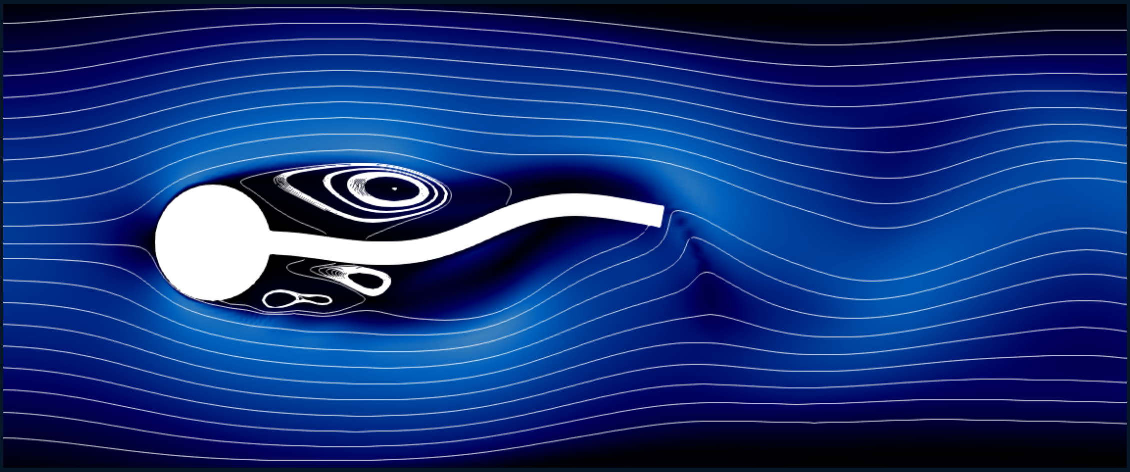
Department of Mechanics of Materials
Institute of Fundamental Technological Research
Polish Academy of Sciences



PhD Thesis

Fluid-structure interaction problems: velocity-based formulation and monolithic computational methods

Author:
Michał Wichrowski

Supervisor:
Prof. Stanisław Stupkiewicz
Co-supervisor:
Dr. Piotr Krzyżanowski

Warsaw, 2021

The author would like to thank Prof. Stanisław Stupkiewicz and Dr. Piotr Krzyżanowski for their patience and valuable comments.

The numerical experiments presented in the thesis were performed using the Marconi supercomputer, courtesy of Prof. Luca Heltai (SISSA, Trieste).

This work was supported by the National Science Center (NCN) in Poland under grant No. 2015/19/N/ST8/03924.

Fluid-structure interaction problems: velocity-based formulation and monolithic computational methods

Zagadnienia oddziaływania płyn-ciało stałe: sformułowanie prędkościowe i monolityczne metody obliczeniowe

Michał Wichrowski

Supervisor: Prof. Stanisław Stupkiewicz, IPPT PAN
Co-supervisor: Dr. Piotr Krzyżanowski, MIM UW

Cover: Visualization created in Paraviev showing velocity magnitude and streamlines in Turek benchmark with Re=1000.

IPPT PAN
Warsaw

Contents

Introduction 1

At the time of writing this introduction, i.e. late 2020, it is hard not to start with a specific application of the fluid-structure interaction (FSI) problems: modeling of the human respiratory system. Since the beginning of the year the topic became discussed not only by scientists but it also commonly appears in the public discussion. The most thought-provoking are oxygen and carbon dioxide exchange rates in presence of an obstacle in front of the face [1]. The efficiency of filtration of inhaled and exhaled gases through permeable membranes is similarly popular [2]. Growing public interest in the topic even led to its emergence in the political discussion, including the first US presidential debate. The awareness of micro-flow characteristics and well-recognized advantages of the N-95 filtering devices may be linked to its shortages in the first half of the year.

Let us fix the definition, by fluid-structure interaction problem we mean the mathematical model of physical phenomena involving both structure and fluid interacting with each other. The coupling between structure and fluid usually occurs at the interface. The structure, also referred to as the solid, could be a point mass, rigid body, elastic body or any other kind of solid model. The fluid is usually considered as continuous medium but other models are possible. The coupling between solid and fluid usually occurs at the interface. The FSI problems fall into the class of the so-called multi-physics problems, where two or more physical models are combined. The other examples of such problems are magneto-hydrodynamics [3, 4] where electrically conducting fluids are studied or magneto-thermodynamic where temperature change is caused by exposing the material to a changing magnetic field. A common misconception in the last one resulted in several arsons of 5G transmitters in UK.

Back to the FSI applications, the N-95 masks operate on three principles [2, 5]. The particles typically 1 micron or larger slam into the mask fibers. The particles smaller than 0.1 micron undergo Brownian motion and get stuck into the material. Finally, the particles between 0.1 and 1 micron are filtered out by electrostatic attraction to charged polymer fibers. The first two ones could be studied using a proper FSI model, while the third additionally requires taking the electrostatic forces into account.

The other well-recognized incident this year that could be modeled as a FSI problem was the explosion of ammonium nitrate stored at the port of Beirut. The vast amount of gas produced by the explosive decomposition of the nitrate caused a shock wave and as the result, a significant part of the city was destroyed while the port alone was turned into a crater roughly 124 m in diameter and 43 m in depth. As a typical illustration of the damage of a structure due to aerodynamic forces is Tacoma Narrows Bridge. The recording from its collapse is a reminder of devastating potential of harmonic resonance. The bridge collapsed during winds occurring seasonally in that area (of just 68 km/h) which unexpectedly produced aeroelastic flutter that matched the bridge's natural frequency.

Among the spectacular examples of applications of FSI problems in civil engineering is design of tall building such as Burj Khalifa, that is currently the world highest building with the tip at 828 m above the ground. The wind-induced vibrations of the tall structures may have a destructive effect. In case of the Burj Khalifa this is resolved by corner modifications of the cross-section and tapering along the height [6]. This ensures different vortex shedding frequency at each part of the building and prevents resonance. A similar effect is used to suppress the vortex-induced vibrations by adding helical strakes at offshore structures, such as riser [7] that is the main connection between the offshore platform and the subsea wellheads. A similar phenomena is also exploited in the benchmark problem commonly used for testing FSI solvers [8].

Decarbonization of the energy generation system is considered the only way to stop advancing climate changes. Among the renewable energy sources, wind turbines currently are becoming more efficient and cheaper. Rotor blades of diameter exceeding 120 m are being designed and built for better performance. The arising engineering challenges must be addressed through research and development, which also involves large-scale simulations, in particular solving the FSI problem [9–11]. Similar problems arise in the aerodynamic optimization of aircraft wings.

The simulation of FSI problems are also applied in biology and medicine. Heart valve dynamics [12, 13] is one of the fundamental problems in understanding human cardiovascular system. According to WHO [14], the heart attack is currently the number one cause of death globally, taking an estimated 17.9 million lives each year. Developing better models and computational tools may help to save some of those lives.

A notable example of the FSI problem at the micro-scale is hydrodynamic lubrication [15, 16] that could be observed in mechanical devices such as bearings or piston engines. Seemingly simple lubrication problem usually results in attention-grabbing two-phase flow as the occurrence of cavitation is common for this type of problems. In this case, the fluid is modelled by the Reynolds lubrication equation usually with a modification for handling two-phase flows.

The FSI problems may be an interesting physical phenomenon by itself. A noteworthy example is the dynamics of groups of particles falling in the fluid may exhibit quasi-periodic motion. The phenomena were studied both experimentally and numerically at low Reynolds numbers [17, 18].

Finally, at the nano-scale the FSI problems could be applied to model filtration of particles, the problem that we addressed in the fist paragraph. Also motion of particles or filaments suspended in fluids is studied to provide a better understanding of cell movements, dynamics of proteins, DNA and other biological polymers [19]. The advances in this field may also allow improvements in tissue engineering and drug delivery methods.

1.1 State of the art

Currently, the research in the FSI problems covers a wide range of problems, we referred to some of them in the previous section. Three collective volumes [20–22] on the subject have been published, as well as at least two monographs: Bazilevs [23] and Richter [24]. Depending on the application, various models and computation method could be employed, here we only consider problems with both solid and fluid modeled as a continuous medium. In this setting, a FSI problem leads to a system of partial differential equations, that is usually solved by a appropriately chosen numerical method. We will focus on applying the Finite Element Method [25], but let us first note that other approaches could be more suitable in some cases. For instance, for a single spherical particle falling in a viscous fluid with very small Reynolds, the analytical solution is known, for groups of particles, the fundamental solutions of the Stokes problem could be exploited to obtain a numerical solution [17, 18, 26].

The Finite Element Method requires introducing triangulation of the domain. The triangulation, also called mesh, consists of polyhedrons (called elements), the solution in each element is approximated by a polynomial. In case of FSI problem, the solid boundaries may coincide with the elements boundaries. If not, the method is referred as non-matching. Both approaches have their advantages and disadvantages and the choice should be done taking into account the specific application.

1.1.1 Non-matching methods

A quite straightforward non-matching method could be obtained by formulating both solid and fluid models in the Eulerian description [24, 27]. This approach overcomes the problems associated with variable geometry but raises several issues. Tracking of the solid, especially the fluid-solid interface, becomes non-trivial. The accurate representation of the interface is hard to obtain. The last issue could be resolved by introducing adaptive refinement of the grid. Additionally, the solid stress typically depends on the deformation gradient that also becomes challenging to compute.

The solid and fluid problems could be separated and then recoupled by introducing Lagrange multipliers. This idea is the underpinning of the fictitious boundary methods also known as immersed finite element methods, introduced by Glowinski [28] followed by numerous works, for example [29–32]. The approach allows a convenient description of the solid in the Lagrangian formulation while the fluid could be described in the Eulerian frame of reference. The main downside of the immersed method is interface tracking. Although its position could be straightforwardly determined, the intersection with the fluid elements has to be computed. This could be a challenging problem by itself, especially in parallel computations [33].

Overcoming implementation issues could be highly rewarding as the method offers great versatility. Less restrictions on the topology of the fluid domain together with a convenient formulation of the solid in its natural frame of reference paved this approach a way to numerous application in FSI problems with solid contact occurring. Among them, we can mention haemodynamics of heart valves or helicopter dynamics [32, 34, 35].

As an interesting application-oriented example of the immersed method, we recall the stimulation of suspended particles [36]. In this case, particles are considered as rigid bodies to reduce complexity of the problem.

1.1.2 Arbitrary Lagrangian Eulerian formulation

The mesh matching the moving solid boundaries may be obtained by deforming the initial mesh. In this approach usually the solid motion is traced in Lagrangian frame of reference, while the fluid is described in an arbitrary coordinate system, hence the name *Arbitrary Lagrangian-Eulerian (ALE)* [37–41]. If possible, this approach is preferable, as it offers convenient Lagrangian formulation of the solid. On the other hand, the ALE methods require existence of a smooth enough and non-degenerate transition between any two configurations, that could not always be guaranteed. As a consequence, the ALE formulation could not be applied to problems where changes in topology are expected. Moreover, a severe deformation could result in deterioration of mesh. Using a meshless method [42] or mesh regeneration [43] may resolve some of those problems.

Despite those limitations, the ALE gained popularity. With numerous applications such as simulation of blood flow in elastic vessels [44–46] or gas flow in human respiratory system [47]. With some further adaptations rotating structures like wind turbines [10, 11, 48, 49] have been considered. Consequently, there is an intensive development of dedicated computational methods for this formulation [23].

Mesh deformation

It is not hard to guess that a proper choice of mesh deformation algorithm is a crucial element affecting the applicability of the ALE formulation. In some time integration schemes, the mesh deformation is a part of a (possibly nonlinear) system of equations solved at each time step, in others, like the one presented in this thesis, it is decoupled from the system. In both cases, simple mesh deformation methods are usually chosen to reduce the complexity of the discrete problem.

A mesh deformation could be computed by solving an auxiliary problem, that could be seen as pseudo-elasticity of the mesh. As an example, the biharmonic equation could applied as in [50]. This approach is capable of handling large deformations relatively well, but comes at the cost of increased computational expense. The biharmonic problem could be a bit tricky to solve when concerning parallel computations.

A popular physical analogy applied as mesh deformation is elasticity. The resulting equation is less computationally demanding, but it requires proper choice of elastic parameters to avoid problems with invalid elements. An interesting review of mesh deformation methods could be found in [51]. The impact of the mesh deformation algorithm on the results can be found in [52].

Time discretization

As in most time-dependent problems, when choosing a time discretization scheme one should balance between stability, and cost of a single time step. The stability offered by the implicit schemes usually comes at the high computational cost of a single time step. Although some implicit time schemes are unconditionally stable, accuracy also comes into play limiting the time step size. Explicit schemes are usually on the other side of the spectrum by offering relatively cheap time step, but fall behind the implicit ones in terms the stability. To find the sweet spot between those methods, semi-implicit schemes are constructed mixing the two approaches.

In the case of FSI problem involving incompressible fluid, fully explicit schemes are considered unstable due to problems with added-mass effect [53]. On the other hand, implicit schemes lead to a system of non-linear equations to be solved at each time step.

The most complex computational scheme yet most stable is the *fully implicit* scheme [54], where the whole problem is treated implicitly. As a result, a coupled system of equations arising from fluid motion, displacement of solid and domain geometry is obtained.

Solving such nonlinear problem is not necessarily needed, a stable integration scheme could be obtained by using explicit methods to first determine the deformed domain and then use an implicit scheme to compute the flow and deformation. This idea lies behind the *geometry-convective explicit (GCE) scheme* [55–57]. Additionally, the computational complexity could be reduced by explicitly treating some non-linear parts of the Navier-Stokes equations. As a result, when considering linear elasticity, the system of equations becomes linear. In this work we will use a modification this concept, adapted to handle incompressible hyperelastic solids. In [58] similar approach is studied and stability of finite element formulation of FSI problem is shown.

Solving a possibly nonlinear problem at each time step is yet another issue. A popular segregated approach could be used, that is to alternately solve flow problem and deformation equations until a convergence criterion is met. In this choice the existing solvers could be reused, greatly reducing the implementation work required. The convergence could be effectively sped up by employing a special technique as in [59].

However, the use of Newton's method to the entire system of equations (monolithic method) appears to be more efficient [60, 61]. A clear drawback of this approach is no possibility to reuse existing fluid or solid solvers, thus the development of highly specialized code is needed but the effort may pay off in the long run. The main bottleneck of this approach is to solve a large system of linear equations.

1.1.3 Linear solvers for the FSI problems

Regardless of the chosen time discretization method, the time integration algorithm for the FSI problem typically requires solving a system of partial differential equations at each time step. Its spatial discretization brings the problem to solving a system of algebraic equations. This comes down to solving a linear problem, possibly repeatedly inside the Newton method or other chosen fixed-point procedure.

The direct methods, suitable for smaller problems, become inefficient with growing problem size due to poor parallel scaling. Therefore, direct methods are not considered in the case of high-performance computations. Instead, dedicated iterative solvers based on Krylov space are used.

Due to the different properties of the fluid and solid, the system matrix is ill conditioned. The number of iterations required for reducing the initial residual by certain factor directly depends on condition number of the matrix [62]. The remedy for that is applying handpicked preconditioner. Therefore, the design of problem-specific preconditioner is crucial for efficiency. In the general case of jump-coefficient problems, *domain decomposition* [63] or *multigrid* [62] seem to be a promising choice.

For the FSI problems, existing preconditioners exploit block structure of matrix [64], based on inexact block LU factorization [65]. In [66] FSI is formulated as a saddle-point problem and augmented lagrangian [67] preconditioner is developed. There are also preconditioners based on domain decomposition [68], algebraic multigrid methods [69] or geometric multigrid [37]. In [70] a review of existing methods have been presented.

Relatively recently first attempts of theoretical analysis of FSI have appeared, proving the well-posedness of the linear problem solved in each time step and optimality of block preconditioners based on saddle point formulation for fluid-structure interaction [66]. Methods presented in [66] have been extended and applied to simulate rotating structure immersed in fluid in [71]. While the preconditioners in [66] were designed with inexact block solvers in mind, the experimental results were provided only for direct block solvers. On the other hand, algebraic multigrid preconditioned GMRES solvers were used in [71] to solve the blocks, but no results concerning the performance of the preconditioner were provided.

The choice of solver behind each block may be challenging. From our experience, it follows that popular choices like algebraic multigrid or incomplete LU fail. For this reason, in this work, we propose a multilevel preconditioner acting on the entire system and then exploit the block structure of the problem at each level.

Matrix-free solvers

An efficient implementation is undoubtedly a key part of any high-performance solver. A poorly organized data handling may kill the performance of any solver, while the efficient management of available resources could result in significant speedups. For some methods, it may be easier to optimize the data flow inside the program, while for others the architecture-dependent optimizations are hard to implement if possible at all.

An advantageous feature of modern processors are *Single Instruction, Multiple Data (SIMD)* instructions allowing operations on multiple floating point values, so-called vectors of data. To be used effectively, the data flow inside that program have to be well-organized so that other bottle-necks do not appear. In particular, minimizing stored data becomes crucial, as the retrieving it from the memory may cause delays.

In the finite element method one typically ends up with a linear system to be solved. The system matrix entities, associated with integrals over individual element, are typically pre-computed, and the matrix together with the right-hand side vector is passed to solving procedure. The matrix is usually sparse, allowing its storage in dedicated structures. The downside of this concept is a hard to regularize data flow.

To avoid that, one may not assemble the system matrix. Instead, the matrix-vector product are computed by calculating integrals over element each time. In this way, we minimize data stored but increase the number of calculations. However, the latter can be effectively sped up by vectorization, resulting in an overall performance boost [72].. Additionally, the time spent on matrix assembly is almost entirely saved, as the matrix-free operators are initialized significantly faster. For detailed insights we refer to `deal.II` library manual [73] (in particular *step-37*) and work by Kronbichler and Kormann [72].

This approach requires special linear solvers since the individual matrix entities could be no longer directly accessed. In this case, Krylov subspace iterative methods become a natural choice since they only require implementing a matrix-vector multiplication. A well-designed preconditioner is crucial for the performance, while it also has to be matrix-free compatible. One of possible choices is the multigrid method with suitable smoother. A simple Jacobi smoother requires a diagonal, that is just a single vector, which can be efficiently pre-computed, stored, and used in the computations. The smoother could be further enhanced by employing polynomial Chebyshev operator [74, 75], considered to be the parallel equivalent of Gauss-Seidel smoother.

Matrix-free solvers for FSI There exist few matrix-free FSI solvers. The most relevant case was studied in [76], where the partitioned solver was implemented. The FSI problem involving compressible fluid with finite volume discretization was considered in [77]. By introducing artificial viscosity in [78] a projection method has been utilized to develop a matrix-free solver for the FSI problem involving a rigid solid.

1.2 Scope of the thesis

A variety of choices of preconditioners for the FSI problems exist in the literature, however, a great majority of them have not been designed to support matrix-free computations. Among a few [76–78] works where a matrix-free implementation was considered none uses a monolithic formulation. The coupling at each time step is enforced by a fixed-point method, that requires multiple solutions of both fluid and solid equations alternately.

The problem formulation and resulting preconditioner proposed by [66] seem to have simple enough form to try adopting it to matrix-free computations. We claim, that it could be done, but still it is far from being trivial since a new preconditioner is required. Another drawback of this formulation is the use of linear elasticity, that is inadequate if considering a case with large deformations.

