

$$\mathcal{R}^{\text{div}}(\tilde{\mathbf{u}})\mathbf{u} = (\nabla \cdot (\mathbf{u}\tilde{u}), \nabla \cdot (\mathbf{u}\tilde{v}))^T = ((u\tilde{u})_x + (v\tilde{u})_y, (u\tilde{v})_x + (v\tilde{v})_y)^T. \tag{4.11}$$

The superscript $^{\text{div}}$ stands for the *divergence form* here. Note, under the divergency constraints $\nabla \cdot \tilde{\mathbf{u}} = 0$ and $\nabla \cdot \mathbf{u} = 0$, we have:

$$\mathcal{N}^{\text{grad}}(\tilde{\mathbf{u}})\mathbf{u} = \mathcal{N}^{\text{div}}(\tilde{\mathbf{u}})\mathbf{u} \quad \text{and} \quad \mathcal{R}^{\text{grad}}(\tilde{\mathbf{u}})\mathbf{u} = \mathcal{R}^{\text{div}}(\tilde{\mathbf{u}})\mathbf{u},$$

respectively. For the nonlinear convection term, we have:

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = \mathcal{N}^{\text{grad}}(\mathbf{u})\mathbf{u} = \mathcal{N}^{\text{div}}(\mathbf{u})\mathbf{u} = \mathcal{R}^{\text{grad}}(\mathbf{u})\mathbf{u} = \mathcal{R}^{\text{div}}(\mathbf{u})\mathbf{u}.$$

In the literature, a mixed form can often be seen (see, e.g. [90]), obtained by averaging the gradient and divergence forms. This is given by:

$$\begin{aligned} 2(\mathbf{u} \cdot \nabla)\mathbf{u} &= (\mathbf{u} \cdot \nabla u + \nabla \cdot (\mathbf{u}u), \mathbf{u} \cdot \nabla v + \nabla \cdot (\mathbf{u}v))^T \\ &= \mathcal{N}^{\text{grad}} + \mathcal{N}^{\text{div}} = \mathcal{R}^{\text{grad}} + \mathcal{R}^{\text{div}} = \mathcal{N}^{\text{grad}} + \mathcal{R}^{\text{div}} = \mathcal{R}^{\text{grad}} + \mathcal{N}^{\text{div}}. \end{aligned} \tag{4.12}$$

This form of the convection operator leads to the skew-self-adjoint operator used, for example, in [91],[92].
Rotational form. In the context of Helmholtz-Hodge decomposition, the rotational form (see, e.g. [93], [94], [95]) is more suited. Firstly, the convection term in the gradient form $\mathcal{N}^{\text{grad}}(\tilde{\mathbf{u}})$ and in the divergence form $\mathcal{N}^{\text{div}}(\tilde{\mathbf{u}})$ under the divergency-constraint, can be equivalently written as follows:

$$\mathcal{N}^{\text{rot}}(\tilde{\mathbf{u}})\mathbf{u} = \frac{1}{2}(\nabla(\tilde{\mathbf{u}} \cdot \mathbf{u}) - [\mathbf{u} \times (\nabla \times \tilde{\mathbf{u}}) + \tilde{\mathbf{u}} \times (\nabla \times \mathbf{u}) + \nabla \times (\tilde{\mathbf{u}} \times \mathbf{u})]), \tag{4.13}$$

where the cross product $\mathbf{u} \times q = (-vq, uq)^T$ in 2D is defined similarly to $\nabla \times q$ from (2.55). Similarly, the reaction term in the gradient form $\mathcal{R}^{\text{grad}}(\tilde{\mathbf{u}})$ and in the divergence form $\mathcal{R}^{\text{div}}(\tilde{\mathbf{u}})$, can be equivalently written as follows:

$$\begin{aligned} \mathcal{R}^{\text{rot}}(\tilde{\mathbf{u}})\mathbf{u} &= \frac{1}{2}(\nabla(\mathbf{u} \cdot \tilde{\mathbf{u}}) - [\tilde{\mathbf{u}} \times (\nabla \times \mathbf{u}) + \mathbf{u} \times (\nabla \times \tilde{\mathbf{u}}) + \nabla \times (\mathbf{u} \times \tilde{\mathbf{u}})]) \\ &= \frac{1}{2}(\nabla(\tilde{\mathbf{u}} \cdot \mathbf{u}) - [\mathbf{u} \times (\nabla \times \tilde{\mathbf{u}}) + \tilde{\mathbf{u}} \times (\nabla \times \mathbf{u}) - \nabla \times (\tilde{\mathbf{u}} \times \mathbf{u})]). \end{aligned} \tag{4.14}$$

Finally, for the convection-reaction term from the Jacobian (4.3), we have:

$$\begin{aligned}(\tilde{\mathbf{u}}\cdot\nabla)\mathbf{u}+(\mathbf{u}\cdot\nabla)\tilde{\mathbf{u}} &= \mathcal{N}^{\mathrm{grad}}(\tilde{\mathbf{u}})\mathbf{u}+\mathcal{R}^{\mathrm{grad}}(\tilde{\mathbf{u}})\mathbf{u}\\ &= \mathcal{N}^{\mathrm{div}}(\tilde{\mathbf{u}})\mathbf{u}+\mathcal{R}^{\mathrm{div}}(\tilde{\mathbf{u}})\mathbf{u} = \mathcal{N}^{\mathrm{rot}}(\tilde{\mathbf{u}})\mathbf{u}+\mathcal{R}^{\mathrm{rot}}(\tilde{\mathbf{u}})\mathbf{u}\\ &= \nabla(\tilde{\mathbf{u}}\cdot\mathbf{u})-[\mathbf{u}\times(\nabla\times\tilde{\mathbf{u}})+\tilde{\mathbf{u}}\times(\nabla\times\mathbf{u})].\end{aligned} \tag{4.15}$$

Hence, the nonlinear convection term in the rotational form looks as follows:

$$(\mathbf{u}\cdot\nabla)\mathbf{u}=\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u})-\mathbf{u}\times(\nabla\times\mathbf{u}).$$

Discretization using centered differences

The discretization of the convection term on the fully staggered grids (2.42) requires averaging. For example, the convection term written in the divergence form (4.10) contains products $\tilde{v}u$ and $\tilde{u}v$ of the functions living on different grids $\mathcal{U}_h$ and $\mathcal{V}_h$. All averaging operators can be assembled from a single 1d matrix A, which is similar to the 1d derivative matrix B from (2.54):

$$\mathrm{A}=\frac{1}{2}\begin{pmatrix}1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1\\ & & & 1\end{pmatrix},\quad \mathrm{A}\in\mathbb{R}^{\mathrm{n}\times(\mathrm{n}-1)}.$$

Firstly, the discrete co-directional averaging of the velocity, denoted $\mathrm{A}_{\mathrm{x}}^{u}:\mathcal{U}_h\to\mathcal{P}_h$ and $\mathrm{A}_{\mathrm{y}}^{v}:\mathcal{V}_h\to\mathcal{P}_h$, are assembled as follows:

$$\begin{aligned}\mathrm{A}_{\mathrm{x}}^{u}&=I^{\overline{\omega}_h}\otimes\mathrm{A},\quad \mathrm{A}_{\mathrm{x}}^{u}\in\mathbb{R}^{\mathrm{nn}\times\mathrm{n}(\mathrm{n}-1)},\\ \mathrm{A}_{\mathrm{y}}^{v}&=\mathrm{A}\otimes I^{\overline{\omega}_h},\quad \mathrm{A}_{\mathrm{y}}^{v}\in\mathbb{R}^{\mathrm{nn}\times(\mathrm{n}-1)\mathrm{n}},\end{aligned}$$

and the discrete averaging of the velocity curl, denoted $\mathrm{A}_{\mathrm{x}}^{q}:\mathcal{Q}_h\to\mathcal{V}_h$ and $\mathrm{A}_{\mathrm{y}}^{q}:\mathcal{Q}_h\to\mathcal{U}_h$, are assembled as follows:

$$\begin{aligned}\mathrm{A}_{\mathrm{x}}^{q}&=I^{\omega_h}\otimes\mathrm{A},\quad \mathrm{A}_{\mathrm{x}}^{q}\in\mathbb{R}^{(\mathrm{n}-1)\mathrm{n}\times(\mathrm{n}-1)(\mathrm{n}-1)},\\ \mathrm{A}_{\mathrm{y}}^{q}&=\mathrm{A}\otimes I^{\omega_h},\quad \mathrm{A}_{\mathrm{y}}^{q}\in\mathbb{R}^{\mathrm{n}(\mathrm{n}-1)\times(\mathrm{n}-1)(\mathrm{n}-1)}.\end{aligned}$$

Next, the averaging operators from pressure to velocity grids $\mathrm{A}_{\mathrm{x}}^{p}:\mathcal{P}_h\to\mathcal{U}_h$, $\mathrm{A}_{\mathrm{y}}^{p}:\mathcal{P}_h\to\mathcal{V}_h$ are given as follows:

$$\mathrm{A}_{\mathrm{x}}^{p}=(\mathrm{A}_{\mathrm{x}}^{u})^{T},\quad \mathrm{A}_{\mathrm{y}}^{p}=(\mathrm{A}_{\mathrm{y}}^{v})^{T}, \tag{4.16}$$

and the averaging operators from velocities to curl $\mathrm{A}_{\mathrm{y}}^{u}:\mathcal{U}_h\to\mathcal{Q}_h$, $\mathrm{A}_{\mathrm{x}}^{v}:\mathcal{V}_h\to\mathcal{Q}_h$ are given as follows:

$$\mathrm{A}_{\mathrm{y}}^{u}=(\mathrm{A}_{\mathrm{y}}^{q})^{T},\quad \mathrm{A}_{\mathrm{x}}^{v}=(\mathrm{A}_{\mathrm{x}}^{q})^{T}. \tag{4.17}$$

The discretization $\mathcal{N}_h^{\mathrm{grad}}(\tilde{\mathbf{u}}_h):(\mathcal{U}_h,\mathcal{V}_h)^T\to(\mathcal{U}_h,\mathcal{V}_h)^T$ of the convection term $\mathcal{N}^{\mathrm{grad}}(\tilde{\mathbf{u}})$ from (4.8) is then obtained as follows:

$$\begin{aligned}&\mathcal{N}_h^{\mathrm{grad}}(\tilde{\mathbf{u}}_h)=\\ &=\begin{bmatrix}\mathrm{A}_{\mathrm{x}}^{p}\mathrm{diag}(\mathrm{A}_{\mathrm{x}}^{u}\tilde{u}_h)\mathrm{B}_{\mathrm{x}}^{u}+\mathrm{A}_{\mathrm{y}}^{q}\mathrm{diag}(\mathrm{A}_{\mathrm{x}}^{v}\tilde{v}_h)\mathrm{B}_{\mathrm{y}}^{u} & \\ & \mathrm{A}_{\mathrm{x}}^{q}\mathrm{diag}(\mathrm{A}_{\mathrm{y}}^{u}\tilde{u}_h)\mathrm{B}_{\mathrm{x}}^{v}+\mathrm{A}_{\mathrm{y}}^{p}\mathrm{diag}(\mathrm{A}_{\mathrm{y}}^{v}\tilde{v}_h)\mathrm{B}_{\mathrm{y}}^{v}\end{bmatrix},\end{aligned} \tag{4.18}$$

and the discretization $\mathcal{R}_h^{\mathrm{grad}}(\tilde{\mathbf{u}}_h) : (\mathcal{U}_h, \mathcal{V}_h)^T \to (\mathcal{U}_h, \mathcal{V}_h)^T$ of the reaction term $\mathcal{R}^{\mathrm{grad}}(\tilde{\mathbf{u}})$ from (4.9) as follows:

$$\mathcal{R}_h^{\mathrm{grad}}(\tilde{\mathbf{u}}_h) = \begin{bmatrix} \mathrm{A}_{\mathrm{x}}^p \mathrm{diag}(\mathrm{B}_{\mathrm{x}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^u & \mathrm{A}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{B}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v \\ \mathrm{A}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{B}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & \mathrm{A}_{\mathrm{y}}^p \mathrm{diag}(\mathrm{B}_{\mathrm{y}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^v \end{bmatrix}.$$