With all that said, let us settle the scope of this thesis.

> In this work, we consider numerical methods for solving interaction between an incompressible hyperelasetic solid and Newtonian incompressible fluid in a range of small to moderate Reynolds numbers. The problem is formulated in the ALE frame of reference. We restrict ourselves to monolithic time integration scheme combined with finite element methods. We also discuss the solution method of the system of linear equations arising from finite element discretization.

We propose a time integration schemes based on first- and second-order BDF [79], extending the *GCE* scheme with several modifications. We derive the velocity formulation similar to the one from [56], where velocity and pressure are the main unknowns while the solid displacement is recovered separately.

Concerning the solid model, there are a variety of results for FSI problems involving hyperelastic solids, but the ones for the incompressible solids are limited [58]. As we will later show, problem involving incompressible solid has its own issues that have to be addressed. On the other hand, its usage pays off by resulting in simplified equations. We expect that our method could be adopted to other solid models without greater problems. We consider incompressible Mooney-Rivlin solid, for the compressible Mooney-Rivlin we refer to [61].

Our time integration scheme consists of several substeps, among them the most computationally demanding one involves the solution of a generalized Stokes problem with discontinuous coefficients. We propose a new matrix-free preconditioner, that combined with Krylov subspace methods is robust with respect to problem size and coefficient jumps.

> The goal of this work is to develop a a parallel monolithic matrix-free solver for FSI problems. To achieve it, a time integration scheme is proposed, generalizing the GCE scheme together with a new linear solver.

We implement our solver within the framework of an established finite element library `deal.II` [73, 80], widely exploiting its support for parallel computations. In particular, our design choice to base the solver building blocks on the matrix-free paradigm, not only allows for low level speed and memory optimizations, but also promotes parallel and extends constructs available in `deal.II`. We note that a different monolithic FSI solver based on `deal.II` already exists [81], but it is based on a direct solver.

The `deal.II` library itself is a well-recognized open-source library widely used in many academic and commercial projects. For its creation, its principal authors have received the 2007 J. H. Wilkinson Prize for Numerical Software.

1.2.1 Structure of this thesis

In the next chapter we define the fluid-structure Interaction problem, starting from domain description, and then reformulate the governing equations from Eulerian frame of reference to arbitrary Lagrangian-Eulerian one. We also define there the fluid and solid models, additionally introducing minor modification that does not affect the solution of a continuous problem while is beneficial when considering time-discrete problems.

In Chapter 3 we introduce time and space discretization, and formulate the linear problem that is the main bottleneck of the time integration algorithm. We intend to solve the system with the Krylov subspace solver, thus the preconditioner is crucial for efficiency. In Chapter 4 we propose a new multilevel preconditioning method based on the one proposed by Braess-Sarazin [82] and refined by Zulehner [83]. We provide a separate introduction there, including a review of the state of the art concerning variable coefficient Stokes problem. In Chapter 5 we test each part of the implementation, heading towards solving the FSI problem in Chapter 6. Bringing things to an end, in Chapter 7 we conclude the thesis.

Formulation of the fluid-structure interaction problem 2

Let us consider domain $\hat{\Omega} \subset R^d$, the initial (reference) configuration, consisting two non-overlapping sub-domains: fluid domain $\hat{\Omega}_f \subset \hat{\Omega}$ and solid domain $\hat{\Omega}_s \subset \hat{\Omega}$, $\hat{\Omega} = \hat{\Omega}_s \cup \hat{\Omega}_f$. As time advances, one can expect changes in configuration, we denote the actual domain at time $t \in [0, T]$ by $\Omega(t)$, with the convention that $\Omega(0) = \hat{\Omega}$. The deformed solid occupies the domain $\Omega_s(t)$, while the fluid domain, $\Omega_f(t)$, is the remaining part of $\Omega(t)$.

Due to a deformation, each individual point $\hat{x}$ of the undeformed solid is moved to its current position $x(t)$ at time t (Figure 2.1). We define the solid deformation $\Phi_s(t) : \hat{\Omega}_s \longrightarrow \Omega_s(t) \subset \mathbb{R}^d$ as $\Phi_s(t;\, \hat{x}) = x$, displacement $\hat{u}_s(t, \hat{x}) = \Phi_s(t;\, \hat{x}) - \hat{x}$ and velocity $\hat{v}_s(t, x) = \frac{\partial}{\partial t}\hat{u}_s(t, x)$. In the deformed configuration $\Omega(t)$ we define solid velocity $v_s(t, x) = \hat{v}(t, \hat{x})$, fluid velocity v_f, and densities ρ_s and ρ_f of the solid and the fluid. For brevity, we will skip the time and space dependence of the variable if possible.

Note that the motion of the domain is unknown and thus it is a part of the solution. Let us set that aside and first discuss the FSI model.

2.1 The fluid-structure interaction model

We derive the model from the conservation of momentum and mass. That is in Eulerian coordinates we have the system of equations [24, 84] for the fluid

$$\begin{cases} \rho_f \frac{\mathrm{D}v_f}{\mathrm{D}t} - \nabla \cdot \sigma_f &= g, \\ \frac{\partial \rho_f}{\partial t} + \nabla \cdot (\rho v_f) &= 0 \end{cases} \tag{2.1}$$

in $\Omega_f(t)$, and for the solid

$$\begin{cases} \rho_s \frac{\mathrm{D}v_s}{\mathrm{D}t} - \nabla \cdot \sigma_s &= g, \\ \frac{\partial \rho_s}{\partial t} + \nabla \cdot (\rho v_s) &= 0 \end{cases} \tag{2.2}$$

in $\Omega_{s.}$, where g is given external force, σ_f and σ_s are Cauchy stress tensors. We assume that there is no external torque field and, as the consequence of conservation of angular momentum, both Cauchy stress tensors are symmetric. The operator ∇ denotes the spatial derivatives with respect to the coordinates x in the deformed configuration and the material derivative $\frac{\mathrm{D}\kappa}{\mathrm{D}t}$ of a vector field κ is defined as

$$\frac{\mathrm{D}\kappa}{\mathrm{D}t} = \frac{\partial}{\partial t}\kappa + \nabla\kappa\; v. \tag{2.3}$$

The solid stress tensor usually depends on the displacement that is related to velocity:

$$\hat{v}_s(t, x) = \frac{\partial}{\partial t}\hat{u}_s(t, x). \tag{2.4}$$

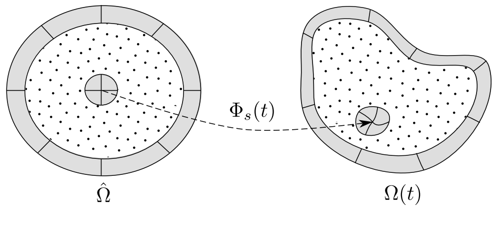


Figure 2.1: Initial domain $\hat{\Omega}$ and deformed domain Ω(t). Solid marked with gray, fluid marked with dots, lines represents transformation of material points by $\Phi_s(t)$

At the boundary $\Gamma = \partial\Omega$ partitioned into Neumann Γ_N and Dirichlet Γ_D parts, we set the boundary conditions

$$\begin{cases} v_s &= v_D^* \text{ on } \Gamma_D \cap \partial\Omega_s, \\ v_f &= v_D^* \text{ on } \Gamma_D \cap \partial\Omega_f, \\ \sigma_f \cdot n_f &= \tau_f^* \text{ on } \Gamma_N \cap \Omega_f, \\ \sigma_s \cdot n_s &= \tau_s^* \text{ on } \Gamma_N \cap \Omega_s. \end{cases} \tag{2.5}$$

where n_f and n_s denote unit outer normal of fluid and solid domains respectively. Finally, the initial conditions for this system

$$\begin{cases} v_s(t=0,\, x) &= v_s^* \text{ in } \hat{\Omega}_s, \\ \hat{u}_s(t=0,\, \hat{x}) &= 0 \\ v_f(t=0,\, x) &= v_f^* \text{ in } \hat{\Omega}_f. \end{cases} \tag{2.6}$$

On the interface between the fluid and the solid $\Gamma_i = \Omega_f \cap \Omega_S$ we set coupling conditions, the so-called kinematic continuity condition

$$v_f = v_s \qquad \text{on } \Gamma_i, \tag{2.7}$$

and the balance of traction

$$\sigma_f n_f + \sigma_s n_s = 0 \qquad \text{on } \Gamma_i. \tag{2.8}$$

Continuity of velocity. The non-slip condition at the fluid-solid interface Γ_i (2.7) is equivalent to the continuity of velocity. Thus, we define a single velocity field for both solid and fluid

$$v(x) = \begin{cases} v_s(x) & x \in \Omega_s \\ v_f(x) & x \in \Omega_f. \end{cases} \tag{2.9}$$

Note that there is no guarantee of any higher smoothness. In fact, in general case the gradient of velocity is discontinuous across the fluid-solid interface due to discontinuity of material parameters.

To close the system we need the relationship between the stress tensors σ_f and σ_s and velocity or displacement, that is the constitutive equations. For that we will need proper description of domain deformation.

2.2 Arbitrary Lagrangian-Eulerian formulation

Let $A(t) : \hat{\Omega} \longrightarrow \mathbb{R}^d$ be an arbitrary diffeomorphism mapping point $\bar{x}$ in reference domain $\hat{\Omega}$ to point $x(t, \bar{x}) = A(t;\, \bar{x})$ on deformed domain $\Omega(t)$. We assume that $A(t)$ is a sufficiently smooth (in both time and space) homeomorphism and in initial configuration $A(0) = \mathrm{Id}_{\hat{\Omega}}$. The popular choice of A is a continuous extension Φ of the solid deformation, i.e. $A(t) = \Phi_s(t)$ in $\hat{\Omega}_s$. We will also follow this way [24], but let us first derive the domain deformation description for an arbitrary mapping.

We change the primal unknowns in Equation (2.1) and Equation (2.2) to velocity $\bar{v}(t, \bar{x})$ and density $\bar{\rho}(t, \bar{x})$ defined on domain $\hat{\Omega}$ and define v and ρ by using mapping A. That is

$$v(t, x) = v(t, x(t, \bar{x})) = \bar{v}(t, \bar{x}) \tag{2.10}$$

since

$$x(t, \bar{x}) = A(t; \bar{x}), \qquad \bar{x}(t, x) = A^{-1}(t; x). \tag{2.11}$$

The bar over the symbol is used to distinguish the result of mapping A^{-1} from the (inverse) material motion of the solid specified by Φ_s^{-1}. Then, using the chain rule, we compute the time derivative of $v(t, x(\bar{x}, t))$ for fixed $\bar{x} \in \hat{\Omega}$, denoted by $\frac{\mathrm{d}}{\mathrm{d}t}v$:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} v(t, x(t, \bar{x})) &= \frac{\partial v}{\partial t}(t, x(t, \bar{x})) + \nabla v(t, x(t, \bar{x}))\, \frac{\partial x}{\partial t}(t, \bar{x}) \\ &= \frac{\partial v}{\partial t}(t, x(t, \bar{x})) + \nabla v(t, x(t, \bar{x}))\, \bar{v}_A(t, \bar{x}). \end{aligned} \tag{2.12}$$

The operator ∇ denotes the gradient with respect to coordinates x, i.e:

$$\nabla \bar{v}(t,\bar{x}) = \frac{\partial \bar{v}(t,\bar{x})}{\partial x} = \frac{\partial \bar{v}(t,\bar{x})}{\partial \bar{x}} \frac{\partial \bar{x}}{\partial x}. \tag{2.13}$$

In Equation (2.12) $\bar{v}_A(t,\bar{x})$ is the frame velocity of point $\bar{x}$, and $v_A(t,x)$ is the corresponding velocity defined on Ω, viz.

$$\bar{v}_A(\bar{x},t) = \frac{\partial x(t,\bar{x})}{\partial t} = \frac{\partial A(t,x)}{\partial t} \quad \text{and} \quad v_A(t,x) := \bar{v}_A(t,\bar{x}(t,x)) = \bar{v}_A(t,A^{-1}(t,x)). \tag{2.14}$$

By rearranging (2.12), we obtain the relation:

$$\frac{\partial v}{\partial t}(t,x(t,\bar{x})) = \frac{\mathrm{d}v}{\mathrm{d}t}(t,x(t,\bar{x})) - \nabla v(t,x(t,\bar{x}))\, v_A(t,x(t,\bar{x})) \tag{2.15}$$

or in short

$$\frac{\partial v}{\partial t} = \frac{\mathrm{d}v}{\mathrm{d}t} - \nabla v\, v_A. \tag{2.16}$$

Here and in the following, the explicit dependence on $x, \bar{x}$ and t is omitted for brevity. It is thus (implicitly) understood that quantities with a superimposed bar are defined on $\hat{\Omega}$ and depend on $\bar{x}$ (and time), e.g., $\bar{v} = \bar{v}(\bar{x},t)$, while those without a bar are defined on Ω and depend on x (and time), e.g., $v = v(x,t)$. We rewrite the material derivative in new setting

$$\frac{\mathrm{D}v}{\mathrm{D}t} = \frac{\mathrm{d}v}{\mathrm{d}t} + \nabla v\,(v - v_A), \tag{2.17}$$

substitute it to (2.1) and (2.2) and obtain

$$\begin{cases} \rho_k \frac{\mathrm{d}v}{\mathrm{d}t} + \rho_k \nabla v\,(v - v_A) - \nabla \cdot \sigma_k &= g, \\ \frac{\partial \rho_k}{\partial t} + \nabla \cdot (\rho v_i) &= 0 \end{cases} \tag{2.18}$$

for $A(t,\bar{x}) \in \Omega_k(t)$, where $k = f, s$. We will consider incompressible solid and fluid, thus $\frac{\partial}{\partial t}\rho_k = 0$ and we do not need to deal with the time derivative of density.

Let us expand the momentum balance from Equation (2.18) to obtain the formulation on the fixed domain $\hat{\Omega}$. By substituting Equation (2.13) to Equation (2.18) we obtain:

$$\rho_k \frac{\mathrm{d}}{\mathrm{d}t}\bar{v}(t,\bar{x}) + \rho_k \frac{\partial \bar{v}(t,\bar{x})}{\partial \bar{x}} \frac{\partial \bar{x}}{\partial x}\,(\bar{v}(t,\bar{x}) - \bar{v}_A(t,\bar{x})) - \left(\frac{\partial}{\partial \bar{x}} \frac{\partial \bar{x}}{\partial x}\right) \cdot \sigma_k = g(x(t,\bar{x})), \tag{2.19}$$

2.2.1 Choice of arbitrary mapping

Let us introduce a pseudo-displacement related to A

$$\bar{u}_A(t,\bar{x}) = A(t;\bar{x}) - \bar{x} = x - \bar{x}. \tag{2.20}$$

so that the frame velocity

$$\bar{v}_A = \frac{\mathrm{d}}{\mathrm{d}t}\bar{u}_A. \tag{2.21}$$

Eulerian setting The trivial choice $\bar{u}_A = 0$ ($A(t) = \mathrm{Id}_\Omega$) corresponds to the fully-Eulerian setting [27, 38] illustrated in Figure 2.2. In particular, this strategy does not have any other restrictions on domain deformation than $\Omega(t) = \Omega$, and, in case of finite element computations, simplifies mesh generation, but raises several problems. Most notable among them, interface tracking method is required, or loss of accuracy is encountered in case of some numerical methods. For more details see [85–87].

Arbitrary Lagrangian-Eulerian In many FSI problems, the mapping $\Phi_s(t)$ can be extended to a sufficiently smooth mapping $\Phi(t)$ defined on the whole domain $\hat{\Omega}$, that is

$$\begin{aligned} \Phi(t) &: \hat{\Omega} \to \Omega(t), \\ \Phi(t)|_{\hat{\Omega}_s} &= \Phi_s(t). \end{aligned} \tag{2.22}$$

In those cases the choice $A = \Phi$, or equivalently

$$\bar{u}_A = \mathrm{Ext}(\hat{u}_s) \tag{2.23}$$

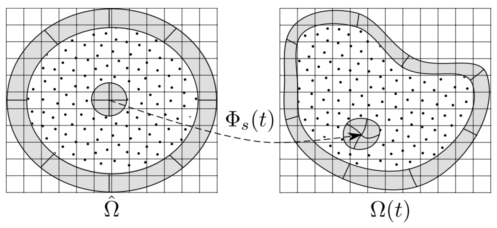


Figure 2.2: Eulerian frame for fluid-structure interaction problem. Grid represents stationary frame of reference while the lines indicate material points motion.

where $\mathrm{Ext}(\hat{u}_s) : \hat{\Omega} \longrightarrow \mathbb{R}^d$ is a sufficiently smooth extension of $\hat{u}_s$ onto $\hat{\Omega}$ is possible. This leads to the Lagrangian description of the solid, hence approach is called Arbitrary Lagrangian-Eulerian (ALE). The problems with interface tracking are automatically resolved. Additionally, a simpler application of higher-order accuracy methods is possible. The main limitation of the approach is the existence of a smooth enough extension of Φ_s for all times. In particular, this implies no topology changes of Ω_s. Moreover, in the case of numerical computations, the mesh distortion may occur especially if severe deformations are considered. In particular, with the ALE framework, one will not be able to resolve FSI problems involving contact. Despite this, numerous problems could be solved in this framework, applications may be found in [46, 47, 71].

Having this in mind, we decide to stick with ALE formulation. From this point on, we will use "hat" $\hat{\cdot}$ variables instead of "bar" $\bar{\cdot}$ ones as those fields are identical ($\bar{v} = \hat{v}$, $\bar{u}_s = \hat{u}_s$, $\bar{u}_A = \hat{u}_A$ and so on). To give an idea how the extension of $\hat{u}_s$ could be obtained let us briefly discuss some possible choices. We will go into details for a specific case in Section 5.4.

ALE mapping

Picking a suitable extrapolation of Φ_s is crucial for the method's applicability. One of the methods is solving an auxiliary equation. As an example, the so-called pseudo-elastic equation may be used[24]:

$$\begin{cases} \nabla \cdot (\mu_A \epsilon(\hat{u}_A)) & = 0 \quad \text{in } \Omega_f, \\ \hat{u}_A & = \hat{u}_s \quad \text{in } \hat{\Omega}_s. \end{cases} \tag{2.24}$$

with, possibly variable, coefficient μ_A. This choice is preferable due to its simplicity, but might result in a clumsily deformed mesh, especially if the solid is severely deformed. The key part of this extrapolation technique is the distribution of the parameter μ_A. We consider also the biharmonic equation [24, 50] as an auxiliary equation

$$\begin{cases} \Delta^2 \hat{u}_A & = 0 \quad \text{in } \Omega, \\ \hat{u}_A & = \hat{u}_s \quad \text{in } \hat{\Gamma}_i \\ \nabla \hat{u}_A \cdot n & = 0 \quad \text{on } \partial\Omega \end{cases} \tag{2.25}$$

that does not require any additional tuning, but is more computationally demanding.