Similarly, for the discretizations $\mathcal{N}_h^{\mathrm{div}}(\tilde{\mathbf{u}}_h)$ and $\mathcal{R}_h^{\mathrm{div}}(\tilde{\mathbf{u}}_h)$ of the convection and reaction operators $\mathcal{N}^{\mathrm{div}}(\tilde{\mathbf{u}})$ and $\mathcal{N}^{\mathrm{div}}(\tilde{\mathbf{u}})$ from (4.10) and (4.11), we have:

$$\begin{aligned} &\mathcal{N}_h^{\mathrm{div}}(\tilde{\mathbf{u}}_h) = \\ &= \begin{bmatrix} \mathrm{B}_{\mathrm{x}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^u + \mathrm{B}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & \\ & \mathrm{B}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v + \mathrm{B}_{\mathrm{y}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^v \end{bmatrix}, \end{aligned}$$

and:

$$\mathcal{R}_h^{\mathrm{div}}(\tilde{\mathbf{u}}_h) = \begin{bmatrix} \mathrm{B}_{\mathrm{x}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^u & \mathrm{B}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v \\ \mathrm{B}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & \mathrm{B}_{\mathrm{y}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^v \end{bmatrix},$$

respectively. To discretize the convection term $\mathcal{N}^{\mathrm{rot}}(\tilde{\mathbf{u}})$ from (4.13), we firstly define the following notation:

$$\mathcal{N}^{\mathrm{rot}}(\tilde{\mathbf{u}}) = \frac{1}{2}(\mathcal{G}^{\cdot} - [\mathcal{Q}^{\tilde{\mathbf{u}}} + \mathcal{Q}^{\mathbf{u}} + \mathcal{Q}^{\times}]),$$

with the corresponding discretization given as follows:

$$\mathcal{N}_h^{\mathrm{rot}}(\tilde{\mathbf{u}}_h) = \frac{1}{2}(\mathcal{G}_h^{\cdot} - [\mathcal{Q}_h^{\tilde{\mathbf{u}}} + \mathcal{Q}_h^{\mathbf{u}} + \mathcal{Q}_h^{\times}]), \tag{4.19}$$

where the potential term $\mathcal{G}^{\cdot}(\tilde{\mathbf{u}})$ is discretized as follows:

$$\mathcal{G}_h^{\cdot}(\tilde{\mathbf{u}}_h) = \begin{bmatrix} \mathrm{B}_{\mathrm{x}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^u & \mathrm{B}_{\mathrm{x}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^v \\ \mathrm{B}_{\mathrm{y}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^u & \mathrm{B}_{\mathrm{y}}^p \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^v \end{bmatrix}, \tag{4.20}$$

the term $\mathcal{Q}^{\tilde{\mathbf{u}}}(\tilde{\mathbf{u}})$ is discretized as follows:

$$\begin{aligned} &\mathcal{Q}_h^{\tilde{\mathbf{u}}}(\tilde{\mathbf{u}}_h) = \\ &= \begin{bmatrix} & -\mathrm{A}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{B}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v + \mathrm{A}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{B}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{x}}^v \\ \mathrm{A}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{B}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{y}}^u - \mathrm{A}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{B}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & \end{bmatrix}, \end{aligned} \tag{4.21}$$

the term $\mathcal{Q}^{\mathbf{u}}(\tilde{\mathbf{u}})$ is discretized as follows:

$$\mathcal{Q}_h^{\mathbf{u}}(\tilde{\mathbf{u}}_h) = \begin{bmatrix} -\mathrm{A}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{B}_{\mathrm{y}}^u & \mathrm{A}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{B}_{\mathrm{x}}^v \\ \mathrm{A}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{B}_{\mathrm{y}}^u & -\mathrm{A}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{B}_{\mathrm{x}}^v \end{bmatrix}, \tag{4.22}$$

and the term $\mathcal{Q}^{\times}(\tilde{\mathbf{u}})$ is discretized as follows:

$$\mathcal{Q}_h^{\times}(\tilde{\mathbf{u}}_h) = \begin{bmatrix} -\mathrm{B}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & \mathrm{B}_{\mathrm{y}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v \\ \mathrm{B}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & -\mathrm{B}_{\mathrm{x}}^q \mathrm{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v \end{bmatrix}. \tag{4.23}$$

Similarly, the reaction term $\mathcal{R}_h^{\mathrm{rot}}(\tilde{\mathbf{u}}_h)$ is assembled.

Finally, we denote $\mathbf{N}(\tilde{\mathbf{u}}_h)$ the matrix corresponding to any of the formulations of the convection term:

$$\mathbf{N}(\tilde{\mathbf{u}}_h) = \begin{cases} \mathcal{N}_h^{\text{grad}} & (\tilde{\mathbf{u}}_h), \\ \mathcal{N}_h^{\text{div}} & (\tilde{\mathbf{u}}_h), \\ \mathcal{N}_h^{\text{rot}} & (\tilde{\mathbf{u}}_h). \end{cases} \tag{4.24}$$

It is worth noting, that the underlying numerical schemes produce the same solution for any discretization of the convection term.

4.6 On the derivation of the Pressure Convection-Diffusion preconditioner and its generalization for tight geometries

In order to genralize the results obtained for the Stokes equations to the Oseen equations (4.2), we try to decompose the convection term using orthogonal operators $\mathbf{B}$ and $\mathbf{C}$ similarly to the decomposition (2.62) of the Neumann Laplacian operator $\mathbf{A_N}$. For this, we naturally use the rotational form $\mathcal{N}^{\text{rot}}$ (4.13). First of all, we define the discrete dot product operator $\mathbf{D}(\tilde{\mathbf{u}}_h)$ which is a discretization of the dot product operator $(\tilde{\mathbf{u}} \cdot \mathbf{u}) = \tilde{u}u + \tilde{v}v$:

$$\mathbf{D} = \begin{bmatrix} \text{diag}(\mathrm{A}_{\mathrm{x}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^u & \text{diag}(\mathrm{A}_{\mathrm{y}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^v \end{bmatrix}, \quad \mathbf{D} : (\mathcal{U}_h, \mathcal{V}_h)^T \to \mathcal{P}_h. \tag{4.25}$$

Similarly, we define the discrete cross product operator $\mathbf{X}(\tilde{\mathbf{u}}_h)$ which is a discretization of the cross product operator $(\tilde{\mathbf{u}} \times \mathbf{u}) = -\tilde{v}u + \tilde{u}v$:

$$\mathbf{X} = \begin{bmatrix} \text{diag}(-\mathrm{A}_{\mathrm{x}}^v \tilde{v}_h)\mathrm{A}_{\mathrm{y}}^u & \text{diag}(\mathrm{A}_{\mathrm{y}}^u \tilde{u}_h)\mathrm{A}_{\mathrm{x}}^v \end{bmatrix}, \quad \mathbf{X} : (\mathcal{U}_h, \mathcal{V}_h)^T \to \mathcal{Q}_h. \tag{4.26}$$

Then, the matrices $\mathcal{G}_h(\tilde{\mathbf{u}}_h)$, $\mathcal{Q}_h^{\mathbf{u}}(\tilde{\mathbf{u}}_h)$, $\mathcal{Q}_h^{\times}(\tilde{\mathbf{u}}_h)$ from (4.19) can be written as follows:

$$\mathcal{G}_h = \mathbf{B}^T\mathbf{D}, \quad \mathcal{Q}_h^{\mathbf{u}} = \mathbf{X}^T\mathbf{C}, \quad \mathcal{Q}_h^{\times} = \mathbf{C}^T\mathbf{X}. \tag{4.27}$$

Note, the reaction term $\mathcal{Q}_h^{\tilde{\mathbf{u}}}(\tilde{\mathbf{u}}_h)$ can not be represented in this way. So, the discretization of the convection term $\mathcal{N}_h^{\text{rot}}$ from (4.19) becomes:

$$\mathcal{N}_h^{\text{rot}} = \frac{1}{2}(\mathbf{B}^T\mathbf{D} - [\mathbf{C}^T\mathbf{X} + \mathbf{X}^T\mathbf{C} + \mathcal{Q}_h^{\tilde{\mathbf{u}}}]), \tag{4.28}$$

and the discretization of the convection-reaction term from (4.15) becomes:

$$\mathcal{N}_h^{\text{rot}} + \mathcal{R}_h^{\text{rot}} = \mathbf{B}^T\mathbf{D} - [\mathbf{X}^T\mathbf{C} + \mathcal{Q}_h^{\tilde{\mathbf{u}}}]. \tag{4.29}$$

If we assume $\mathbf{C}\tilde{\mathbf{u}}_h = 0$, then the reaction term $\mathcal{Q}_h^{\tilde{\mathbf{u}}}$ disappears, and the discretization of the auxiliary (enclosed Neumann) Oseen equations becomes:

$$\mathbf{A_N} + \mathcal{N}_h^{\text{rot}} = [\mathbf{B}^T\mathbf{B} + \frac{1}{2}\mathbf{B}^T\mathbf{D}] + [\mathbf{C}^T\mathbf{C} - \frac{1}{2}(\mathbf{X}^T\mathbf{C} + \mathbf{X}^T\mathbf{C})]. \tag{4.30}$$

Note, the left term disappears when (4.30) is multiplied by $\mathcal{P}_{\text{Ker}\mathbf{B}}$ from the left, and the right term disappears when (4.30) is multiplied by $\mathcal{P}_{\text{Ker}\mathbf{C}}$ from the right. If we additionally assume:

$$\mathbf{B}^T\mathbf{A} = \mathbf{A}^T\mathbf{B}, \quad \mathbf{C}^T\mathbf{X} = \mathbf{X}^T\mathbf{C}, \tag{4.31}$$

then, we have:

$$(\mathbf{A_N} + \mathcal{N}_h^{\text{rot}})^{-1} = [\mathbf{B}^T\mathbf{B} + \frac{1}{2}\mathbf{B}^T\mathbf{D}]^\dagger + [\mathbf{C}^T\mathbf{C} - \frac{1}{2}(\mathbf{X}^T\mathbf{C} + \mathbf{X}^T\mathbf{C})]^\dagger. \tag{4.32}$$

Based on this observation, the convection-diffusion preconditioner [96, 97, 90, 98, 64] can be derived, which is given as follows:

$$\hat{S}_N^\dagger = (\mathbf{B}\mathbf{B}^T)^\dagger(\mathbf{B}\mathbf{B}^T + \frac{1}{2}\mathbf{D}\mathbf{B}^T), \tag{4.33}$$

where $\mathbf{B}\mathbf{B}^T + \frac{1}{2}\mathbf{D}\mathbf{B}^T$ is discretization of the scalar pressure convection-diffusion operator with Neumann b.c, given as follows:

$$-\Delta p + \tilde{u}p_x + \tilde{v}p_y = -\Delta p + \frac{1}{2}\nabla \cdot (\tilde{\mathbf{u}}p).$$