2.3 Weak formulation in space

2.3.1 Momentum balance

We will first define the Sobolev space on domain ω

$$H^1_D(\omega) = \{v \in H^1(\omega)^d : v = 0 \text{ on } \Gamma_D \cap \partial\omega\}, \tag{2.26}$$

multiply momentum balance equations (Equation (2.18)) by test function $\phi \in H^1_D(\Omega)$, integrate by parts and obtain

$$\begin{aligned}
\int\limits_{\Omega_f} \rho_f \frac{\mathrm{d}v}{\mathrm{d}t} \cdot \phi \,\mathrm{d}x + \int\limits_{\Omega_f} \sigma_f : \nabla\phi \,\mathrm{d}x - \int\limits_{(\partial\Omega_f \cap \Gamma_N)\cup\Gamma_i} \sigma_f n_f \cdot \phi \,\mathrm{d}s &= \int\limits_{\Omega_f} g \cdot \phi \,\mathrm{d}x \qquad \forall \phi \in H^1_D(\Omega), \\
\int\limits_{\Omega_s} \rho_s \frac{\mathrm{d}v}{\mathrm{d}t} \cdot \phi \,\mathrm{d}x + \int\limits_{\Omega_s} \sigma_s : \nabla\phi \,\mathrm{d}x - \int\limits_{(\partial\Omega_s \cap \Gamma_N)\cup\Gamma_i} \sigma_s n_s \cdot \phi \, ds &= \int\limits_{\Omega_s} g \cdot \phi \,\mathrm{d}x \qquad \forall \phi \in H^1_D(\Omega),
\end{aligned} \tag{2.27}$$

where $\sigma : \tau$ denotes the scalar product of tensors σ and τ.

Elimination of interface traction conditions First we will refer to density without distinguishing between solid and fluid

$$\rho(x) = \begin{cases} \rho_s(x) & x \in \Omega_s, \\ \rho_f(x) & x \in \Omega_f, \end{cases} \tag{2.28}$$

as well as the traction at boundary

$$\tau^*(x) = \begin{cases} \tau^*_s(x) & x \in \Gamma_N \cap \partial\Omega_s, \\ \tau^*_f(x) & x \in \Gamma_N \cap \partial\Omega_f, \end{cases} \tag{2.29}$$

and the pressure

$$p(x) = \begin{cases} p_s(x) & x \in \Omega_s, \\ p_f(x) & x \in \Omega_f \end{cases} \tag{2.30}$$

that we will needed in the next section. Note that we do not assume any continuity of ρ or p at the interface.

We add Equations (2.27) and observe that the balance of the normal traction at the interface (2.8) implies

$$\int\limits_{\Gamma_i} \sigma_f n_f \cdot \phi \,\mathrm{d}s = - \int\limits_{\Gamma_i} \sigma_s n_s \cdot \phi \,\mathrm{d}s \tag{2.31}$$

and thus the integrals over Γ_i cancel out. The momentum conservation in weak form becomes:

$$\int\limits_{\Omega} \rho \frac{\mathrm{d}v}{\mathrm{d}t} \cdot \phi \,\mathrm{d}x + \int\limits_{\Omega_f} \sigma_f : \nabla\phi \,\mathrm{d}x + \int\limits_{\Omega_s} \sigma_s : \nabla\phi \,\mathrm{d}x = \int\limits_{\Omega} g \cdot \phi \, dx + \int\limits_{\Gamma_N} \tau^* \cdot \phi \,\mathrm{d}s \qquad \forall \phi \in H^1_D(\Omega). \tag{2.32}$$

2.3.2 Conservation of mass

For the mass conservation we will use test function $q \in L_2(\Omega)$. We multiply the equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0 \tag{2.33}$$

by q and obtain

$$\int\limits_{\Omega} \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) \right) q \,\mathrm{d}x = 0 \qquad \forall q \in L_2(\Omega) \tag{2.34}$$

2.4 Constitutive equations

2.4.1 Fluid model

We will consider classic incompressible Newtonian fluid:

$$\sigma_f = 2\mu_f \epsilon(v_f) + p_f I, \tag{2.35}$$

where p_f is the fluid pressure, μ_f is the viscosity and I is the identity tensor. The operator ϵ denotes the symmetrized gradient: $\epsilon(v) = \frac{1}{2}(\nabla v + \nabla v^T)$. The density of the fluid is constant, thus all its derivatives are zero and the mass conservation leads to

$$\nabla \cdot v_f = 0, \tag{2.36}$$

or in the weak form:

$$\int_{\Omega_S} q\,(\nabla\cdot v)\,\mathrm{d}x = 0 \qquad \forall q \in L^2(\Omega) \tag{2.37}$$

Stress-related integrals For the fluid we expand stress-related integral and use the fact that σ_f is symmetric, thus $\sigma_f : \nabla\phi = \sigma_f : \epsilon(\phi)$ and

$$\int_{\Omega_f} \sigma_f : \nabla\phi\ \mathrm{d}x = \int_{\Omega_f} 2\mu_f\epsilon(v) : \epsilon(\phi)\ \mathrm{d}x + \int_{\Omega_f} p_f\nabla\cdot\phi\ \mathrm{d}x. \tag{2.38}$$

2.4.2 Solid model

We will consider the Mooney-Rivlin solid model. We also discuss the Neo-Hookean model as a special case of Mooney-Rivlin model as it is a popular choice. We will need the gradient with respect to coordinates $\hat{x}$ on the reference domain $\hat{\Omega}$, that for an arbitrary vector $\hat{\kappa}$ field is defined as

$$\hat{\nabla}\hat{\kappa} = \frac{\partial\hat{\kappa}}{\partial\hat{x}} \tag{2.39}$$

and the deformation gradient

$$\hat{F} = \hat{\nabla}\Phi_s = I + \hat{\nabla}\hat{u}_s, \tag{2.40}$$

$$F = \left(\hat{\nabla}\Phi_s\right) \circ \Phi_s^{-1}. \tag{2.41}$$

The general rule is that the "hat" variables are related to fields on the domain $\hat{\Omega}$ while the variables without the hat are mapped via $A(t)$ (that is equal to Φ_s on $\hat{\Omega}_s$) onto deformed domain Ω.

We consider an incompressible material thus let us deal with the constraints first.

Conservation of mass and incompressibility constraints In our case ρ_s = const and the mass conservation leads to

$$\nabla\cdot v_s = 0. \tag{2.42}$$

or

$$\hat{J} = 1, \tag{2.43}$$

where $\hat{J} = \det\hat{F}$.

Mooney-Rivlin solid

The Cauchy stress in case of Mooney-Rivlin solid [88, 89] can be expressed using the left Cauchy–Green deformation tensor

$$B = FF^T \tag{2.44}$$

as follows

$$\sigma = \mu_1 B - \mu_2 B^{-1} + p_s^* I, \tag{2.45}$$

where $\mu_1 \geq 0$ and $\mu_2 \geq 0$ are material parameters and p_s^* is a Lagrange multiplier, related to solid pressure, which enforces incompressibility. Let us simplify Equation (2.45). First, we expand (2.40) using formula for the derivative of inverse function:

$$F = \left(\hat{\nabla}\Phi_s\right) \circ \Phi_s^{-1} = \left(\nabla(\Phi_s^{-1})\right)^{-1}. \tag{2.46}$$

Then, the inverse deformation gradient is

$$F^{-1} = \nabla\left(\Phi_s^{-1}\right), \tag{2.47}$$

For $\hat{x} \in \hat{\Omega}_s$, $x \in \Omega_s$ so that $\hat{x} = \Phi_s^{-1}(t;\, x)$ from the definition of the solid displacement

$$\hat{u}_s(t,\hat{x}) = \Phi_s(t;\,\hat{x}) - \hat{x} \tag{2.48}$$

it follows

$$\Phi_s^{-1}(x) = \underbrace{\Phi_s(t;\, \Phi_s^{-1}(t;\, x))}_{x} - \underbrace{\hat{u}_s(t, \Phi_s^{-1}(t;\, x))}_{u(t,x)} \tag{2.49}$$

and thus

$$\Phi_s^{-1}(x) = x - u_s(t, x). \tag{2.50}$$

The inverse of the deformation gradient is

$$F^{-1} = \nabla\left(\Phi_s^{-1}\right) = 1 - \nabla u_s, \tag{2.51}$$

and by substituting it into Equation (2.45) we obtain:

$$\sigma = \mu_1 FF^T + \mu_2\left(2\epsilon(u_s) - (\nabla u_s)^T\nabla u_s - I\right) + p_s^* I \tag{2.52}$$

With term $\mu_1 FF^T$ we will deal in the weak form. Let us consider the stress-related integral:

$$\int\limits_{\Omega_s} \sigma_s : \nabla\phi \,\mathrm{d}x = \int\limits_{\Omega_s} \left(\mu_1 FF^T + \mu_2\left(2\epsilon(u_s) - (\nabla u)^T\nabla u - I\right) + p_s^* I\right) : \nabla\phi_s \,\mathrm{d}x. \tag{2.53}$$

By changing the coordinates to $\hat{x} = \Phi_s^{-1}(t;\, x)$ in the integral:

$$\int\limits_{\Omega_s} FF^T : \nabla\phi \,\mathrm{d}x = \int\limits_{\hat{\Omega}_s} \hat{F}\hat{F}^T : \hat{\nabla}\hat{\phi}\hat{F}^{-1}\,\hat{J}\mathrm{d}\hat{x}$$

we obtain:

$$\int\limits_{\Omega_s} FF^T : \nabla\phi \,\mathrm{d}x = \int\limits_{\hat{\Omega}_s} (\hat{J}\hat{F}\hat{F}^T\hat{F}^{-T}) : \hat{\nabla}\hat{\phi}\,\mathrm{d}\hat{x} = \int\limits_{\hat{\Omega}_s} \hat{F} : \hat{\nabla}\hat{\phi}\,\mathrm{d}\hat{x} \tag{2.54}$$

where $\hat{J} = 1$ (Equation (2.43)) and thus the stress integral becomes:

$$\int\limits_{\Omega_s} \sigma_s : \nabla\phi \,\mathrm{d}x = \int\limits_{\hat{\Omega}_s} \mu_1\hat{F} : \hat{\nabla}\hat{\phi}\,\mathrm{d}\hat{x} + \int\limits_{\Omega_s} \left(\mu_2\left(2\epsilon(u_s) - (\nabla u)^T\nabla u - I\right) + p_s^* I\right) : \nabla\phi_s \,\mathrm{d}x. \tag{2.55}$$

In case of undeformed solid, i. e. $\hat{u}_s = 0$ the deformation gradient $\hat{F} = I$ and the stress integral becomes:

$$\int\limits_{\Omega_s} \sigma_s : \nabla\phi \,\mathrm{d}x = \int\limits_{\hat{\Omega}_s} \mu_1 I : \hat{\nabla}\hat{\phi}\,\mathrm{d}\hat{x} - \int\limits_{\hat{\Omega}_s} \left(\mu_2 I + p_s^* I\right) : \hat{\nabla}\hat{\phi}\,\mathrm{d}\hat{x}. \tag{2.56}$$

If there is no stress in the solid $\sigma_s = 0$ and $p_s^* = \mu_1 - \mu_2$. This is inconsistent with the fluid model, where the zero stress implies $p_f = 0$, thus we will use the shifted Lagrange multiplier:

$$p_s = p_s^* - \mu_1 + \mu_2. \tag{2.57}$$

In case of $\mu_2 = 0$ we obtain the neo-Hookean solid model. Moreover, in 2D problems incompressible Mooney-Rivlin solid is equivalent to neo-Hookean one with shear modulus $\mu_s = \mu_1 + \mu_2$. In the following we restrict ourselves to the case $\mu_1 = 0$. This choice allows us to simplify our derivation and implementation as well. The second one is important for us since we are developing a matrix-free solver. Implementing integration in both Ω_s and $\hat{\Omega}_s$ is possible, but requires additional work.

With that said, with $\mu_1 = 0$ and $\mu_2 = \mu_s$ the integral involving stress is transformed as follows:

$$\int\limits_{\Omega_s} \sigma_s : \nabla\phi \,\mathrm{d}x = \int\limits_{\Omega_s} 2\mu_s\, \epsilon(u_s) : \nabla\phi \,\mathrm{d}x - \int\limits_{\Omega_s} \mu_s\, (\nabla u_s)^T\nabla u : \nabla\phi \,\mathrm{d}x + \int\limits_{\Omega_s} p_s\nabla\cdot\phi\mathrm{d}x = \tag{2.58}$$

$$= \int\limits_{\Omega_s} 2\mu_s\, \epsilon(u_s) : \epsilon(\phi)\,\mathrm{d}x - \int\limits_{\Omega_s} \mu_s\, (\nabla u_s)^T\nabla u_s : \epsilon(\phi)\,\mathrm{d}x + \int\limits_{\Omega_s} p_s\nabla\cdot\phi\mathrm{d}x \tag{2.59}$$

Volumetric damping

The divergence of velocity corresponds to local production of mass

$$\nabla\cdot(\rho_s v_s) = -\frac{\partial\rho_s}{\partial t}. \tag{2.60}$$

Thus, the constraint (Equation 2.42) guarantees that the time derivative of the volume is zero

$$\frac{\partial \rho_s}{\partial t} = 0 \tag{2.61}$$

with initial a density $\rho_s(0, x) = \rho_{s\,0}$. If for whatever reasons, the density is disturbed it will remain unchanged through the rest of the time. This is an issue in the numerical solution of the problem, since the volume may change as a result of the numerical errors. Let us replace the right-hand side of the Equation (2.61)

$$\frac{\partial \rho_s}{\partial t} = -\frac{1}{\eta_V}\left(\frac{\rho_s}{\rho_{s\,0}} - 1\right) \tag{2.62}$$

so that the solution would approach the density $\rho_{s\,0}$ regardless of what is the starting point is. The numerical parameter η_V controls the rate of density correction. The fraction $\rho_s/\rho_{s\,0}$ could be expressed by using the deformation gradient

$$\frac{\rho_s}{\rho_{s\,0}} = \det(\hat{F}). \tag{2.63}$$

Substituting Equations (2.60) and (2.63) into the Equation (2.62) leads to the the new constraint for the solid with volumetric damping

$$\nabla \cdot (\rho_s v) = -\frac{1}{\eta_V}(\det(\hat{F}) - 1). \tag{2.64}$$

For the weak form, we multiply this equation by a test function $q \in L^2(\Omega)$ and integrate over Ω

$$\int\limits_{\Omega_s} q\,(\nabla \cdot v)\,\mathrm{d}x = -\int\limits_{\Omega} \frac{1}{\eta_V}(\det(\hat{F}) - 1)q\,\mathrm{d}x \quad \forall q \in L^2(\Omega). \tag{2.65}$$

By changing the coordinates in the right-hand side integral we obtain

$$\int\limits_{\Omega_s} q\,(\nabla \cdot v)\,\mathrm{d}x = -\int\limits_{\hat{\Omega}} \frac{1}{\eta_V}(\det(\hat{F}) - 1)\hat{q}\,J\mathrm{d}\hat{x} \quad \forall q \in L^2(\Omega) \tag{2.66}$$

or equivalently

$$\int\limits_{\Omega_S} q\,(\nabla \cdot v)\,\mathrm{d}x = -\int\limits_{\hat{\Omega}} \frac{1}{\eta_V}(\det(\hat{F}) - 1)\hat{q}\,\mathrm{d}\hat{x} \quad \forall q \in L^2(\Omega), \tag{2.67}$$

since $J = 1$.

2.5 Complete weak formulation with ALE

Let us summarize the results from this chapter by presenting the weak form of the FSI problem in ALE frame of reference. Before that, let us unify the incompressibility constraints into the single equation. We exploit the fact that in case of both fluid and solid the mass conservation has the same form in the domains Ω_f and Ω_S correspondingly. By adding equations (2.67) and (2.37) we obtain

$$\int\limits_{\Omega} q\,(\nabla \cdot v)\,\mathrm{d}x = -\int\limits_{\hat{\Omega}_s} \frac{1}{\eta_V}(\det(\hat{F})\hat{q}\,\mathrm{d}\hat{x} \quad \forall q \in L^2(\Omega). \tag{2.68}$$

We also introduce $(\cdot,\cdot)_\omega$, the $L^2(\omega)$ scalar product

$$(p, q)_\omega = \int\limits_{\omega} p \cdot q \mathrm{d}x. \tag{2.69}$$

and forms

$$\begin{aligned} a(v, \phi) &= (\rho[\frac{\mathrm{d}v}{\mathrm{d}t} + \nabla v\,(v - v_A)], \phi)_\Omega + (2\mu_f \epsilon(v) : \epsilon(\phi))_{\Omega_f} \\ &\quad + (2\mu_s \epsilon(u), \epsilon(\phi))_{\Omega_S} - (\mu_s \nabla u_s^T\,\nabla u_s, \epsilon(\phi))_{\Omega_S}, \\ b(v, q) &= (\nabla \cdot v, q)_\Omega, \\ g(\phi) &= (g, \phi)_\Omega. \end{aligned} \tag{2.70}$$

Then, the FSI problem is to find for all times $t \in [0, T]$: $v \in H^1(\Omega)^d$, $p \in L^2(\Omega)$, $\hat{u}_s \in H^1_D(\hat{\Omega}_s)^d$, $\hat{u}_A \in H^1_D(\hat{\Omega})^d$ such that:

$$\begin{cases} a(v, \phi) + b(\phi, p) & = g(\phi) \quad \forall \phi \in H^1_D, \\ b(v, q) & = -\left(\frac{1}{\eta_V}(\det(\hat{F}) - 1), \hat{q}\right)_{\hat{\Omega}_s} \qquad \forall q \in L^2(\Omega), \\ \partial_t \hat{u}_s & = v_s \circ A^{-1}, \\ \hat{u}_A & = \mathrm{Ext}(\hat{u}_s), \\ A & = \mathrm{Id} + \hat{u}_A. \end{cases} \tag{2.71}$$

The subdomains Ω_s and Ω_f appearing in integrals in Equation (2.70) are defined as image of undeformed ones: $\Omega_s = A(\hat{\Omega}_s)$ and $\Omega_f = A(\hat{\Omega}_f)$. The extension of the solid displacement $\mathrm{Ext}(\hat{u}_s)$ is defined by an auxiliary equation as discussed in Section 2.2.1. Additionally, the following initial conditions have to be met:

$$\begin{cases} v_s(t = 0,\, x) & = v_s^* \text{ in } \hat{\Omega}_s, \\ \hat{u}_s(t = 0,\, \hat{x}) & = u_s^* \\ v_f(t = 0,\, x) & = v_f^* \text{ in } \hat{\Omega}_f. \end{cases} \tag{2.72}$$

Our weak formulation is similar to the ones appearing in literature [23, 56, 66, 81]. The first difference is the right-hand side of the continuity equation, arising from the reformulated solid constraints. Secondly, we formulated the problem on the unknown domain Ω to simplify derivation of the time integration scheme.

Time and space discretization 3

In this chapter, we gradually move from the weak form of the FSI problem to its discretization. We first define discrete set of points in time interval referred as time steps. At each time step, we introduce an approximation of the time derivative, eliminate the solid displacement to obtain a velocity-based formulation with decoupled displacement. We then proceed to the space discretization, we introduce the finite element discretization by restricting the test and trial function spaces to the finite-dimensional one. Finally, we formulate the system of linear equations.

3.1 Time discretization

We consider the uniform time discretization, i.e. we replace the time interval $[0, T]$ with the set of discrete points $\{t^0, ..., t^N\}$ where $t^n = n\Delta t$, with the time step size $\Delta t = T/N$.