Similarly to the preconditioner (2.80) derived for the Stokes equations, the analogue of (4.33) in the limiting case $\mathbf{I}_{\sim}^{\mathbf{u}} = \mathbf{I}^{\mathbf{u}}$ becomes:

$$\hat{S}_{lim}^\dagger = (\mathbf{B}\mathbf{B}^T)^\dagger(\frac{2}{h^2}I^p + \mathbf{B}\mathbf{B}^T + \frac{1}{2}\mathbf{D}\mathbf{B}^T), \tag{4.34}$$

where in the right brackets there is discretization of the pressure convection-diffusion-reaction equation.
In [64] it is said that the preconditioner (4.33) is an analog of the SIMPLE preconditioner due to the inverse Laplacian term $(\mathbf{B}\mathbf{B}^T)^\dagger$, but actually this is an analogue of the Uzawa preconditioner. Similarly how the preconditioner (2.80) is related to the SIMPLE for Stokes equations, the analog of the SIMPLE preconditioner for the PCD preconditioner will be as follows:

$$\hat{S}_D^\dagger = (\mathbf{B}\mathbf{B}^T)^\dagger(\frac{2}{h^2}I^p + \frac{1}{2}\mathbf{D}\mathbf{B}^T). \tag{4.35}$$

Chapter 5

Summary

An efficient pore-scale Stokes solver for low porosity images has been developed and analyzed. A thorough comparative study of the CG-SIMPLE and CG-Uzawa algorithms has been conducted. The results demonstrate that the CG-SIMPLE algorithm, when applied to 3D rock samples from tight reservoirs, ensures robust and fast convergence to high accuracy in solving the Schur complement problem and in computing permeability. In contrast, the established CG-Uzawa algorithm tends to stagnate. This behavior is further explained through a systematic study of synthetic 2D geometries. The primary conclusion drawn is that the condition number of the Schur complement matrix increases linearly with increasing the surface-to-volume ratio. Conversely, the condition number of the Schur complement matrix, preconditioned with the SIMPLE preconditioner, decreases super-linearly as the surface-to-volume ratio increases. It has also been demonstrated that the number of non-unit eigenvalues of the Schur complement matrix is determined by the number of the boundary nodes where the no-slip boundary conditions are imposed on the velocity, as well as by the connectivity of the flow domain. These findings provide essential insights into the behavior of solvers for the Schur complement matrix and demonstrate the effectiveness of the SIMPLE preconditioner in solving the Stokes problem in tight geometries.
A workflow has been developed for multiclass images derived from tight sandstones. This workflow comprises an image classification stage and customized, efficient Stokes-Brinkman and Darcy solvers. Rigorous testing has been conducted to validate the workflow's effectiveness. The first stage of the workflow involves classifying images based on the presence of Stokes percolation patches. This classification enables us to select an appropriate solver tailored to each specific image. The developed solvers have been rigorously validated using data computed with commercial software tools. Extensive testing has been carried out on samples from a real tight reservoir. The developed Stokes-Brinkman solver demonstrates robust convergence and high efficiency, enabling simulations not only for ternary images but also for multiclass images. The latter, derived directly from grayscale images, contain individual permeability values for each porous voxel. It has also been demonstrated that in cases where pure fluid percolation is missing, the Darcy approximation of the Stokes-Brinkman problem can be employed. This results in a significant acceleration of the computations while maintaining accuracy.

Bibliography

[1] W. Wang, D. Fan, G. Sheng, Z. Chen, and Y. Su, "A review of analytical and semi-analytical fluid flow models for ultra-tight hydrocarbon reservoirs," Fuel, vol. 256, p. 115737, 2019.

[2] S. Li, Q. Sang, M. Dong, and P. Luo, "Determination of inorganic and organic permeabilities of shale," International Journal of Coal Geology, vol. 215, p. 103296, 2019.

[3] X. Lu, R. T. Armstrong, and P. Mostaghimi, "Analysis of gas diffusivity in coal using micro-computed tomography," Fuel, vol. 261, p. 116384, 2020.

[4] R. Sander, Z. Pan, and L. D. Connell, "Laboratory measurement of low permeability unconventional gas reservoir rocks: A review of experimental methods," Journal of Natural Gas Science and Engineering, vol. 37, pp. 248–279, 2017.

[5] D. Chai, Z. Fan, X. Li et al., "A new unified gas-transport model for gas flow in nanoscale porous media," SPE Journal, vol. 24, no. 02, pp. 698–719, 2019.

[6] F. Javadpour, D. Fisher, M. Unsworth et al., "Nanoscale gas flow in shale gas sediments," Journal of Canadian Petroleum Technology, vol. 46, no. 10, 2007.

[7] S. Kelly, H. El-Sobky, C. Torres-Verdín, and M. T. Balhoff, "Assessing the utility of fib-sem images for shale digital rock physics," Advances in water resources, vol. 95, pp. 302–316, 2016.

[8] P. Karoly and L. S. Ruehlman, "Psychological "resilience" and its correlates in chronic pain: findings from a national community sample," Pain, vol. 123, no. 1-2, pp. 90–97, 2006.

[9] C. Jia, M. Zheng, and Y. Zhang, "Some key issues on the unconventional petroleum systems," Petroleum Research, vol. 1, no. 2, pp. 113–122, 2016.

[10] I. Verri, A. Della Torre, G. Montenegro, A. Onorati, S. Duca, C. Mora, F. Radaelli, and G. Trombin, "Development of a digital rock physics workflow for the analysis of sandstones and tight rocks," Journal of Petroleum Science and Engineering, vol. 156, pp. 790–800, 2017.

[11] Thermofisher, "Thermofisher web page," https://www.thermofisher.com, accessed: 21-11-04. [Online]. Available: https://www.thermofisher.com

[12] Math2Market, "Geodict web page," https://www.geodict.com/Solutions/aboutGD.php, accessed: 21-09-26. [Online]. Available: https://www.geodict.com/Solutions/aboutGD.php

[13] OpenFoam, "Openfoam web page," https://www.openfoam.com, accessed: 21-11-04. [Online]. Available: https://www.openfoam.com

[14] P. Mostaghimi, M. Blunt, and B. Bijeljic, "Computations of absolute permeability on micro-ct images," Mathematical Geosciences, vol. 45, 01 2012.

[15] M. Krotkiewski, I. S. Ligaarden, K.-A. Lie, and D. W. Schmid, "On the importance of the stokes-brinkman equations for computing effective permeability in karst reservoirs," Communications in Computational Physics, vol. 10, no. 5, pp. 1315–1332, 2011.

[16] D. Koroteev, O. Dinariev, N. Evseev, D. Klemin, A. Nadeev, S. Safonov, O. Gurpinar, S. Berg, C. Van Kruijsdijk, R. Armstrong et al., "Direct hydrodynamic simulation of multiphase flow in porous rock," Petrophysics-The SPWLA Journal of Formation Evaluation and Reservoir Description, vol. 55, no. 04, pp. 294–303, 2014.

[17] V. Balashov, E. Savenkov, and B. Chetverushkin, "Dimp-hydro solver for direct numerical simulation of fluid microflows within pore space of core samples," Mathematical Models and Computer Simulations, vol. 12, no. 2, pp. 110–124, 2020.

[18] H. Andrä, N. Combaret, J. Dvorkin, E. Glatt, J. Han, M. Kabel, Y. Keehm, F. Krzikalla, M. Lee, C. Madonna et al., "Digital rock physics benchmarks—part ii: Computing effective properties," Computers & Geosciences, vol. 50, pp. 33–43, 2013.

[19] N. Saxena, R. Hofmann, F. O. Alpak, S. Berg, J. Dietderich, U. Agarwal, K. Tandon, S. Hunter, J. Freeman, and O. B. Wilson, "References and benchmarks for pore-scale flow simulated using micro-ct images of porous media and digital rocks," Advances in Water Resources, vol. 109, pp. 211–235, 2017.

[20] S. Turek, Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approaches. Springer Science & Business Media, 1999, vol. 6.

[21] D. Orlov, M. Ebadi, E. Muravleva, D. Volkhonskiy, A. Erofeev, E. Savenkov, V. Balashov, B. Belozerov, V. Krutko, I. Yakimchuk et al., "Different methods of permeability calculation in digital twins of tight sandstones," Journal of Natural Gas Science and Engineering, vol. 87, p. 103750, 2021.

[22] R. Verfurth, "A combined conjugate gradient-multi-grid algorithm for the numerical solution of the stokes problem," IMA Journal of Numerical Analysis, vol. 4, no. 4, pp. 441–455, 1984.

[23] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," Acta numerica, vol. 14, pp. 1–137, 2005.

[24] H. C. Elman and G. H. Golub, "Inexact and preconditioned uzawa algorithms for saddle point problems," SIAM Journal on Numerical Analysis, vol. 31, no. 6, pp. 1645–1661, 1994.

[25] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, "Analysis of the inexact uzawa algorithm for saddle point problems," SIAM Journal on Numerical Analysis, vol. 34, no. 3, pp. 1072–1092, 1997.

[26] M. Fortin and R. Glowinski, Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems. Elsevier, 2000.

[27] O. Axelsson and J. Karátson, "Krylov improvements of the uzawa method for stokes type operator matrices," Numerische Mathematik, vol. 148, no. 3, pp. 611–631, 2021.

[28] Y. Notay, "Convergence of some iterative methods for symmetric saddle point linear systems," SIAM Journal on Matrix Analysis and Applications, vol. 40, no. 1, pp. 122–146, 2019.

[29] ——, "A new analysis of block preconditioners for saddle point problems," SIAM journal on Matrix Analysis and Applications, vol. 35, no. 1, pp. 143–173, 2014.

[30] S. V. Patankar and D. B. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," in Numerical prediction of flow, heat transfer, turbulence and combustion. Elsevier, 1983, pp. 54–73.

[31] S. Patankar, Numerical heat transfer and fluid flow. Taylor & Francis, 2018.

[32] Z.-Z. Liang and G.-F. Zhang, "Simple-like preconditioners for saddle point problems from the steady navier–stokes equations," Journal of Computational and Applied Mathematics, vol. 302, pp. 211–223, 2016.

[33] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, "A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible navier–stokes equations," Journal of Computational Physics, vol. 227, no. 3, pp. 1790–1808, 2008.

[34] J. B. Perot, "An analysis of the fractional step method," Journal of Computational Physics, vol. 108, no. 1, pp. 51–58, 1993.

[35] M. Pernice and M. D. Tocci, "A multigrid-preconditioned newton–krylov method for the incompressible navier–stokes equations," SIAM Journal on Scientific Computing, vol. 23, no. 2, pp. 398–418, 2001.

[36] A. Meier, E. Bänsch, and F. Frank, "Schur preconditioning of the stokes equations in channel-dominated domains," Computer Methods in Applied Mechanics and Engineering, vol. 398, p. 115264, 2022.

[37] V. Pimanov, V. Lukoshkin, P. Toktaliev, O. Iliev, E. Muravleva, D. Orlov, V. Krutko, A. Avdonin, K. Steiner, and D. Koroteev, "On a workflow for efficient computation of the permeability of tight sandstones," arXiv preprint arXiv:2203.11782, 2022.

[38] M. Griebel, T. Dornseifer, and T. Neunhoeffer, Numerical simulation in fluid dynamics: a practical introduction. SIAM, 1998.

[39] F. H. Harlow and J. E. Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface," The physics of fluids, vol. 8, no. 12, pp. 2182–2189, 1965.

[40] V. I. Lebedev, "Difference analogues of orthogonal decompositions, basic differential operators and some boundary problems of mathematical physics. i," USSR Computational Mathematics and Mathematical Physics, vol. 4, no. 3, pp. 69–92, 1964.

[41] H. Ulrich, Homogenization and Porous Mediae. Springer-Verlag New York, 1997.

[42] M. Griebel and M. Klitz, "Homogenization and numerical simulation of flow in geometries with textile microstructures," Multiscale Modeling & Simulation, vol. 8, no. 4, pp. 1439–1460, 2010.

[43] S. Whitaker, "Flow in porous media i: A theoretical derivation of darcy's law," Transport in porous media, vol. 1, no. 1, pp. 3–25, 1986.

[44] A. Wiegmann, "Computation of the permeability of porous materials from their microstructure by fff-stokes," Berichte des Fraunhofer ITWM, 2007 (129), 2007.

[45] J. W. Ruge and K. Stüben, "Algebraic multigrid," in Multigrid methods. SIAM, 1987, pp. 73–130.