3.1.1 Time-discrete fields

At the n-th time step t^n, we define $\hat{v}^n$, $\hat{u}_s^n$ and $\hat{u}_A^n$ as an approximation of $\hat{v}$, $\hat{u}_s$ and $\hat{u}_A$ respectively, i.e.

$$\begin{aligned} \hat{v}(t^n, \hat{x}) &\approx \hat{v}^n(\hat{x}), \\ \hat{u}_s(t^n, \hat{x}) &\approx \hat{u}_s^n(\hat{x}), \\ \hat{u}_A(t^n, \hat{x}) &\approx \hat{u}_A^n(\hat{x}), \end{aligned} \tag{3.1}$$

The approximate displacement defines the approximation of the solid deformation at time t^n:

$$\Phi_s(t^n, \hat{x}) \approx \Phi_s^n(\hat{x}) = \hat{x} + \hat{u}_s^n(\hat{x}), \qquad \hat{x} \in \hat{\Omega}_s \tag{3.2}$$

and therefore the solid domain

$$\Omega_s(t^n) \approx \Omega_s^n = \Phi_s^n(\hat{\Omega}_s). \tag{3.3}$$

The pseudo-displacement defines the approximation of the mapping A:

$$A(t^n, \hat{x}) \approx A^n(\hat{x}) = \hat{x} + \hat{u}_A^n(\hat{x}) \qquad \hat{x} \in \hat{\Omega} \tag{3.4}$$

and therefore the domains

$$\Omega(t^n) \approx \Omega^n = A^n(\hat{\Omega}), \qquad \Omega_f(t^n) \approx \Omega_f^n = A^n(\hat{\Omega}_f). \tag{3.5}$$

At the time step n, the relation between point $x_n \in \Omega^n$ and point $\hat{x} \in \hat{\Omega}$ is defined by mapping A^n, i.e:

$$x_n(\hat{x}) = A^n(\hat{x}), \tag{3.6}$$

and the inverse relation between point $x \in \Omega^n$ and point $\hat{x}_n \in \hat{\Omega}$

$$\hat{x}_n(x) = (A^n)^{-1}(x).$$

This leads to definition of spatial derivatives of arbitrary vector field κ defined on Ω^n. The gradient and symmetric gradient are defined as:

$$\nabla\kappa = \frac{\partial}{\partial x}\kappa, \qquad \epsilon(\kappa) = \frac{\nabla\kappa + (\nabla\kappa)^T}{2} \tag{3.7}$$

where $x = x_n(\hat{x})$ is defined by (3.6). That is, the spatial derivatives are associated with transformation A^n. Since A^n is defined by pseudo-displacement, those derivatives are implicitly defined by via $\hat{u}_A^n$.

In the subsequent derivations, we are going to replace A^n with its approximation $A^\#$. Note that this approximation also affects the derivatives.

3.1.2 Time derivative approximation on a moving domain

We approximate the time derivative of arbitrary field $\hat{\kappa}$ defined on $\hat{\Omega} \times [0, T]$ by the backward differentiation formula (BDF) [79] of order k, that is the following stencil

$$\left.\frac{\partial \hat{\kappa}(\hat{x}, t)}{\partial t}\right|_{t=t^n} \approx \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i \hat{\kappa}^{n-i}(\hat{x}) \tag{3.8}$$

where the coefficients $\alpha_0 = 1$ and

$$\gamma = 1, \quad \alpha_1 = -1 \qquad \text{for } k = 1 \tag{3.9}$$

and

$$\gamma = \frac{2}{3}, \quad \alpha_1 = -\frac{4}{3}, \quad \alpha_2 = \frac{1}{3} \qquad \text{for } k = 2. \tag{3.10}$$

As a consequence of the second Dahlquist barrier [90], any linear multistep method of order greater than 2 is not A-stable. On the other hand, the governing equation on the solid domain is a stiff hyperbolic equation, hence the unconditional stability is crucial for us. We have to admit that we learned it in a hard way, that is by implementing the 3rd order scheme and finding out that it is only stable if the time step size is insanely low. What is even worse, the maximum time step size went down with mesh refinement. For this reason, we do not consider any higher-order method.

Displacement Let us approximate the time derivative of the displacement according to (3.8)

$$\left.\frac{\partial \hat{u}_s(t, \hat{x})}{\partial t}\right|_{t=t^n} \approx \delta_k \hat{u}_s^n = \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i \hat{u}_s^{n-i}(\hat{x}). \tag{3.11}$$

Velocity The derivative of velocity $\frac{\mathrm{d}}{\mathrm{d}t} v(t^n, x(t^n, \hat{x}))$ is computed for the fixed $\hat{x} = \hat{x}(t^n, x)$, therefore

$$\left.\frac{\mathrm{d}v}{\mathrm{d}t}(t, x(t, \hat{x}))\right|_{t=t^n} = \left.\frac{\partial \hat{v}}{\partial t}(t, \hat{x}(t^n, x))\right|_{t=t^n} \approx \delta_k v^n \approx \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i \hat{v}^{n-i}(\hat{x}_n(x)). \tag{3.12}$$

For time step t^n, we introduce the velocity $\hat{v}^{n-i}$ mapped by A^n on Ω^n

$$v^{n,i}(x_n(t^n, \hat{x})) = \hat{v}^{n-i}(\hat{x}). \tag{3.13}$$

Note that the following relation between $v^{n,i}$ and v^{n-i} holds

$$v^{n,i}(x) = v^{n-i}(x_{n-i}(\hat{x}_n(x))) \tag{3.14}$$

or equivalently

$$v^{n,i} = v^{n-i} \circ A^{n-i} \circ (A^n)^{-1}. \tag{3.15}$$

This allows us to compute approximation of the velocity derivative in domain Ω^n [24, 55, 66]:

$$\delta_k v^n = \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i \hat{v}^{n-i}(\hat{x}_n(x)) = \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i v^{n,i}(x). \tag{3.16}$$

Frame velocity We obtain the approximate frame velocity by applying stencil (3.8) as follows

$$\hat{v}_A(t^n, \hat{x}) = \left.\frac{\mathrm{d}}{\mathrm{d}t} \hat{u}_A(t^n, \hat{x})\right|_{t=t^n} \approx \hat{v}_A^n(\hat{x}) = \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i \hat{u}_A^{n-i}(\hat{x}), \tag{3.17}$$

and

$$v_A^n(x) = \frac{1}{\gamma \Delta t} \sum_{i=0}^{k} \alpha_i \hat{u}_A^{n-i}(\hat{x}(x)). \tag{3.18}$$

Similarly to formula (3.13), we introduce the displacement $\hat{u}_A^{n-i}$ mapped by A^n on Ω^n

$$u_A^{n,i}(x) = \hat{u}_A^{n-i}(\hat{x}_n(x)), \tag{3.19}$$

or equivalently,

$$u_A^{n,i} = u_A^{n-i} \circ A^{n-i} \circ (A^n)^{-1}. \tag{3.20}$$

3.1.3 Time integration scheme

We replace the derivatives $\frac{\mathrm{d}}{\mathrm{d}t}v$ and $\frac{\partial}{\partial t}\hat{u}$ in Equation (2.71) by their approximates $\delta_k \hat{u}^n$ (Equation (3.11)) and $\delta_k v^n$(Equation (3.12)) correspondingly to obtain the fully implicit time integration scheme.

Fully implicit scheme

In the fully implicit scheme with $\hat{v}^0$, $\hat{u}_s^0$, $\hat{u}_A^0$ (Equation (2.72)) defined by the initial conditions, the following problem is to be solved, for every $n > 0$ such that $n\Delta t \in [0, T]$ find $v^n \in H^1(\Omega)^d$, $p^n \in L^2(\Omega^n)$, $\hat{u}_s^n \in H_D^1(\hat{\Omega}_s)^d$, $\hat{u}_A^n \in H_D^1(\hat{\Omega})^d$ such that

$$\begin{cases} a(v, \phi) + b(\phi, p) & = g(\phi) \quad \forall \phi \in H_D^1, \\ b(v^n, q) & = -\left(\frac{1}{\eta_V}(\det(\hat{F}) - 1), \hat{q}\right)_{\hat{\Omega}_s} \qquad \forall q \in L^2(\Omega), \\ \delta_k \hat{u}_s^n & = \hat{v}_s^n, \\ \hat{u}_A^n & = \mathrm{Ext}(\hat{u}_s^n), \end{cases} \tag{3.21}$$

where

$$\begin{aligned} a_i(v^n, \phi) &= (\rho\, \delta_k v^n + \rho\, \nabla v^\star\, v^\circ, \phi)_{\Omega^n} + (2\mu_f \epsilon(v^n) : \epsilon(\phi))_{\Omega_f^n} \\ &\quad + (2\mu_s \epsilon(u_s^n), \epsilon(\phi))_{\Omega_s^n} - \left(\mu_s (\nabla u_s)^T\, \nabla u_s^n, \epsilon(\phi)\right)_{\Omega_S^n}, \\ b(v^n, q) &= (\nabla \cdot v^n, q)_{\Omega^n}, \end{aligned} \tag{3.22}$$

and the velocities $v^\star$ and v_f° in the advection part are

$$\begin{aligned} v^\circ &= v^n - v_A^n, \\ v^\star &= v^n. \end{aligned} \tag{3.23}$$

Notice that in Ω_s we have $v^n = v_A^n$ and $v^\circ = 0$. The problem (3.21) is nonlinear, thus it could be solved by Newton method [37, 60, 61] or another fixed-point method. It could be also linearized without greater impact on stability [58]. We will propose a fixed point method, however our experiments show that one or 2 iterations are enough.

We fix the time step number n and modify the scheme by replacing some of the quantities with their explicit approximations to end up with a linear problem at each time step. We expect that it may have some effect on stability, however, as it follows from our experience, this is not an issue since the main limiting factor of time step size remains the accuracy (details in Section 6.4).

We first linearize the part related to the domain motion and solid stress, then deal with the advection.

Explicit scheme for the geometry

Let us the decouple domain deformation dependence from the integrals in the form $a(\cdot, \cdot)$ according to *Geometry-Convective Explici*t (*GCE*) scheme [55, 57, 66]. We redefine the approximate solid deformation Φ_s^n as

$$\Phi_s^n(\hat{x}) = \hat{x} + \hat{u}_s^\#(\hat{x}), \tag{3.24}$$

where $\hat{u}_s^\#$ is an explicit approximation of the solid displacement $\hat{u}_s^n$. According to the BDF scheme the solid displacement is

$$\hat{u}_s^n = \gamma \Delta t \hat{v}_s^n - \sum_{i=1}^{k} \alpha_i \hat{u}_s^{n-i}. \tag{3.25}$$

We replace the velocity $\hat{v}_s^n$ by its approximation $\hat{v}_s^\#$ to obtain:

$$\hat{u}_s^\# = \gamma \Delta t \hat{v}_s^\# - \sum_{i=1}^{k} \alpha_i \hat{u}_s^{n-i}. \tag{3.26}$$

For now, we will use $\hat{v}_s^\# = \hat{v}_s^{n-1}$, we later propose a better approximation by introducing a predictor. Using the BDF-like formula in (3.26) may seem inappropriate. Replacing it with a more suitable linear multistep method such as the Adams-Bashford scheme may seem like a better idea. However, this is exactly the solution we tried at first and, unfortunately, it turns to have poor stability properties even if considering solid part only. In fact, we did not manage to find a time step size small enough for the scheme to be stable. On the other hand, we did not experience any stability issues when (3.26) was used, while it produces the meaningful results as presented in Section 5.3 (solid dynamics) and Chapter 6 (FSI problem).

ALE mapping We consider the ALE setting, that is

$$\hat{u}^n_A = \mathrm{Ext}(\hat{u}^n_s). \tag{3.27}$$

We will approximate the new pseudo-displacement at each time step by using the extension of $\hat{u}^\#_s$:

$$\hat{u}^n_A \approx \hat{u}^\#_A = \mathrm{Ext}(\hat{u}^\#_s).$$

The pseudo-displacement defines the associated mapping $A^\#$ which in turn defines the domain $\Omega^\# = A^\#(\hat{\Omega})$ that is an approximation of Ω^n. The approximation of the fluid domain is defined as $\Omega^\# \setminus \Omega^\#_s$. We also redefine the fields specified on Ω^n in Section 3.1.2: $\delta_k v^n$ and $v^n_A(x)$ by replacing mapping A^n with $A^\#$.

Spatial derivatives The pseudo-displacement $\hat{u}^\#_A$ defines the arbitrary mapping $A^\# : \hat{\Omega} \longrightarrow \Omega^\#$, which in turn defines the derivative operators ∇ and $\epsilon(\cdot)$. Both derivative operators appear only inside integrals over the corresponding domains, therefore it should be clear to which transformation we refer to. In particular, we apply the following approximation

$$(\rho\, \delta_k v^n + \rho \nabla v^\star\, v^\circ, \phi)_{\Omega^n} \approx (\rho\, \delta_k v^n + \rho \nabla v^\star\, v^\circ, \phi)_{\Omega^\#} \tag{3.28}$$

where the quantities and derivatives on the left-hand side are defined on Ω^n while the ones on the right-hand side are defined on $\Omega^\#$. We will replace all remaining terms involving A^n with $A^\#$ in Section 3.1.3.

Frame velocity The frame velocity is obtained by substituting Equation (3.27) for the displacements from previous time steps to Equation (3.17), and we obtain:

$$\hat{v}^n_A = \frac{1}{\gamma \Delta t}\left(\hat{u}^\# + \sum_{i=1}^{k} \alpha_i \mathrm{Ext}(\hat{u}^{n-i}_s)\right). \tag{3.29}$$

We will consider two methods of computing the extension, in both cases the operator $\mathrm{Ext}(\cdot)$ is linear and thus the frame velocity in our case is

$$\hat{v}^n_A = \mathrm{Ext}(\hat{v}^\#_s). \tag{3.30}$$

In the implementation, computing the extrapolation of velocity may be handier, as we will see later.

Solid stress linearization

In the system (3.21) there are three solid-related non-linear forms: volumetric damping and two integrals arising from the stress. For the first one we do not require higher-order accuracy, we only need effective damping of excess volume with the lowest possible impact on stability. We set

$$\det(\hat{F}) \approx \det(\hat{F}^{n-1}) = \det(I + \hat{\nabla}\hat{u}^{n-1}_s) \tag{3.31}$$

that corresponds to applying the forward Euler method to Equation (2.62). In the quadratic part of the solid stress we use the extrapolation of displacement $\hat{u}^\#_s$

$$\left(\left(\nabla u^n_s\right)^T \nabla u^n_s, \epsilon(\phi)\right)_{\Omega^n} \approx \left(\left(\nabla u^\#_s\right)^T \nabla u^\#_s, \epsilon(\phi)\right)_{\Omega^\#}. \tag{3.32}$$

Finally, we approximate the bilinear form $(2\mu_s \epsilon(u^n_s), \epsilon(\phi))_{\Omega^n_s}$ with a semi-implicit formula as follows:

$$\left(2\mu_s \epsilon(u^n_s), \epsilon(\phi)\right)_{\Omega^n_s} \approx \left(2\mu_s \epsilon(u^n_s), \epsilon(\phi)\right)_{\Omega^\#_s} \tag{3.33}$$

Notice that in Equation (3.32) and Equation (3.33) derivatives on the left-hand side are defined by the transformation A^n, while the ones on the right-hand side are defined by $A^\#$. In Equation (3.33) we additionally replaced $\hat{u}^n_s \circ (A^n)^{-1}$ with $\hat{u}^n_s \circ (A^\#)^{-1}$.

Before proceeding to the advection, let us rearrange the solid part of the problem.

Velocity formulation Following [56, 66]we use the relation between the displacement and velocity — Equation (3.11), and, by substituting $\hat{u}_s^n = \gamma\Delta t \hat{v}_s^n - \sum_{i=1}^{k} \alpha_i \hat{u}_s^{n-i}$

$$\mu_s(\epsilon(u_s^n), \epsilon(\phi))_{\Omega_s^\#} = \underbrace{\mu_s \gamma \Delta t \left(\epsilon(v_s^n), \epsilon(\phi)\right)_{\Omega_s^\#}}_{\text{implicit part}} - \underbrace{\mu_s \left(\epsilon\left(\sum_{i=1}^{k} \alpha_i u_s^{n,i}\right), \epsilon(\phi)\right)_{\Omega_s^\#}}_{\text{explicit part}} \tag{3.34}$$

we decoupled the displacement from the system (3.21).

Semi-implicit advection

The advection term consists of two "velocities": v° and $v^\star$. We set the first one as explicit extrapolation [91]

$$\begin{aligned} v^\circ &= v^{n,1} - v_A^n \qquad \text{for } k = 1, \\ v^\circ &= 2(v^{n,1} - v_A^n) - (v^{n,2} - v_A^{n,1}) \qquad \text{for } k = 2. \end{aligned} \tag{3.35}$$

As $v^\star$ we consider two choices, the explicit advection

$$\begin{aligned} v^\star &= v^{n,1} \qquad \text{for } k = 1, \\ v^\star &= 2v^{n,1} - v^{n,2} \qquad \text{for } k = 2 \end{aligned} \tag{3.36}$$

and the semi-implicit advection

$$v^\star = v^n. \tag{3.37}$$

The scheme with the explicit advection for the Navier-Stokes equation is also known as IMEX [92] method. The main advantage of the explicit scheme is the symmetric form $a_v(\cdot,\cdot)$ (that will be defined later on), while the semi-implicit one offers better stability. For further discussion we refer to work by Dong [93] or Turek [92].

3.1.4 Predictor-corrector scheme in ALE setting

Let us first conclude all the manipulation we have done to the fully implicit scheme. The problem we solve at the time step $n > 0$ is to find $v^n \in H^1(\Omega^\#)^d$, $p^n \in L^2(\Omega^\#)$, $\hat{u}_s^n \in H_D^1(\hat{\Omega}_s)^d$, $\hat{u}_A^n \in H_D^1(\hat{\Omega})$ so that

$$\begin{cases} a_v(v,\phi) + b(\phi, p^n) &= g_v(\phi) \quad \forall \phi \in H_D^1(\Omega), \\ b(v,q) &= g_p(q) \quad \forall q \in L^2(\Omega^\#), \\ \hat{u}_A^\# &= \mathrm{Ext}(\hat{u}_s^\#), \\ \hat{u}_s^n &= \gamma \Delta t \hat{v}_s^n - \sum_{i=1}^{k} \alpha_i \hat{u}_s^{n-i}, \end{cases} \tag{3.38}$$

where:

$$\begin{aligned} a_v(v,\phi) &= (\rho\, \delta_k v^n + \rho \nabla v^\star\, v^\circ, \phi)_{\Omega^\#} + (2\mu_f \epsilon(v_f^n) : \epsilon(\phi))_{\Omega_f^\#} + \mu_s \gamma \Delta t \left(\epsilon(v_s^n), \epsilon(\phi)\right)_{\Omega_s^\#}, \\ b(v,q) &= (\nabla \cdot v^n, q)_{\Omega^\#}, \end{aligned} \tag{3.39}$$

and

$$\begin{aligned} g_v(\phi) &= (g,\phi)_\Omega - \mu_s \left(\epsilon\left(\sum_{i=1}^{k} \alpha_i u_s^{n,k}\right), \epsilon(\phi)\right)_{\Omega_s^\#} + \mu_s \left(\nabla \left(u_s^\#\right)^T \nabla u_s^\#, \epsilon(\phi)\right)_{\Omega_s^\#}, \\ g_p(q) &= -\left(\frac{1}{\eta_V}(\det(I + \hat{\nabla}\hat{u}_s^{n-1}), \hat{q}\right)_{\hat{\Omega}_s}. \end{aligned} \tag{3.40}$$

Notice that in Equation (3.38) we defined the problem on the domain $\Omega^\#$ instead of $\hat{\Omega}$ as in Equation (3.21). The quantities on the reference domain are obtained by remapping the ones form domain $\Omega^\#$ to $\hat{\Omega}$ via the inverse of mapping $A^\#$. Let us tidy up the formulation and present the time-stepping algorithm.

Having in mind that operator $\mathrm{Ext}(\cdot)$ is linear with respect to boundary condition implied by the solid displacement, let us rearrange the last two equations in Equation (3.38) to facilitate the implementation. We first define the unified displacement

$$\hat{u}^n = \mathrm{Ext}(\hat{u}^n_s) \tag{3.41}$$

so that the equation $\delta_k \hat{u}^n_s = \hat{v}^n_s$ is replaced by

$$\delta_k \hat{u}^n = \mathrm{Ext}(\hat{v}^n_s). \tag{3.42}$$

The extrapolated domain displacement $\hat{u}^{\#}_A$ according to Equation (3.29) is

$$\hat{u}^{\#}_A = \gamma \Delta t \hat{v}^{\#} - \sum_{i=1}^{k} \alpha_i \hat{u}^{n-i} \tag{3.43}$$

where $\hat{v}^{\#} = \mathrm{Ext}(\hat{v}^n_s) = \hat{v}^n_{\mathrm{Ext}}$. Then, the system in Equation (3.38) could be solved in few steps as demonstrated in the Algorithm 1. Note that the extension of velocity $\hat{v}^n_{\mathrm{Ext}}$ from time step n is used twice: in step 4 of time step n and in step 1 of time step $n+1$.

Algorithm 1: Geometry-explicit scheme

```
Data: $\hat{u}^{n-1}_A, v^{n-1}, \hat{v}^{n-1}_{\mathrm{Ext}}, \hat{u}^{n-1}$
Result: $\hat{u}^n_A, v^n, \hat{v}^n_{\mathrm{Ext}}, \hat{u}^n$
1 begin
2     $v^n_{\mathrm{Ext}} := v^{n-1}_{\mathrm{Ext}}$
3     $\hat{u}^{\#}_A := \gamma \Delta t \hat{v}^n_{\mathrm{Ext}} - \sum_{i=1}^{k} \alpha_i \hat{u}^{n-k}$        ◃ Explicit step
4     $\hat{v}^n_A := \hat{v}^n_{\mathrm{Ext}}$
5     $A^{\#} = \mathrm{Id} + \hat{u}^{\#}_A, \quad \Omega^{\#} = A^{\#}(\hat{\Omega})$        ◃ New geometry
6     Find $v^n \in H^1(\Omega^{\#})$ and $p^n \in L_2(\Omega)$ so that:        ◃ Implicit step
7     $\begin{cases} a_v(v^n, \phi) + b(\phi, p^n) = g_v(\phi) & \forall \phi \in H^1_D(\Omega^{\#}), \\ b(v^n, q) = g_p(q) & \forall q \in L^2(\Omega^{\#}). \end{cases}$
8     $v^n_{\mathrm{Ext}} := \mathrm{Ext}(v^n)$        ◃ Extension
9     $\hat{u}^n := \gamma \Delta t \hat{v}^n_{\mathrm{Ext}} - \sum_{i=1}^{k} \alpha_i \hat{u}^{n-k}$        ◃ Recover displacement
```

Predictor-corrector scheme

As we promised while discussing the GCE scheme, we can replace the simple approximation $\hat{v}^{\#}_s = \hat{v}^{n-1}_s$ with a more accurate one. We compute the prediction of the velocity $\hat{v}^{\#}$ by taking v^n from Algorithm 1 and then repeat the process to obtain the corrected velocity v^n. We demonstrate the process in Algorithm 2. Note that this is close to applying two iterations of a fixed-point method to the fully implicit scheme. However, the advection term in the first and second iteration may be defined differently. In particular, there is no need to use semi-implicit advection in the predictor.

Advection The explicit advection offers a symmetric form $a_v(\cdot,\cdot)$ that is preferable for the linear solver (presented in the next chapter) while the semi-implicit one results in better stability of the scheme. Since we intend to amend the velocity anyway in the corrector step, we use the explicit advection in the predictor. We set:

$$\begin{aligned} v^{\circ} &= v^{n,1} - v^n_A \qquad \text{for } k = 1, \\ v^{\circ} &= 2(v^{n,1} - v^n_A) - (v^{n,2} - v^{n,1}_A) \qquad \text{for } k = 2, \end{aligned} \tag{3.44}$$

and

$$\begin{aligned} v^{\star} &= v^{n,1} \qquad \text{for } k = 1, \\ v^{\star} &= 2v^{n,1} - v^{n,2} \qquad \text{for } k = 2. \end{aligned} \tag{3.45}$$

In the corrector step we use the implicit scheme, that is:

$$v^{\circ} = \hat{v}^{\square} - \hat{v}^{\square}_{\text{Ext}} \tag{3.46}$$

and

$$v^{\star} = v^{n}. \tag{3.47}$$

This scheme could be seen as an attempt to solve nonlinear problem arising from the fully implicit scheme by using two fixed-point iterations. For this reason we will refer to it by GCE$_k$(2). The previous scheme – Equation (3.38), i.e. without the corrector is in this convention GCE$_k$(1).

Algorithm 2: Predictor-corrector GCE$_k(F)$ scheme

```
Data: $\hat{u}_A^{n-1}, v^{n-1}, \hat{v}_{\text{Ext}}^{n-1}, \hat{u}^{n-1}$
Result: $\hat{u}_A^{n}, v^{n}, \hat{v}_{\text{Ext}}^{n}, \hat{u}^{n}$
1  begin
2      $\hat{v}^{\square}_{\text{Ext}} := \hat{v}^{n-1}_{\text{Ext}}$
3      for $j = 1$ to $F$ do
4          $\hat{u}^{\#}_A := \gamma \Delta t \hat{v}^{n}_{\text{Ext}} - \sum_{i=1}^{k} \alpha_i \hat{u}^{n-k}$        ◃ Explicit step
5          $\hat{v}^{\square}_A := \hat{v}^{\square}_{\text{Ext}}$
6          $A^{\#} = \text{Id} + \hat{u}^{\#}_A, \quad \Omega^{\#} = A^{\#}(\hat{\Omega})$        ◃ New geometry
7          Find $v^{\square} \in H^1(\Omega^{\#})$ and $p^n \in L_2(\Omega)$        ◃ Implicit step
8          $\begin{cases} a_v(v^{\square}, \phi) + b(\phi, p^n) = g_v(\phi) & \forall \phi \in H^1_D(\Omega^{\#}), \\ b(v^{\square}, q) = g_p(q) & \forall q \in L^2(\Omega^{\#}). \end{cases}$
9          $\hat{v}^{\square}_{\text{Ext}} := \text{Ext}(\hat{v}^{\square})$        ◃ Extension
10     $v^n := v^{\square} \quad \hat{v}^{n}_{\text{Ext}} := \hat{v}^{\square}_{\text{Ext}}$
11     $\hat{u}^n := \gamma \Delta t \hat{v}^{n}_{\text{Ext}} - \sum_{i=1}^{k} \alpha_i \hat{u}^{n-k}$        ◃ Recover displacement
```

3.2 Spatial discretization

Let us now introduce the fully discrete approximation of the fluid-structure interaction problem. We consider triangulation $\mathcal{T}$ of domain $\hat{\Omega}$ with characteristic element size of h. In our case, triangulation $\mathcal{T}$ consists of quadrilateral (2D) or hexahedral (3D) elements. We consider a matching grid, i.e assume that the initial fluid-solid interface does not intersect with any element. With sets of polynomials $\mathcal{P}_p(\tau)$ of order p on each element τ, we define the finite element spaces on triangulation $\mathcal{T}$

$$\begin{aligned} \hat{\mathbb{V}}_h &= \{v \in H^1(\hat{\Omega}) \;:\; v|_\tau \in \mathcal{P}_{p_1}(\tau) \quad \forall \tau \in \mathcal{T}_h\}^d, \\ \hat{\mathbb{Q}}_h &= \{q \in H^1(\hat{\Omega}) \;:\; q|_\tau \in \mathcal{P}_{p_2}(\tau) \quad \forall \tau \in \mathcal{T}_h\}. \end{aligned} \tag{3.48}$$

where p_1 and p_2 are the orders of finite elements for the velocity and pressure, respectively. We first discretize the solid displacement $\hat{u}_s \in \hat{\mathbb{V}}_h$ and the pseudo-displacement $\hat{u}_A \in \hat{\mathbb{V}}_h$ that defines discrete mapping $A^{\#}$. We then define the triangulation of $\Omega^{\#}$ as a transformed triangulation $\mathcal{T}_h$ by mapping $A^{\#}$. Note that triangulation $\mathcal{T}_h$ is a matching triangulation of $\Omega^{\#}$ i.e. the solid-fluid interface does not intersect with any element. The finite element spaces on domain $\Omega^{\#}$ are defined as:

$$\begin{aligned} \mathbb{V}_h &= \{v \circ (A^{\#})^{-1} \;:\; v \in \hat{\mathbb{V}}_h\}^d, \\ \mathbb{Q}_h &= \{q \circ (A^{\#})^{-1} \;:\; q \in \hat{\mathbb{Q}}_h\}. \end{aligned} \tag{3.49}$$

We assume that $\mathbb{V}_h$ and $\mathbb{Q}_h$ satisfy the Ladyzhenskaya–Babuška–Brezzi condition [94]. Will use Taylor-Hood finite element pair, that is $p_1 = 2$ and $p_2 = 1$.

3.2.1 Fully discrete scheme

Let us now define the finite element approximation of the problem solved at each time step – Equation (3.38). For every time step $n > 0$ we need to find $v^n \in \mathbb{V}_h$, $p^n \in \mathbb{Q}_h$, $\hat{u}_s^n \in \hat{\mathbb{V}}_h$, $\hat{u}_A^n \in \hat{\mathbb{V}}_h$ so that

$$\begin{cases} a_v(v,\phi) + b(\phi,p^n) & = g_v(\phi) \quad \forall \phi \in \mathbb{V}_h, \\ b(v,q) & = -\left(\frac{1}{\eta_V}(\det(I + \hat{\nabla}\hat{u}_s^{n-1}), \hat{q}\right)_{\hat{\Omega}_s} \quad \forall q \in \mathbb{Q}_h, \\ \hat{u}_A^{\#} & = \mathrm{Ext}(\hat{u}_s^{\#}), \\ \hat{u}_s^n & = \gamma\Delta t \hat{v}_s^n - \sum\limits_{i=1}^{k} \alpha_i \hat{u}_s^{n-i}, \end{cases} \tag{3.50}$$

with forms $a_v(\cdot,\cdot)$ and $b(\cdot,\cdot)$ defined by Equation (3.39). As we demonstrated in Algorithm 1 the system (3.50) could be solved in several sub-steps. In the next section we will focus on the implicit step that is to find $v^n \in \mathbb{V}_h$, $p^n \in \mathbb{Q}_h$ such that

$$\begin{cases} a_v(v^n,\phi) + b(\phi,p) & = g_v(\phi) \quad \forall \phi \in \mathbb{V}_h, \\ b(v^n,q) & = -\left(\frac{1}{\eta_V}(\det(1 + \hat{\nabla}\hat{u}_s^{n-1}), \hat{q}\right)_{\hat{\Omega}_s} \quad \forall q \in \mathbb{Q}_h. \end{cases} \tag{3.51}$$

This is the most computationally demanding part of both Algorithm 2 and Algorithm 1. We note, that another system of linear equations appears in those algorithms when computing extension of velocity. We will discuss this step when assembling the FSI solver in Section 5.4.

Let us separate bilinear forms from linear one, that is move part of $\delta_k v^n$ to the left-hand side. The continuity equation from (3.51) remains unchanged, while the first one in case of semi-implict advection becomes

$$a_V(v^n,\phi) + b(\phi,p) = g_v(\phi) - (\sum_{i=1}^{k} \alpha_i v^{n,i}, \phi)_{\Omega^{\#}} \quad \forall \phi \in \mathbb{V}_h, \tag{3.52}$$

where

$$a_V(v^n,\phi) = \left(\frac{\rho}{\gamma\Delta t} v^n, \phi\right)_{\Omega^{\#}} + (\rho\nabla v^{\star}\, v^{\circ}, \phi)_{\Omega^{\#}} + (2\mu_f \epsilon(v_f^n) : \epsilon(\phi))_{\Omega_f^{\#}} + \mu_s\gamma\Delta t\, \left(\epsilon(v_s^n), \epsilon(\phi)\right)_{\Omega_s^{\#}}.$$

We also define the right-hand side functions:

$$f_v(\phi) = g_v(\phi) - (\sum_{i=1}^{k} \alpha_i v^{n,i}, \phi)_{\Omega^{\#}}$$

and

$$f_p(\phi) = -\left(\frac{1}{\eta_V}(\det(1 + \hat{\nabla}\hat{u}_s^{n-1}), \hat{q}\right)_{\hat{\Omega}_s}.$$

In the case of explict advection, the term $\left(\frac{\rho}{\gamma\Delta t} v^n, \phi\right)$ also has to be moved to left-hand side and becomes part of the f_v. Finally, the problem (3.51) is

$$\begin{cases} a_V(v^n,\phi) + b(\phi,p) & = f_v(\phi) \quad \forall \phi \in \mathbb{V}_h, \\ b(v^n,q) & = f_p(q) \quad \forall q \in \mathbb{Q}_h. \end{cases} \tag{3.53}$$

Streamline-upwind stabilization

The above problem is actually an advection-diffusion problem with a divergence constraint on the velocity. We define the Pecelet number as

$$\mathrm{Pe} = \frac{\|v^{\circ}\|L}{\mu_f} \tag{3.54}$$

where L is the length scale of the domain. It characterizes the kind of equation we are dealing with, if $\mathrm{Pe} > 1$ the problem is advection-dominated, otherwise the problem is diffusion dominated. For the explicit advection $\mathrm{Pe} = 0$, while using the semi-implicit advection the Pecelet number is proportional to Reynolds number. Having in mind application of the method to flows with moderate and high Reynolds number, we expect the advection dominated problem. A straightforward finite element discretization typically results in oscillatory solutions [95].

This issue could be resolved by introducing additional stabilization to the form $a(\cdot,\cdot)$. Among the many possibilities we choose the streamline upwind/Petrov-Galerkin (SUPG) method [95]:

$$a_{V,s}(v,\phi) = a_V(v,\phi) + (-\varepsilon\Delta v + v^\circ \cdot \nabla v - f_v,\, r v^\circ \cdot \nabla\phi) \tag{3.55}$$

where r is a cell-wise constant stabilization parameter. Unfortunately the stabilization requires computing the second-order derivatives that, in case of matrix-free implementation, is signifficantly more complex than the first-order ones. In particular, this feature has not been yet implemented in the `deal.II` matrix-free framework, that is the base of our implementation. As a replacement we use the stabilized form $a(\cdot,\cdot)$:

$$a_{V,s}(v,\phi) = a_V(v,\phi) + (v^\circ \cdot \nabla v, r v^\circ \cdot \nabla\phi). \tag{3.56}$$

The stabilization parameter inside cell K is [95, 96]:

$$r = \frac{h_\tau}{2\|v^\circ\| p_1} \frac{\coth(\mathrm{Pe}_\tau) - 1}{\mathrm{Pe}_\tau} \tag{3.57}$$

where h_τ is the diameter of cell τ and the Pecelet number Pe_τ computed with respect to the cell size is

$$\mathrm{Pe}_\tau = \|v^\circ\| \frac{h_\tau}{2\mu p_1}. \tag{3.58}$$

In cells where $\mathrm{Pe}_K < 1$ we set $r = 0$ to avoid problems with floating-point arithmetics.

The linear problem and the system matrix

Finally, in numerical computation we will be operating on real vectors. The spaces $\mathbb{V}_h$ and $\mathbb{Q}_h$ could be equivalently defined as linear span of a chosen bases:

$$\mathbb{V}_h = \mathrm{span}\{\phi_1, \ldots, \phi_m\}, \tag{3.59}$$

$$\mathbb{Q}_h = \mathrm{span}\{\xi_1, \ldots, \xi_n\} \tag{3.60}$$

where $m = \dim \mathbb{V}_h(\omega)$ and $n = \dim \mathbb{Q}_h(\omega)$.

The spaces $\mathbb{V}_h$ and $\mathbb{Q}_h$ are isomorphic to real vector spaces

$$\begin{aligned} \mathsf{V}_h &= \mathbb{R}^m \\ \mathsf{Q}_h &= \mathbb{R}^n \end{aligned} \tag{3.61}$$

via isomorphisms $\Upsilon : \mathsf{V}_h \longrightarrow \mathbb{V}_h$ and $\Psi : \mathsf{Q}_h \longrightarrow \mathbb{Q}_h$ that are defined by their action on unit vectors

$$\Upsilon(e_i) = \phi_i, \qquad \Psi(e_i) = q_i. \tag{3.62}$$

Thus, the implicit step corresponds to the linear problem in $\mathbb{R}^{m+n}$. Both operators operators $a_{V,s}(\cdot,\cdot)$ and $b(\cdot,\cdot)$ are linear, thus we define matrices A and B

$$\begin{aligned} \mathsf{u}^T A \mathsf{v} &= a_{V,s}(\Upsilon(\mathsf{u}), \Upsilon(\mathsf{v})) \quad \forall \mathsf{u}, \mathsf{v} \in \mathsf{V_h}, \\ \mathsf{q}^T B \mathsf{u} &= b(\Upsilon(\mathsf{u}), \Psi(\mathsf{q})) \quad \forall \mathsf{u} \in \mathsf{V_h}, \mathsf{q} \in \mathsf{Q}_h, \end{aligned} \tag{3.63}$$

vector f_v

$$\mathsf{u}^T \mathsf{f_v} \mathsf{v} = \left(f_v(\Upsilon(\mathsf{v})), \Upsilon(\mathsf{u})\right) \quad \forall \mathsf{u}, \mathsf{v} \in \mathsf{V_h}, \tag{3.64}$$

and vector f_p

$$\mathsf{p}^T \mathsf{f_p} \mathsf{q} = \left(f_p(\Upsilon(\mathsf{v})), \Psi(\mathsf{u})\right) \qquad \forall \mathsf{p}, \mathsf{v} \in \mathsf{Q_h}. \tag{3.65}$$

In this setting, solving the problem appearing in the implicit step corresponds to solving the linear problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathsf{v} \\ \mathsf{p} \end{bmatrix} = \begin{bmatrix} \mathsf{f_v} \\ \mathsf{f_p} \end{bmatrix}. \tag{3.66}$$

Relevance to variable coefficient generalized Stokes problem

Let us recall Equation (3.53) with explicit advection:

$$\begin{cases} a_V(v^n, \phi) + b(\phi, p) &= f_v(\phi) \qquad \forall \phi \in \mathbb{V}_h, \\ b(q, v) &= f_p(\phi) \qquad \forall q \in \mathbb{Q}_h. \end{cases} \tag{3.67}$$

where

$$a_V(v^n,\phi) = \left(\frac{\rho}{\gamma\Delta t}v^n,\phi\right)_{\Omega^\#} + \left(2\mu_f\,\epsilon(v^n),\epsilon(\phi)\right)_{\Omega_f^\#} + \left(2\gamma\Delta t\mu_s\,\epsilon(v^n),\epsilon(\phi)\right)_{\Omega_s^\#} + \left(\rho\nabla v^\star\ v^\circ,\phi\right)_{\Omega^\#}.$$

If we introduce unified the apparent "viscosity":

$$\tilde{\mu}(x) = \begin{cases} 2\mu_f & x\in\Omega_f^\# \\ 2\gamma\Delta t\mu_s & x\in\Omega_s^\# \end{cases}$$

and the apparent density:

$$\tilde{\rho}(x) = \frac{\rho}{\gamma\Delta t}$$

the system (3.67) becomes

$$\begin{cases} (\tilde{\rho}v^n,\phi)_{\Omega^\#} + \left(\tilde{\mu}\,\epsilon(v^n),\epsilon(\phi)\right)_{\Omega^\#} + \left(\rho\nabla v^n\ v^\circ,\phi\right)_{\Omega^\#} + (p,\nabla\cdot\phi)_{\Omega^\#} & = f_v(\phi) \quad \forall\phi\in\mathbb{V}_h, \\ (q,\nabla\cdot v)_{\Omega^\#} & = f_p(\phi) \qquad \forall q\in\mathbb{Q}_h. \end{cases} \tag{3.68}$$

which is a generalized Stokes problem with piece-wise constant coefficients. Note that $\Delta t\mu_s$ is typically bigger than μ_f by several orders of magnitude. In the next chapter, we propose a new multilevel method based on [83] and [82] that is capable of solving such a problem. We note that in case of semi-implicit advection, the term $(\rho\nabla v^\star\ v^\circ,\phi)_{\Omega^\#}$ together with stabilization would appear on the left-hand side. To cope with that, we will add a minor modification of the preconditioner in Section 5.2.