[46] Y. Saad, Iterative methods for sparse linear systems. SIAM, 2003.

[47] H. Rui and X. Li, "Stability and superconvergence of mac scheme for stokes equations on nonuniform grids," SIAM Journal on Numerical Analysis, vol. 55, no. 3, pp. 1135–1158, 2017.

[48] E. F. Kaasschieter, "Preconditioned conjugate gradients for solving singular systems," Journal of Computational and Applied mathematics, vol. 24, no. 1-2, pp. 265–275, 1988.

[49] G. H. Golub and C. F. Van Loan, Matrix computations. JHU press, 2013.

[50] H. A. Van der Vorst, Iterative Krylov methods for large linear systems. Cambridge University Press, 2003, no. 13.

[51] U. M. Yang et al., "Boomeramg: a parallel algebraic multigrid solver and preconditioner," Applied Numerical Mathematics, vol. 41, no. 1, pp. 155–177, 2002.

[52] M. A. Afaq, A. Fatima, S. Turek, and A. Ouazzi, Robust Monolithic Multigrid FEM Solver for Three Field Formulation of Incompressible Flow Problems. Institute for Applied Mathematics and Numerics (LS3), TU Dortmund University, 2023.

[53] A. Voronin, Y. He, S. MacLachlan, L. N. Olson, and R. Tuminaro, "Low-order preconditioning of the stokes equations," Numerical Linear Algebra with Applications, vol. 29, no. 3, p. e2426, 2022.

[54] Y. Notay, "A new algebraic multigrid approach for stokes problems," Numerische Mathematik, vol. 132, pp. 51–84, 2016.

[55] ——, "Algebraic multigrid for stokes equations," SIAM Journal on Scientific Computing, vol. 39, no. 5, pp. S88–S111, 2017.

[56] P.-L. Bacq, S. Gounand, A. Napov, and Y. Notay, "An all-at-once algebraic multigrid method for finite element discretizations of stokes problem," International Journal for Numerical Methods in Fluids, vol. 95, no. 2, pp. 193–214, 2023.

[57] A. Voronin, S. MacLachlan, L. N. Olson, and R. Tuminaro, "Monolithic algebraic multigrid preconditioners for the stokes equations," arXiv preprint arXiv:2306.06795, 2023.

[58] P. E. Farrell, Y. He, and S. P. MacLachlan, "A local fourier analysis of additive vanka relaxation for the stokes equations," Numerical Linear Algebra with Applications, vol. 28, no. 3, p. e2306, 2021.

[59] D. Silvester and A. Wathen, "Fast iterative solution of stabilised stokes systems part ii: Using general block preconditioners," SIAM Journal on Numerical Analysis, vol. 31, no. 5, pp. 1352–1367, 1994.

[60] ——, "Fast iterative solution of stabilised stokes systems part ii: Using general block preconditioners," SIAM Journal on Numerical Analysis, vol. 31, no. 5, pp. 1352–1367, 1994.

[61] S. Linden, A. Wiegmann, and H. Hagen, "The lir space partitioning system applied to the stokes equations," Graphical Models, vol. 82, pp. 58–66, 2015.

[62] Y. Jung and S. Torquato, "Fluid permeabilities of triply periodic minimal surfaces," Physical Review E, vol. 72, no. 5, p. 056319, 2005.

[63] A. S. Sangani and A. Acrivos, "Slow flow through a periodic array of spheres," International Journal of Multiphase Flow, vol. 8, no. 4, pp. 343–360, 1982.

[64] H. C. Elman, D. J. Silvester, and A. J. Wathen, Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. Numerical Mathematics and Scie, 2014.

[65] J. M. Hyman and M. Shashkov, "Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids," Applied Numerical Mathematics, vol. 25, no. 4, pp. 413–442, 1997.

[66] A. Abba and L. Bonaventura, "A mimetic finite difference discretization for the incompressible navier–stokes equations," International journal for numerical methods in fluids, vol. 56, no. 8, pp. 1101–1106, 2008.

[67] J. Brzenski and J. E. Castillo, "Solving navier–stokes with mimetic operators," Computers & Fluids, vol. 254, p. 105817, 2023.

[68] M. Shashkov, "Support operator methods and numerical modeling hydrodynamic flows in the case of strong deformations (in russian)," Ph.D. dissertation, Moscow State University, Faculty of Computational Mathematics and Cybernetics, 1989.

[69] W. Hackbusch and B. N. Khoromskij, "Low-rank kronecker-product approximation to multi-dimensional nonlocal operators. part i. separable approximation of multi-variate functions," Computing, vol. 76, pp. 177–202, 2006.

[70] E. A. Muravleva, "On the kernel of the discrete gradient operator," Numerical Methods and Programming (Vychislitel'nye Metody i Programmirovanie), vol. 9, pp. 93–100, 2008.

[71] I. Oseledets and E. Muravleva, "Fast orthogonalization to the kernel of the discrete gradient operator with application to stokes problem," Linear algebra and its applications, vol. 432, no. 6, pp. 1492–1500, 2010.

[72] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," The Annals of Mathematical Statistics, vol. 21, no. 1, pp. 124–127, 1950.

[73] O. Axelsson and M. Neytcheva, "Preconditioning methods for linear systems arising in constrained optimization problems," Numerical linear algebra with applications, vol. 10, no. 1-2, pp. 3–31, 2003.

[74] ——, "Eigenvalue estimates for preconditioned saddle point matrices," Numerical Linear Algebra with Applications, vol. 13, no. 4, pp. 339–360, 2006.

[75] G. H. Golub and M. L. Overton, "The convergence of inexact chebyshev and richardson iterative methods for solving linear systems," Numerische Mathematik, vol. 53, no. 5, pp. 571–593, 1988.

[76] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Web page," https://www.mcs.anl.gov/petsc, 2021. [Online]. Available: https://www.mcs.anl.gov/petsc

[77] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang, "PETSc/TAO users manual," Argonne National Laboratory, Tech. Rep. ANL-21/39 - Revision 3.16, 2021.