The multilevel preconditioner 4

As mentioned in Section 3.2.1 the GCE scheme requires the solution of a generalized Stokes problem in every time step of the algorithm. In the case when the advection is treated explicitly, it reads:

$$\begin{cases}(\tilde{\rho}v,\phi)_\Omega + \big(\tilde{\mu}\,\epsilon(v),\epsilon(\phi)\big)_\Omega + (p,\nabla\cdot\phi)_\Omega &= f_v(\phi) \qquad \forall\phi\in\mathbb{V}_h,\\ (q,\nabla\cdot v)_\Omega &= f_p(\phi) \qquad \forall q\in\mathbb{Q}_h.\end{cases} \tag{4.1}$$

For the simplicity of the presentation, we shall assume that the time step is fixed, and we seek (v, p) defined on prescribed $\Omega = \Omega^\#$, where also the corresponding finite element spaces $\mathbb{V}_h$ and $\mathbb{Q}_h$ are defined. We also assume non-distorted mesh, i.e. $A^\# = I$.

This generalized Stokes problem is of interest in its own. Such systems appear not only in fluid-structure interaction, but also in mantle convection problems [97, 98], simulations of two-phase [99] or viscoelastic fluids flows [100]. Since its finite element discretization leads to a large, sparse symmetric system of linear equations with a 2×2 block matrix,

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} V \\ P \end{bmatrix} = \begin{bmatrix} F_v \\ F_p \end{bmatrix} \tag{4.2}$$

with the positive-definite block A, the system has to be solved approximately, usually with a Krylov iterative method. Since the convergence speed of Krylov methods depends on the spectral properties of the matrix, which in our case can be adversely affected by both the huge number of unknowns or high the contrast in $\tilde{\mu}$, an efficient preconditioner is vital for the overall performance.

4.1 State of the art

Preconditioning strategies for the Stokes problem with constant viscosity have been in active development for several decades now. One of the difficulties lies in the divergence-free constraint, which makes the system indefinite. While the Stokes problem turns positive definite in the divergence-free space, standard construction of local divergence-free finite element basis leads to a linear system whose condition number is much worse [101], so most efforts tackle the original system (4.2), of saddle-point type.

One of the most popular ways to improve the convergence rate of Krylov space methods on (4.2) is to use block preconditioning, see e.g. [102–105]. These methods take advantage of the block structure of the matrix and build the preconditioner from smaller, usually symmetric and positive definite ones, which in turn can be solved with well-established methods, such as the domain decomposition [106] or the multigrid [107].

For the Stokes problem with discontinuous velocity with a single interface, Olshanskii and coworkers [108, 109] developed and analyzed a MINRES method with block preconditioners whose performance was independent of the size of the system and robust with respect to the contrast in $\tilde{\mu}$ coefficient. The precondioner for the Schur complement was based on the pressure mass matrix scaled by the inverse of the viscosity. Another approach to the so called *single sinker* problem (where there is one island of high viscosity surrounded by low viscosity fluid) was demonstrated by Aksoylu et. al. [110], who used a strong preconditioner for the velocity part and a preconditioned conjugate gradient solve for the Schur complement.

The multiple inclusion case was investigated in [98, 111], where a hybrid (algebraic/geometric) multigrid approximation for the velocity block and a weighted BFBT preconditioner for the Schur complement was used. The BFBT components were approximated by a two step process, consisting first of a re-discretization of a certain infinite-dimensional differential operator, and then, approximation of its inverse by a V–cycle multigrid. In [111] it was shown that this method performs well on the multi–sinker benchmark problem, in

The ideas presented here have been proposed by the author of this thesis, while Dr Piotr Krzyżanowski has greatly improved large portions of the text, based on a joint paper (in preparation).

the case when discontinuities are replaced by a smooth function with steep gradients. It also indicated that simplified approximation to the Schur complement, such as the weighted pressure mass matrix considered in [112], is not robust when the contrast is high (see also [113]).

Another approach is to develop a multilevel method directly for (4.2). A multigrid method for the Stokes equations (with $\tilde{\mu} \equiv 1$) was proposed and analyzed by Braess and Sarazin [82]. Their smoother makes use of the block structure of the problem, too, and includes a projection step on the discrete divergence–free velocity subspace, requiring that a certain Schur complement problem is solved on each level. Zulehner [83] generalized this method to cover block smoothers with inexact solves, and provided a set of conditions which guarantee the convergence rate is independent of the the number of levels; however only constant $\tilde{\mu}$ was considered. See also [114–117], and [118] where independence of certain equation parameters was investigated. More recently, a class of block smoothers for the Stokes and Lamé problems were considered in [119], mixing both block and multigrid approaches. Finally, in [120], the application of a multilevel preconditioner based on Vanka-type smoother [121] was shown to perform well for the single-sinker problem with smooth, strongly varying viscosity, however its performance degraded for steeper gradients.

When the system matrix is too big to be efficiently stored, or dealt with, it is typically stored in a sparse matrix format. The other approach is to use a matrix-free method that does not need access to matrix elements: rather, it replaces the matrix with a linear operator, whose action on a vector (substituting the usual sparse matrix-vector multiplication) is given as a computer procedure only, offering potential for extensive memory usage optimization. Usually, the data flow in matrix-free methods is regular enough to make it possible to vectorize the code via single-instruction/multiple-data (SIMD) processor instructions. Among papers mentioned above, matrix-free preconditioners for strongly variable Stokes problem were investigated in [112] and [113]. Moreover, Bauer et al. [122] presented a highly scalable matrix-free preconditioner based on Uzawa multigrid, introduced earlier in [123].

In this chapter, we propose and evaluate a matrix-free preconditioner, based on a multilevel scheme for (4.2) under restriction that all blocks in (4.2), except the coarsest level, are provided as matrix-free operators. In this respect our approach is similar to [112] and [113]. In contrast to [111, 113] we examine our method on discontinuous, piecewise constant viscosities, as specified in Section 3.2.1. Our goal is to achieve robustness of the convergence speed with respect to problem size and jumps in the coefficient, while keeping the list of tuning parameters relatively short.

We define our preconditioner as a single iteration of the multigrid method with block constrained smoother as in [83]. This choice again makes an approximate, matrix-free, Schur system solver key to the performance and robustness of the preconditioner. We follow the ideas of Xu and Zhu developed in [62] for diffusion problems with discontinuous coefficient, and apply the preconditioned Conjugate Gradient method (CG) with a relatively simple multigrid preconditioner. Where applicable, we enhance the smoother blocks via the Chebyshev iteration [74] which, while maintaining its matrix-free character is crucial for the overall performance of the preconditioner. We also demonstrate that a simple approximation of Schur complement based on Chebyshev iteration in most cases results in a similar convergence rate.

While the cost of our pressure smoothing block is higher than the corresponding cost of the Schur complement preconditioner in [111], our method remains simpler to implement, building only on standard blocks available in `deal.II` library; in particular, we neither resort to auxiliary operator discretizations [111], nor require other than standard geometric multigrid methods.

It follows that the convergence speed of the FGMRES iteration [124, 125] with our preconditioner is only weakly sensitive to the mesh size, contrast ratio or distribution of the discontinuities of the viscosity. In particular, for structured grids, we observe that the convergence is independent of both problem size and coefficient jump. Moreover, it follows from the experiments that the work and memory per degree of freedom required to solve the linear system are bounded by a small constant.

In Section 4.1.1 we introduce finite element discretization of (4.2) on a family of nested grids. Next, in Section 4.2, we describe the preconditioner in a top-down manner, beginning with the outer multigrid iteration, its block smoother description, and ending with definitions of block solvers utilizing Chebyshev smoothers and inner iterations. Finally, in Section 4.3 we demonstrate desired properties of the method, on both structured and unstructured grids in 2D or 3D, and with various values of $\tilde{\rho}$ and arrangements of discontinuities in $\tilde{\mu}$.

4.1.1 Problem setting

We assume that each subdomain Ω_f, Ω_s is a sum of a finite number of (possibly disjoint) polyhedrons, and that (4.1) is scaled properly so that

$$\tilde{\mu}(x) = \begin{cases} \mu_1 = 1 & \text{for } x \in \Omega_f, \\ \mu_2 \geq 1 & \text{for } x \in \Omega_s. \end{cases} \tag{4.3}$$

In the case of piecewise constant viscosity the contrast in the viscosity between subdomains Ω_s and Ω_f is therefore $\mu_2/\mu_1 = \mu_2 \geq 1$.

For the simplicity of the presentation, we assume $\partial\Omega$ is split into two disjoint parts of positive measure, $\partial\Omega = \Gamma_D \cup \Gamma_N$. On Γ_D we impose homogeneous Dirichlet boundary condition, $u = 0$. On the other part of the boundary, $\Gamma_N = \partial\Omega \setminus \Gamma_D$, we assume Neumann condition $(pI + \frac{1}{2}\tilde{\mu}(\nabla u + \nabla u^T) \cdot n = g$, where n denotes the unit outward normal vector to $\partial\Omega$.

To define a multilevel method, we consider a family of nested conforming quasi-uniform simplicial (or rectangular) triangulations of Ω,

$$\mathcal{T}_0 \subset \mathcal{T}_1 \subset \ldots \subset \mathcal{T}_J$$

such that $\mathcal{T}_j$ is obtained from $\mathcal{T}_{j-1}$ by uniform refinement. (For example, in 2D case, we can split each quadrilateral into four smaller ones.) We will refer to index j as the refinement level. In particular, the mesh parameter of $\mathcal{T}_j$, understood as the maximum diameter of elements in $\mathcal{T}_j$, equals $h_j = 2^{-j}h_0$. We assume that the coarsest mesh correctly resolves the subdomains, that is, no element in $\mathcal{T}_0$ is crossed by the inner interface $\Gamma_I = \partial\Omega_f \cap \partial\Omega_s$.

We approximate (4.1) on the finest mesh, $\mathcal{T}_J$, with the finite element method, using inf-sup stable [101] finite dimensional spaces $\mathbb{V}_J \subset \mathbb{V}$ and $\mathbb{Q}_J \subset \mathbb{Q}$. (One possible choice could be the Taylor–Hood Q_2-Q_1 pair [25] associated with $\mathcal{T}_J$). The discrete problem is then to find $(u_J, p_J) \in \mathbb{X}_J = \mathbb{V}_J \times \mathbb{Q}_J$ such that

$$\begin{aligned} a_V(v_J, \phi_J) + b(\phi_J, p_J) &= \int_\Omega f\ \phi_J\, dx + \int_{\Gamma_N} g\ \phi_J\, ds, \\ b(v_J, q_J) &= 0 \end{aligned} \tag{4.4}$$

for all $(\phi_J, q_J) \in \mathbb{V}_J \times \mathbb{Q}_J$, where $b(v,q) = (q, \nabla \cdot v)_\Omega$. After choosing standard finite element basis in $\mathbb{X}_J = \mathbb{V}_J \times \mathbb{Q}_J$ we finally arrive at a system of linear equations with appropriately prescribed right-hand side vector F_J, for unknown coefficients x_J

$$\mathcal{M}_J x_J = F_J, \tag{4.5}$$

whose matrix $\mathcal{M}_J$ is very large, sparse, symmetric and indefinite, and with a 2×2 block structure corresponding to the splitting of the unknowns to the velocity and the pressure part:

$$\mathcal{M}_J = \begin{bmatrix} A_J & B_J^T \\ B_J & 0 \end{bmatrix}.$$

We will focus on the practically important case when blocks in $\mathcal{M}_J$ are represented as matrix-free linear operators. Since $\mathcal{M}_J$ is ill-conditioned, with respect to both the number of levels J (in other words, the mesh size, or the number of unknowns) [117] and to the contrast in the viscosity coefficient [62]. Therefore, in order to solve (4.5) we apply a preconditioner $\mathcal{P}_J$ to the system,

$$\mathcal{P}_J \mathcal{M}_J x_J = \mathcal{P}_J F_J$$

and solve the resulting left–preconditioned system by the FGMRES iteration [124, 125]. We need the flexible variant of GMRES because inside the preconditioner we will use a few CG iterations that make the whole operator non-linear.

The goal of this chapter is to develop and evaluate a multilevel preconditioner $\mathcal{P}_J$ for (4.5), which is efficient and robust with respect to both the mesh size (i.e., the number of levels J) and the contrast in the viscosity.

Algorithm 3: One step of a generic multigrid V-cycle procedure. Matrix R_j corresponds to standard restriction operator from $\mathbb{X}_j$ to $\mathbb{X}_{j-1}$.

```
Function $y = \mathrm{MGM}(\mathcal{M}_j, F_j, \mathcal{K}_j, m, x, j)$
    if $j = 0$ then
        Solve $\mathcal{M}_0 y = F_0$                              ◃ Direct solve on the coarsest grid $\mathcal{T}_0$
        return $y$
    $x^0 = x$
    for $s = 1$ to $m$ do
        $x^s = x^{s-1} + \mathcal{K}_j(F_j - \mathcal{M}_j x^{s-1})$                  ◃ pre-smoothing
    $r_{j-1} = R_j\left(F_j - \mathcal{M}_j x^m\right)$                  ◃ restriction to the coarser grid
    $e_{j-1} = \mathrm{MGM}(\mathcal{M}_{j-1}, r_{j-1}, \mathcal{K}_{j-1}, m, 0, j-1)$   ◃ coarse correction; recursive call
    $e_j = R^T_{j-1} e_H$                                  ◃ prolongation from the coarser grid
    $y^0 = x^m + e_j$
    for $s = 1$ to $m$ do
        $y^s = y^{s-1} + \mathcal{K}_j(F_j - \mathcal{M}_j y^{s-1})$                  ◃ post-smoothing
    return $y^m$
```

Algorithm 4: n iterations of generic multigrid V-cycle method on level j

```
Data: $F$
Result: $x = \mathrm{MG}_n(M, K, m, j) \cdot F$
$x = 0$
for $i = 1$ to $n$ do
    $x = \mathrm{MGM}(M, F, K, m, x, j)$
return $x$
```

4.2 Multilevel preconditioner

Note that the family of grids $\mathcal{T}_j$, $j = 0, \ldots, J$, actually gives rise to a family of discrete problems (4.4) posed in $\mathbb{X}_j = \mathbb{V}_j \times \mathbb{Q}_j$ (where the finite element spaces $\mathbb{V}_j$ and $\mathbb{Q}_j$ are defined on mesh $\mathcal{T}_j$), with corresponding block matrices

$$\mathcal{M}_j = \begin{bmatrix} A_j & B_j^T \\ B_j & 0 \end{bmatrix}, \tag{4.6}$$

where A_j, B_j are matrix representations of the bilinear forms $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, respectively. For given j, we will refer to such problem as the j-th level problem.

Our preconditioner $\mathcal{P}_J$ will rely on the standard V-cycle multigrid method [82, 83], see Algorithm 3, with suitably chosen smoothers $\mathcal{K}_j$. We define $\mathcal{P}_J$ as the result of n iterations of the multigrid method on the finest level J, starting from a zero initial guess:

$$\mathcal{P}_J = \mathrm{MG}_n(\mathcal{M}_J, \mathcal{K}_J, m, J), \tag{4.7}$$

where the action of the linear operator $\mathrm{MG}_n(M, K, m, j)$ on a vector is formally described in Algorithm 4.

It is well known that the choice of appropriate smoothers $\mathcal{K}_j$ is vital for the overall performance of $\mathcal{P}_J$. On the j-th level, $\mathcal{K}_j$ is a smoother for the system with matrix $\mathcal{M}_j$ with block structure specified in (4.6). In this chapter we make use of block smoothers proposed and analysed by Zulehner [83]:

$$\mathcal{K}_j = \begin{bmatrix} \hat{A}_j & B_j^T \\ B_j & B_j \hat{A}_j^{-1} B_j^T - \hat{S}_j \end{bmatrix}^{-1}. \tag{4.8}$$

Let us mention that to apply $\mathcal{K}_j$ to a vector $\begin{bmatrix} f \\ g \end{bmatrix}$ partitioned according to the block structure of $\mathcal{K}_j$, one has to solve a block system of linear equations. It can easily be verified that this reduces to two solves with $\hat{A}_j$ and

Algorithm 5: Application of $\mathcal{K}_j$ defined in (4.8) to a vector

```
Data: $f, g$
Result: $\begin{bmatrix} u \\ p \end{bmatrix} = \mathcal{K}_j \begin{bmatrix} f \\ g \end{bmatrix}$
1 $u^* = \hat{A}_j^{-1} f$                    ◃ Solve $\hat{A}_j u^* = f$
2 $p = \hat{S}_j^{-1}(u^* - g)$               ◃ Solve $\hat{S}_j p = u^* - g$
3 $u = \hat{A}_j^{-1}(f - B_j^T p)$           ◃ Solve $\hat{A}_j u = f - B_j^T p$
4 return $\begin{bmatrix} u \\ p \end{bmatrix}$
```

one with $\hat{S}_j$, as described in Algorithm 5. It should be stressed that in order to be able to apply $\mathcal{K}_j$, one only requires to be able to apply $\hat{A}_j^{-1}$ and $\hat{S}_j^{-1}$ to vectors: neither $\hat{A}_j^{-1}$ nor $\hat{S}_j^{-1}$ have to be explicitly constructed, so in this sense $\mathcal{K}_j$ is a matrix-free operator, provided $\hat{A}_j^{-1}$ and $\hat{S}_j^{-1}$ are. Moreover, the very matrices $\hat{A}_j$ and $\hat{S}_j$ are never used in the algorithm and never have to be formed.