[78] "Porechem – simulation of reactive material transport in porous media," https://www.itwm.fraunhofer.de/en/departments/sms/products-services/porechem.html.

[79] "Samg – solving large systems of linear equations efficiently," https://www.scai.fraunhofer.de/de/geschaeftsfelder/schnelle-loeser/produkte/samg.html.

[80] T. Arbogast and H. L. Lehr, "Homogenization of a darcy–stokes system modeling vuggy porous media," Computational Geosciences, vol. 10, no. 3, pp. 291–302, 2006.

[81] I. Y. Willian J. Layton, Schieweck Friedhelm, "Coupling fluid flow with porous media flow," SIAM Journal on Numerical Analysis, vol. 40, no. 6, p. 2195–2218, 2003.

[82] A. Q. Marco Discacciati, "Navier-stokes/darcy coupling: modeling, analysis, and numerical approximation." Revista Matemática Complutense, vol. 22, no. 2, pp. 315–426, 2009. [Online]. Available: http://eudml.org/doc/43558

[83] F. Golfier, D. Lasseux, and M. Quintard, "Investigation of the effective permeability of vuggy or fractured porous media from a darcy-brinkman approach," Computational Geosciences, vol. 19, no. 1, pp. 63–78, 2015.

[84] H. C. Brinkman, "A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles," Flow, Turbulence and Combustion, vol. 1, no. 1, pp. 27–34, 1949.

[85] V. Laptev, "Numerical solution of coupled flow in plain and porous media," Ph.D. dissertation, Technische Universität Kaiserslautern, 2003.

[86] A. Avdonin, M. Ebadi, and V. Krutko, "Application of high-contrast μct and fib-sem for the improvement in the permeability prediction of tight rock samples," in SPE Russian Petroleum Technology Conference. OnePetro, 2021.

[87] B. A. Galler and M. J. Fisher, "An improved equivalence algorithm," Communications of the ACM, vol. 7, no. 5, pp. 301–303, 1964.

[88] M. Ebadi, D. Orlov, V. Alekseev, A. Burukhin, V. Krutko, and D. Koroteev, "Lift the veil of secrecy in sub-resolved pores by xe-enhanced computed tomography," Fuel, vol. 328, p. 125274, 2022.

[89] T. C. Fisher, M. H. Carpenter, J. Nordström, N. K. Yamaleev, and C. Swanson, "Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions," Journal of Computational Physics, vol. 234, pp. 353–375, 2013.

[90] H. C. Elman, "Preconditioning for the steady-state navier–stokes equations with low viscosity," SIAM Journal on Scientific Computing, vol. 20, no. 4, pp. 1299–1316, 1999.

[91] P. Kjellgren, "A semi-implicit fractional step finite element method for viscous incompressible flows," Computational Mechanics, vol. 20, no. 6, pp. 541–550, 1997.

[92] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," SIAM Journal on scientific and statistical computing, vol. 7, no. 3, pp. 856–869, 1986.

[93] M. A. Olshanskii and A. Reusken, "Navier–stokes equations in rotation form: A robust multigrid solver for the velocity problem," SIAM Journal on Scientific Computing, vol. 23, no. 5, pp. 1683–1706, 2002.

[94] W. Layton, C. C. Manica, M. Neda, M. Olshanskii, and L. G. Rebholz, "On the accuracy of the rotation form in simulations of the navier–stokes equations," Journal of Computational Physics, vol. 228, no. 9, pp. 3433–3447, 2009.

[95] G. Lube and M. A. Olshanskii, "Stable finite-element calculation of incompressible flows using the rotation form of convection," IMA Journal of Numerical Analysis, vol. 22, no. 3, pp. 437–461, 2002.

[96] D. Kay, D. Loghin, and A. Wathen, "A preconditioner for the steady-state navier–stokes equations," SIAM Journal on Scientific Computing, vol. 24, no. 1, pp. 237–256, 2002.

[97] D. Silvester, H. Elman, D. Kay, and A. Wathen, "Efficient preconditioning of the linearized navier–stokes equations for incompressible flow," Journal of Computational and Applied Mathematics, vol. 128, no. 1-2, pp. 261–279, 2001.

[98] M. A. Olshanskii and Y. V. Vassilevski, "Pressure schur complement preconditioners for the discrete oseen problem," SIAM Journal on Scientific Computing, vol. 29, no. 6, pp. 2686–2704, 2007.

Academic Curriculum Vitae

2008 – 2013	Middle school Abakan Gymnasium
2013 – 2015	High school Specialized Educational Scientific Center at NSU
2013 – 2015	Bachelor Student, Information Technology Novosibirsk State University (NSU)
2015 – 2017	Bachelor Student, Mathematics and System Programming Lomonosov Moscow State University (MSU)
2017 – 2019	Master Student, Computational Science and Engineering Skolkovo Institute of Science and Technology (Skoltech)
2019 – 2022	Doctoral Student, Computational and Data Science and Engineering Skolkovo Institute of Science and Technology (Skoltech)
2022 – 2023	Doctoral Student, Mathematics Fraunhofer Institute for Industrial Mathematics (ITWM), The Rhineland-Palatinate Technical University Kaiserslautern-Landau (RPTU)

Akademischer Lebenslauf

2008 – 2013	Mittelschule Abakan Gymnasium
2013 – 2015	Gymnasium Spezialisiertes Bildungswissenschaftliches Zentrum an der NSU
2013 – 2015	Bachelor-Student, Informatik Staatliche Universität Nowosibirsk (NSU)
2015 – 2017	Bachelor-Student, Mathematik und Systemprogrammierung Lomonossow-Universität Moskau (MSU)
2017 – 2019	Master-Student, Computational Science and Engineering Skolkovo Institute of Science and Technology (Skoltech)
2019 – 2022	Doktorand, Computational and Data Science and Engineering Skolkovo Institute of Science and Technology (Skoltech)
2022 – 2023	Doktorand, Mathematik Fraunhofer-Institut für Industriemathematik (ITWM), Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau (RPTU)