The smoother $\mathcal{K}_j$ is specified by the choice of two symmetric positive definite operators, $\hat{A}_j$ and $\hat{S}_j$. One may think about them as (sometimes crude) approximations to the velocity block A_j and its Schur complement,

$$S_j = B_j \hat{A}_j^{-1} B_j^T, \tag{4.9}$$

respectively. In particular, taking $\hat{A}_j = \operatorname{diag}(A_j)$ and $\hat{S}_j = S_j$, we recover the smoother developed by Braess and Sarazin [82]. This choice, however, is computationally quite demanding, so already in [82] the authors recommend to solve systems with S_j only approximately — in other words, to replace S_j^{-1} with a cheaper and more efficient $\hat{S}_j^{-1}$.

The analysis of smoothers (4.8) in [83] assumes that of matrices $\hat{A}_j, \hat{S}_j$, only one has to be a "good" preconditioner. Since Algorithm 5 requires two solves with $\hat{A}_j$, whose dimension is larger than that of $\hat{S}_j$, it is convenient to choose $\hat{A}_j$ as simple as possible. In such a case, according to [83], it is sufficient that

$$\hat{A}_j > A_j \tag{4.10}$$

and

$$\hat{S}_j \leq B_j \hat{A}_j^{-1} B_j^T \leq \frac{4}{3} \hat{S}_j, \tag{4.11}$$

where for symmetric matrices X, Y of equal size, by $X \leq Y$ we mean there holds $x^T X x \leq x^T Y x$ for all nonzero vectors x. In order to satisfy assumption (4.11), one should carefully scale $\hat{S}_j$ prior to plugging it into $\mathcal{K}_j$. Let us remark though, that numerical experiments indicate that in certain cases such scaling may be not necessary; see [83, Remark 5] or Section 4.3.1 below.

In the remaining part of this section we complete the definition of the preconditioner, providing a detailed description of $\hat{A}^{-1}$ and $\hat{S}^{-1}$ or, rather, in accordance with the matrix-free framework, procedures for their application to a vector.

4.2.1 Smoother building blocks

For clarity, in this subsection we suppress the level index j, assuming it is fixed. The construction of both $\hat{A}^{-1}$ and $\hat{S}^{-1}$ will be founded on Chebyshev smoothers, which we briefly describe below.

Chebyshev smoothers

In the matrix-free context, one should restrict only to smoothers which can be set up without direct access to matrix entries, with the obvious exception of matrix diagonals, which are easy to compute. This is done essentially by looping over all cells and discarding off-diagonal entries. This condition excludes Gauss-Seidel and other smoothers based on triangular matrix splittings, and promotes the use of Jacobi-type smoothers which only need access to matrix diagonal. On the other hand, since diagonal smoothers are less efficient than their triangular splitting relatives, a common practice to improve the properties of the former is to supplement them with several iterations of the Chebyshev method [126]. Thus, following [74], we define the Chebyshev smoother of order k for a *generic* linear system $Mx = F$ as

$$\mathrm{Cheb}(M, D, k) = P_k(D^{-1}M)D^{-1}, \tag{4.12}$$

where P_k is a prescribed polynomial of degee k and D is a cheap preconditioner for M (a usual choice is to set $D = \mathrm{diag}(M)$). In particular, the 0-th order Chebyshev smoother with diagonal preconditioner simplifies to the (damped) Jacobi method. The polynomial P_k is chosen to obtain smoothing in some desired range $[a, b]$, by utilizing the Chebyshev polynomial of the first kind T_k:

$$P_k(X) = \frac{T_k\left(\frac{2X-(a+b)I}{b-a}\right)}{T_k\left(\frac{2-(a+b)}{b-a}\right)}. \tag{4.13}$$

To act as a smoother, $\mathrm{Cheb}(M, D, k)$ should efficiently remove high frequency components of the error which in our case are associated with the largest eigenvalues of the preconditioned matrix $D^{-1}M$. Hence, the smoothing range $[a, b]$ is set to $[\frac{s_1}{s_2}\lambda_M, s_1\lambda_M]$, where $\lambda_M = \lambda_{\max}(D^{-1}M)$ is the largest eigenvalue of $D^{-1}M$, with user selected parameters s_1 and $s_2 > s_1$. In practice it suffices to compute an approximate value of $\lambda_{\max}(D^{-1}M)$, e.g. by running a couple of CG iterations [72, 74], an approach which allows Chebyshev operators to self-adjust to the choice of D. In order not to proliferate the parameters, we exclude s_1, s_2 and the number n_{CG} of CG iterations used to approximate λ_M from the formal parameters list in (4.12), treating them rather as free constants, to be specified when necessary.

Construction of $\hat{A}^{-1}$

We define $\hat{A}^{-1}$ as the Chebyshev smoothing operator (4.12) with diagonal preconditioner:

$$\hat{A}^{-1} = \mathrm{Cheb}(A, \mathrm{diag}(A), k_A). \tag{4.14}$$

By tuning the order of the Chebyshev operator k_A, the user can balance its cost and smoothing properties.

Construction of $\hat{S}^{-1}$

The design of $\hat{S}^{-1}$ is more complex and turns out crucial for the final efficiency of the smoother, since the geometric multigrid method alone may be inefficient for a high contrast coefficient (elliptic) problem [62]. As already discussed, direct computation of the Schur complement matrix S defined in (4.9) — not to mention its inverse — is usually prohibitively expensive. Because condition (4.11) assumes $\hat{S}$ is a good enough preconditioner for S, in the construction of this smoothing block we will exploit the self-adjusting property of Chebyshev smoothers and thus avoid direct reference to matrix entries to stay within the matrix-free framework.

We begin with a diagonally preconditioned Chebyshev smoother for $S = B\hat{A}^{-1}B^T$,

$$\tilde{S}_{k_S}^{-1} = \mathrm{Cheb}(S, \mathrm{diag}\left(B(\mathrm{diag}\,A)^{-1}B^T\right), k_S), \tag{4.15}$$

which will serve as a basis for the later definition of $\hat{S}^{-1}$. This operator aims at providing effective smoothing while maintaining flexibility in the choice of $\hat{A}$, with no need for direct computation of all entries of matrix S. The rationale for the above choice is as follows: If a 0-th order Chebyshev smoother was to be used as $\hat{A}^{-1}$, the diagonal of $B(\mathrm{diag}A)^{-1}B^T$ would be cheap to compute, even if both A and B were in a matrix-free format. One can, therefore, expect that for a higher order smoother the diagonal of $B(\mathrm{diag}A)^{-1}B^T$ will still provide a good enough approximation to $\mathrm{diag}(S)$.

Then, from the founding preconditioner $\mathrm{MGCheb}(n_S, k_S, m_S)$ defined as

$$\mathrm{MGCheb}(n_S, k_S, m_S) = \mathrm{MG}_{n_S}(S, \tilde{S}_{k_S}^{-1}, m_S, j), \tag{4.16}$$

(cf. Algorithm 4), we propose three possible definitions of $\hat{S}^{-1}$:

1. $\hat{S}^{-1} = \mathrm{MGCheb}(n_S, k_S, m_S)$, or

2. $\hat{S}^{-1} = \mathrm{MGCG}(n_S, k_S, m_S)$, or
3. $\hat{S}^{-1} = \mathrm{ChebCG}(n_S, k_S)$,

Here, for a given vector g we define $\mathrm{MGCG}(n_S, k_S, m_S) \cdot g$, $\mathrm{ChebCG}(n_S, k_S) \cdot g$ as the result of n_S iterations of the preconditioned CG method applied to the system $Sp = g$ with the following selection of the preconditioner:

- ▶ $\mathrm{MGCheb}(1, k_S, m_S)$, for $\mathrm{MGCG}(n_S, k_S, m_S)$;
- ▶ $\tilde{S}^{-1}_{k_s}$, for $\mathrm{ChebCG}(n_S, k_S)$,

and zero initial guess. Note that the two last choices of $\hat{S}^{-1}$ are not fixed linear operators, thus, as we mentioned the use of flexible GMRES method is necessary.

We note that for the constant coefficient case the theory [83] only covers the first choice. The second one could be considered as intuitive extension for the variable coefficient case, while for the third one we have only numerical evidence.

4.2.2 Summary of user specified parameters

The preconditioner $\mathcal{P}_J$ is finally selected by the choice of nine parameters:

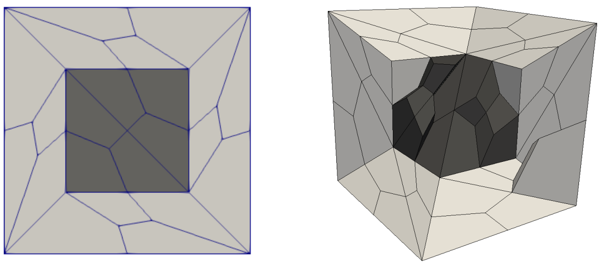
- ▶ the number n of outer iterations and the number m of smoothing steps in (4.7),
- ▶ the order k_A of the Chebyshev smoother defining $\hat{A}^{-1}$ in (4.14),
- ▶ the number n_S of inner iterations and the number m_S of smoothing steps in (4.16),
- ▶ the order k_S of the Chebyshev smoother defining $\tilde{S}^{-1}_{k_S}$ in (4.15),
- ▶ smoothing range parameters s_1, s_2 and the number n_{CG} of CG iterations used to compute $\lambda_{\max}$ for Chebyshev smoothers, see Section 4.2.1. While in principle one can choose different values for each of the two Chebyshev smoothers we employ, we decided to have a common set of these parameters for both.

4.3 Numerical results

In this section, we investigate properties of the preconditioner designed in Section 4.2. All experiments were implemented in the finite element library `deal.II` version `9.2.0` [73] using its matrix-free toolbox [72].

Our model problem is (4.1) on a unit square in 2D or unit cube in 3D, with the homogeneous Dirichlet condition on the top face and zero Neumann condition elsewhere. We choose this a bit artificial setting, to test the solver in settings that could be easily replicable. More realistic cases will be examined in Chapter 6.

Except Section 4.3.1, where continuous viscosity case, $\mu_2 = \mu_1 = 1$ is briefly treated, for the comparison with baseline solvers; other experiments will consider high contrast test cases only, i.e. $\mu_2/\mu_1 \gg 1$, see (4.3). If



(a) 2D **(b)** 3D

Figure 4.1: Unstructured coarse grids $\mathcal{T}_0$ used for single inclusion problem. Light gray color marks regions where the viscosity $\tilde{\mu} = \mu_1 = 1$. Region where $\tilde{\mu} = \mu_2$ is marked with a darker shade.

Table 4.1: Number of degrees of freedom on 2D grids for varying number of levels J.

J	1	5	6	7	8	9	10
Structured	387	83,907	333,699	1,330,947	5,316,099	21,249,027	84,965,379
Unstructured	221	51011	203,395	812,291	3,246,595	12,981251	51,914,755

Table 4.2: Number of degrees of freedom, N, on structured grid used for the checkerboard problem in 2D, for varying number of levels J.

J	1	2	3	4	5	6	7	8
N	2,467	$9,539$	$37,507$	$148,739$	$592,387$	$2,364,419$	$9,447,427$	$37,769,219$

not specified otherwise, we then set $\mu_2 = 10^6$. We treat two types of high contrast test problems: with single inclusion located centrally in Ω, cf. Figure 4.1, and with multiple inclusions with checkerboard distribution, see Figure 4.2.

Since the case $\tilde{\rho} = 0$ turns out to be most computationally demanding, cf. Tables 4.7 and 4.10, in the experiments we preset $\tilde{\rho} = 0$, unless explicitly specified otherwise.

We consider both structured Cartesian and unstructured grids obtained from uniform refinement of the coarsest grid $\mathcal{T}_0$, which is always aligned with discontinuities of the viscosity. Examples of unstructured coarsest grids in 2D and 3D are shown in Figure 4.1. For the checkerboard problem, we restrict ourselves to Cartesian grids, as in Figure 4.2.

The discrete system (4.5) comes from standard Taylor–Hood Q_2-Q_1 element [25] on the finest grid $\mathcal{T}_J$. Tables 4.1, 4.2 and 4.3 list the sizes of the discrete problems for various types of meshes and discretization levels J. In particular, the first column ($J = 1$) corresponds to the size of the coarsest problem, posed on $\mathcal{T}_0$.

In Section 4.3.1 we investigate the convergence speed of the FGMRES method preconditioned with $\mathcal{P}_J$ defined in Section 4.2, while in Section 4.3.2 we discuss its performance. Throughout the experiments we fix $n = 1$, $m = 2$, $m_S = 2$ (cf. Section 4.2.2 for their description), focusing on the influence of k_A, k_S and n_S — i.e. the parameters defining the smoothing block operators — on the performance of the method. The cost of $\mathcal{P}_J$ increases with the growth of each of these parameters, therefore it is important to choose them as small as possible. We also fix $s_1 = 1.2$, $s_2 = 15$, $n_{CG} = 10$, which are the default settings in `deal.II` [72, 73].

If not specified otherwise, in our experiments we set

$$\hat{S}^{-1} = \mathrm{MGCG}(n_S, k_S, m_S),$$

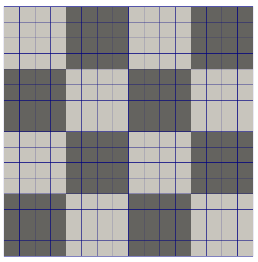
because in our preliminary tests it proved more efficient than MGCheb(n_S, k_S, m_S). While ChebCG in many cases offers even better performance, sometimes its convergence is slower than MGCheb or may require selection of specific parameters. If not prescribed explicitly, we set $n_S = 1$ in ChebCG(n_S, k_S).

4.3.1 Convergence

We measure the convergence speed of the method by reporting the number of iterations required to reduce the Euclidean norm of the residual by a factor of 10^{-8}. The right hand side function is $f(x) = \big(-\cos(x)\cos(y), -\sin(x)\sin(y)\big)$ in 2D and $f(x) = \big(-\cos(x)\cos(y), -\sin(x)\sin(y), 0\big)$ in 3D. We always start from a zero initial guess. We declare the method fails to converge and mark this in the tables with a dash if the number of iterations exceeds 100.

Table 4.3: Number of degrees of freedom on 3D grids, for varying number of levels J.

J	1	2	3	4	5	6
Structured	2,312	15,468	112,724	859,812	6,714,692	53,070,468
Unstructured	4,606	34,876	273,220	2,166,724	17,265,796	137,870,596


(a) 4 × 4 checkerboard

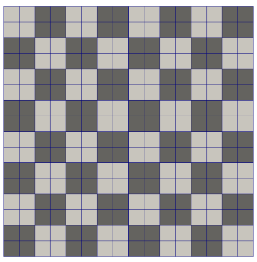

(b) 8 × 8 checkerboard

Figure 4.2: Coarse grids $\mathcal{T}_0$ used for multiple inclusion problems in 2D. Light gray color marks regions where the viscosity $\tilde{\mu} = \mu_1 = 1$. Regions where $\tilde{\mu} = \mu_2$ are marked with a darker shade.

Table 4.4: Dependence of the number of iterations on the choice of preconditioner parameters k_S, n_S, k_A in the constant viscosity case $\mu_2 = \mu_1 = 1$ and with $\tilde{\rho} = 0$, in 2D. The bottom line corresponds to the smoother by Braess and Sarazin. Structured grid with $J = 9$ levels.

k_S	n_S	k_A			
		1	2	3	4
1	1	8	7	6	19
1	2	8	6	5	5
2	1	8	6	5	5
4	1	8	6	5	5
S^{BS}		8	6	5	5

In order to compare our approach to the smoother analyzed by Braess and Sarazin [82], in this section we additionally include results for the case when $\hat{S}_j = S_j = B_j^T \hat{A}^{-1} B_j$. Indeed, if $k_A = 0$, so $\hat{A} = \text{diag}(A)$, this choice corresponds to the base setting from [82], with slightly different relaxation. We refer to this choice as S^{BS}. Let us mention that S^{BS} does *not* lead to a matrix-free method, because S_j have to be constructed and then solved on each level $j = 0, \dots, J$. So, when dealing with S^{BS} in our testing framework, which is confined to matrix-free tools, we solve systems with S_j by (obviously very inefficient in terms of timings) the CG method iterated until the error is reduced by a factor of 10^{-8}, in order to simulate the effect of applying a direct solver.

Constant viscosity case

Setting $\nu_2 = 1$ we obtain classical Stokes problem, with constant viscosity over entire domain Ω. Then, for both structured and unstructured grids in 2D with fixed $J = 9$, we investigate the influence of k_A, k_S and n_S on the convergence speed. The results are presented in Tables 4.4 and 4.5. As expected, the preconditioner behaves much better on the structured mesh, driven by the superior performance of the geometric multigrid which appears in several places inside $\mathcal{P}_J$.

Table 4.4 reveals the preconditioner works very well on structured grids even when its parameters k_A, k_S, n_S are very small. It is also worth noticing that for other choices of k_A, k_S, n_S we get essentially the same performance as with "exactly" solved smoother by Braess and Sarazin. It is also interesting that, starting from a quite low threshold, increasing the order of Chebyshev smoothers does not lead to improvement in the convergence speed.

This picture looks a bit different for the unstructured grid, see Table 4.5. In general, the number of iterations is larger than for the structured grid (even though in the latter, the number of degrees of freedom is almost twice as large, cf. Table 4.1), a phenomenon already observed in [83]. On the unstructured mesh, $k_A = 1$ produces $\hat{A}_j^{-1}$ which is too weak to make the preconditioned iteration convergent, with the notable exception of the Braess and Sarazin smoother. It turns out that in order to match the number of iterations of S^{BS} for

Table 4.5: Dependence of the number of iterations on the choice of preconditioner parameters k_S, n_S, k_A in the constant viscosity case $\mu_2 = \mu_1 = 1$ and with $\tilde{\rho} = 0$, in 2D. The bottom row corresponds to the smoother by Braess and Sarazin. Unstructured grid with $J = 9$ levels.

k_S	n_S	k_A			
		1	2	3	4
1	1	—	—	27	16
1	2	—	19	14	12
2	1	—	20	14	12
4	1	43	18	13	12
S^{BS}		33	18	13	12

Table 4.6: Dependence of the number of iterations on the number of levels J both in the constant viscosity ($\mu_2 = 1$) and the high contrast ($\mu_2 = 10^6$) case on structured and unstructured meshes in 2D, with $\tilde{\rho} = 0$, $k_A = k_S = 2$, $n_S = 1$.

	μ_2	J					
		5	6	7	8	9	10
Structured	1	6	6	6	6	6	6
	10^6	10	10	11	11	11	11
Unstructured	1	16	17	19	21	23	24
	10^6	19	20	22	23	26	26

fixed $k_A > 1$, one has to prescribe sufficiently large k_S, n_S. Moreover, the larger k_A, the smaller k_S, n_S are sufficient to obtain good convergence rate.

In the next series of experiments, we keep the viscosity continuous and constant, and fix some parameters of the preconditioner: $k_A = k_S = 2$, $n_S = 1$ (justified by results above; see also Figure 4.9). In such conditions, we test the dependence of the convergence speed on the number of levels J and nonzero density $\tilde{\rho}$. As previously, we consider both structured and unstructured grids in 2D. We present the numbers of iterations in Tables 4.6 and 4.7. For the structured grid the number of iterations is clearly independent of mesh size, but for the unstructured one the number of iterations is larger and slowly growing with the number of levels, similarly as observed in [83]. The corresponding results for the structured grid in 3D are presented in Table 4.8 and Table 4.10.

Discontinuous viscosity problem with single inclusion

In this, and the next subsection, we investigate the case when $\mu_2 > \mu_1$, so the viscosity is discontinuous. Following the results of the previous section we fix $\rho = 0$ throughout this and the following subsections, as this results in the largest number of iterations.

Here, we focus on the single inclusion problem, when Ω_2 is simply a box centrally located inside Ω, as depicted in Figure 4.1.

First, let us investigate the influence of k_A, k_S and n_S on the convergence speed in 2D, this time only for the more demanding unstructured grid case, with fixed $J = 9$. We report the iteration count in Table 4.9. The overall pattern of the behaviour is quite similar to this obtained in the constant coefficient case. The number of iterations is slightly larger than that in Table 4.5, because of the presence of the inclusion. Similar conclusions can be drawn when inspecting Table 4.6. Quite surprisingly, the method behaves even better in 3D, see Table 4.10, with the number of iterations essentially independent of J and somewhat smaller than in 2D. On the other hand, for the high contrast problem in 3D, the performance degrades roughly twice as compared to the constant viscosity case. In the case of ChebCG and unstructured mesh we had to use higher order smoother, because in case of $k_A = k_s = 2$ the method failed to converge in 100 iterations.

It is also enlightening to track the convergence history for several choices of k_A and k_S, see Figure 4.3. For small values of k_A or k_S, the convergence is not only slower, but it also is delayed during the first two or three

Table 4.7: Dependence of the number of iterations on the number of levels J both in the low density ($\rho = 1$) and high density ($\rho = 10^8$) case with fixed viscosity $\mu_2 = 10^6$ on structured and unstructured meshes in 2D, with $k_A = k_S = 2, n_S = 1$. Upper part: MGCG(n_S, k_S, m_S); lower part: ChebCG(n_S, k_S).

	k_A	k_S	ρ	J					
				5	6	7	8	9	10
Structured	2	2	1	11	11	11	11	11	11
			10^8	6	6	7	7	7	7
Unstructured	2	2	1	19	19	20	20	20	21
			10^8	12	11	11	11	11	11
Structured	2	2	0	12	12	13	13	14	15
			10^6	11	11	11	11	12	11
Unstructured	4	4	0	14	17	17	19	21	19
			10^6	11	12	12	10	10	10

Table 4.8: Dependence of the number of iterations on the number of levels J both in the low viscosity ($\mu_2 = 1$) and high contrast ($\mu_2 = 10^6$) case on structured mesh in 3D, with $\rho = 0$, $k_A = k_S = 2$, $n_S = 1$.

	μ_2	J				
		2	3	4	5	6
Structured	1	5	5	5	5	5
	10^8	5	5	5	5	13

iterations, when the residual essentially stagnates. As we shall see later, this stagnation phase gets prolonged with the increase of the number of inclusions. This seems to correspond well with the findings of [62].

Finally, let us explore the dependence of the number of iterations on the contrast in the viscosity for a wider range of μ_2. In this test, we consider the structured grid in 2D with $J = 9$, and in 3D, with $J = 4$. We keep $k_A = k_S = 2$ as before, but vary n_S, see Table 4.11. The iteration count shows interesting behavior. It grows very modestly for a range of μ_2, starting from $\mu_2 = 1$. However, for certain, very large, values of μ_2, the method breaks down. The magnitude of μ_2 above which the breakdown occurs, clearly depends on n_S. It seems that for those huge μ_2 the system we solve when applying $\hat{S}_j^{-1}$ is so ill–conditioned, that the CG iteration used inside struggles.

We tested the solver in 3D using unstructured grid with number of levels $J = 2, \dots, 5$. The number of iterations grows with problem size, see Table 4.12. For small J, $k_A = 2$ is better in terms of the overall cost (similar number of iterations while each iteration is cheaper), for $J > 4$ $k_A = 4$ dramatically outperforms $k_S = 2$. It turns out that, in comparison to MGCG, ChebCG-based smoother requires higher values of k_A and k_S to obtain similar convergence rate.

Influence of the scaling of $\hat{S}^{-1}$ on the convergence rate As mentioned before, in the constant viscosity coefficient case Zulehner already noticed [83] that the multigrid method works well even if assumption (4.11) is not satisfied. Our experiments presented so far also show that our preconditioner $\hat{S}_j^{-1}$ works well "as is", without any prescaling.

It is nevertheless interesting to investigate how the scaling of $\hat{S}_j^{-1}$ influences the convergence rate of our method, eventually relating Zulehner's observation to the high contrast case (we fix here $\mu_2 = 10^6$). Let us define the average convergence rate as

$$R = \left(\frac{||r_n||_2}{||r_0||_2} \right)^{\frac{1}{n}} \tag{4.17}$$

where r_i is the residual after i-th iteration and n is number of iterations required to meet the stopping criterion. For the unstructured mesh in 2D with $J = 9$ and with fixed $k_A = k_S = 2$ and $n_S \in \{1, 2\}$ we compute R for the iteration with $\hat{S}_j^{-1}$ replaced with $\beta \hat{S}_j^{-1}$ where $\beta \in [0.8, 6.25]$. Of course, the original preconditioner is recovered for $\beta = 1$.

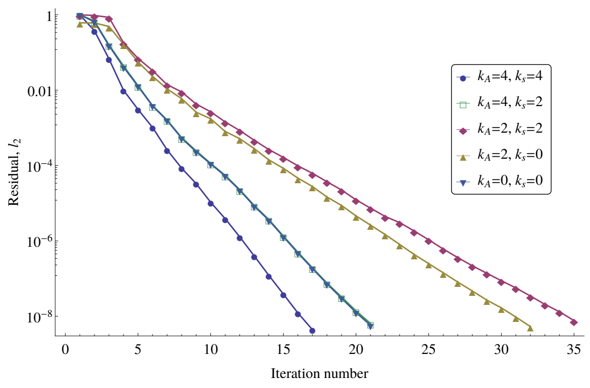


Figure 4.3: Convergence history for various choices of smoother parameters.

Table 4.9: Dependence of the number of iterations on the choice of preconditioner parameters k_S, n_S, k_A in the 2D high contrast ($\mu_2 = 10^6$, $\rho = 0$) case. The bottom row in the left table corresponds to the smoother by Braess and Sarazin. Unstructured grid with $J = 9$ levels.

(a) Unstructured, MGCG

k_S	n_S	k_A			
		1	2	3	4
1	1	—	—	23	17
1	2	—	21	17	14
2	1	—	21	17	14
4	1	41	21	17	14
S^{BS}		25	19	16	14

(b) Structured, ChebCG

k_S	n_S	k_A			
		1	2	3	4
1	1	—	14	12	10
1	2	29	13	11	10
2	1	25	14	12	12
4	1	—	12	10	9

(c) Unstructured, ChebCG

k_S	n_S	k_A			
		1	2	3	4
1	1	—	—	—	50
2	2	—	—	23	16
2	1	—	—	69	19
4	1	—	—	25	21

Table 4.10: Dependence of the number of iterations on the number of levels J both in the constant viscosity ($\mu_2 = 1$) and high contrast ($\mu_2 = 10^6$) case with various densities on structured mesh in 3D, with $k_A = k_S = 2$, $n_S = 1$.

	μ_2	ρ	J				
			2	3	4	5	6
Structured	1	0	5	5	5	5	5
	10^6	0	7	8	9	9	9
	10^6	1	5	5	5	5	5
	10^6	10^8	5	5	5	5	5

Table 4.11: Dependence of the number of iterations on coefficient contrasts on single inclusion problem. Structured grid with $J = 9$ levels (in 2D) and $J = 4$ levels (in 3D), with $k_A = k_S = 2$, $\rho = 0$.

	n_S	μ_2								
		10^0	10^1	10^2	10^4	10^6	10^8	10^{10}	10^{12}	10^{14}
2D	1	6	8	11	11	11	83	—	—	—
	2	6	8	11	11	11	11	13	84	—
	3	6	8	11	11	11	11	11	11	—
3D	1	5	7	9	9	11	—	—	—	—
	2	5	7	9	9	9	9	15	—	—
	3	5	7	9	9	9	9	9	9	11

Table 4.12: Dependence of the number of iterations on the choice of preconditioner parameters k_S, k_A with fixed $n_S = 1$ for the 3D single inclusion problem on unstructured grid.

(a) MGCG

k_A	k_S	J			
		2	3	4	5
2	2	23	29	36	—
	4	23	28	32	55
4	2	14	17	19	21
	4	14	17	19	20

(b) ChebCG

k_A	k_S	J			
		2	3	4	5
4	2	15	20	34	—
	4	14	16	19	40
6	2	12	15	19	49
	4	12	13	14	16

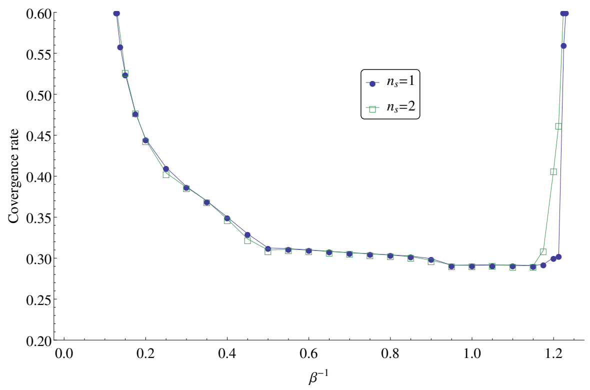


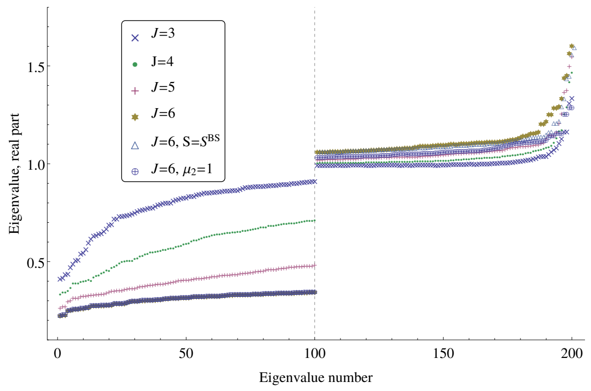
Figure 4.4: Average convergence rate of the preconditioned FGMRES, as a function of β^{-1}. Single inclusion problem with $\mu_2 = 10^6$ on unstructured grid, $J = 9$.

The experiment shows that for $\beta^{-1} \in (0.5, 1.17)$ the convergence rate does not degrade by more than 10% as compared to its minimal value (approximately equal to 0.3), indicating that the method remains effective for a range of scalings, cf. Figure 4.4. Moreover, choosing the smoother without scaling ($\beta = 1$) seems to result in a nearly optimal convergence rate.

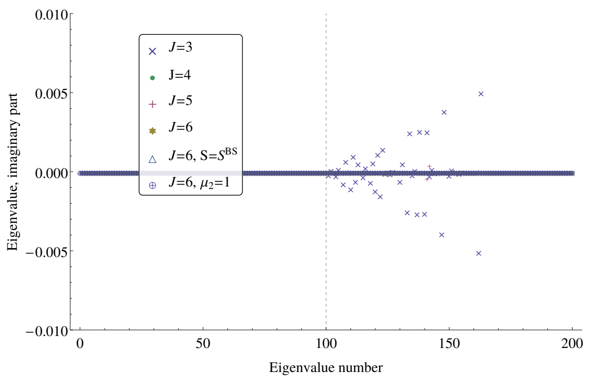
Spectrum analysis Finally, we examine the clustering of the eigenvalues of the preconditioned matrix $\mathcal{P}_J\mathcal{M}_J$, which is known to influence the convergence speed of Krylov space methods. Since, in turn, our preconditioner relies on the quality of $\hat{S}_J^{-1}$ which in principle acts as a preconditioner to S_J, see Table 4.9, we also inspect the eigenvalues of $\hat{S}_J^{-1}S_J$. Because the operator MGCG(n_S, k_S, m_S) is not linear, for the spectral analysis we switch to $\hat{S}_J^{-1}$ = MGCheb(n_S, k_S, m_S). In order to assess the clustering of the spectrum, we investigate both ends of the spectrum, i.e. the eigenvalues largest and smallest by magnitude. To make the experiment computationally feasible, we restrict ourselves to the 2D problem on unstructured grid with $J = 3, \ldots, 6$ with $\mu_1 = 1$ and $\mu_2 = 10^6$. We fix the usual set of preconditioner parameters: $k_A = k_S = 2$ and $n_S = 1$, and compute 100 smallest and 100 largest eigenvalues using ARPACK [127] with tolerance set to 10^{-6} and Arnoldi space size equal to 1000.

Figure 4.5 shows extreme eigenvalues of $\mathcal{P}_J\mathcal{M}_J$. It turns out that for $J = 3$ a few of the computed eigenvalues have small imaginary part (Figure 4.5b); otherwise, they are real. Significant differences between eigenvalues of the preconditioned system for the constant viscosity case and the discontinuous coefficient case are visible in the rightmost part of spectrum (Figure 4.5a), where several eigenvalues are larger for the variable coefficient case than for the constant coeficient. Interestingly, the use of exact Schur complement changes the structure of this end of the spectrum, but the largest eigenvalue remains nearly unchanged. This seems to explain the increase in the convergence rate after the first 2–3 iterations, cf. Figure 4.3, when the corresponding components of the residual have been eliminated.

Figure 4.6 shows the distribution of the eigenvalues of $\hat{S}_J^{-1}S_J$ which cluster around the unity with no outlying eigenvalues. When the problem size increases, the spread of the eigenvalues grows, which may explain slight deterioration in the efficiency of $\hat{S}_J^{-1}$ as reported in Table 4.6.



(a) Real part



(b) Imaginary part

Figure 4.5: Eigenvalues of $P_J \mathcal{M}_J$ for single inclusion problem on unstructured grid with $J = 3, \ldots, 6$ levels. The first 100 eigenvalues (numbered 1-100) and last 100 eigenvalues (numbered 101-200).

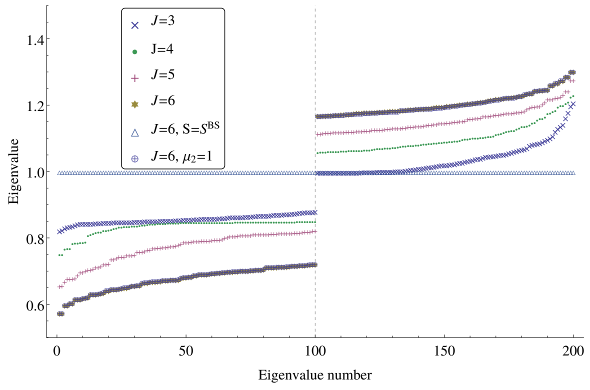


Figure 4.6: Eigenvalues of $\hat{S}^{-1}S$ for single inclusion problem on unstructured grid with $J = 3, \ldots, 6$ levels. The first 100 eigenvalues (numbered 1-100) and last 100 eigenvalues (numbered 101-200), see text.

Discontinuous viscosity problem with multiple inclusions in 2D

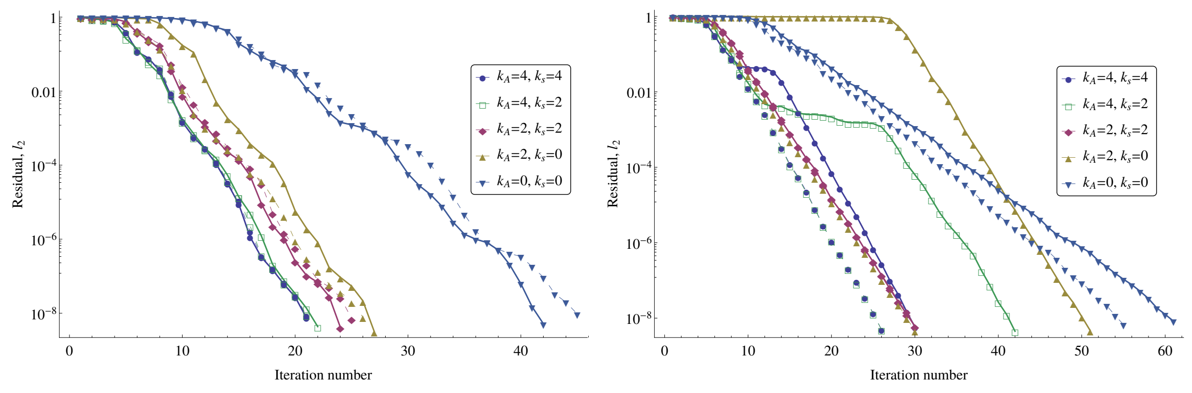
Next, we investigate if the number of high contrast inclusions influences the convergence rate of the methodLet us first consider two 2D test problems with checkerboard distribution of μ_2: 4×4 and 8×8. Both problems are then discretized on a grid $\mathcal{T}_J$ derived from the uniform refinement of the same coarsest level Cartesian grid $\mathcal{T}_0$, see Figure 4.2 (note that this grid is finer than the grid induced by discontinuities in the viscosity). In Table 4.13 we compare the number of iterations for both problems with $J = 2, \ldots, 8$ and several choices of parameters k_A, k_S and n_S. It follows that the number of iterations slowly grows with both problem size and the number of inclusions. For fine enough grids, the increase in the number of iterations is greater when there are more inclusions in the domain. On the other hand, it also follows from Table 4.13 that a partial remedy is to employ higher order smoothers (that is, to increase k_A, k_S) and do more internal CG iterations n_S which, however, increases the overall cost of the iteration. As presented in Table 4.13 in the case of ChebCG, the number of iterations grows with the number of levels for both 4×4 and 8×8 checkerboard problems. Note that despite using 3 CG iterations in ChebCG, we ended up with a higher number of iterations than for MGCG.

It follows from Figure 4.7 that the reason for increased iteration counts is the initial plateau in the convergence history, which was not present in constant viscosity case. The length of the plateaus increases with the number of inclusions which can clearly be seen in both Figure 4.3 (when it was very short for the single inclusion case) and in Figure 4.7, when more, and longer plateaus are observed in the 8×8 case.

We illustrate the quality of the Schur complement approximation by solving the linear equation $Sx = y$ with MGCheb(n_s, k_S, m_s), $n_s = 1$ and $m_s = 1$ used as a preconditioner. As the right hand side vector y we use a vector filled with 1. We report the number of PCG iterations required for reducing residual by a factor of

Table 4.13: Dependence of the number of iterations on the number of levels J in 2D checkerboard problem with various k_A,k_S and n_S. Upper part: MGCG(n_S, k_S, m_S); lower part: ChebCG(n_S, k_S).

	n_S	k_A	k_S	J						
				2	3	4	5	6	7	8
4×4	1	2	2	16	19	21	22	26	29	32
		4	4	18	18	24	25	23	26	27
	2	2	2	13	15	18	21	23	25	26
		4	4	11	13	16	18	20	22	23
8×8	1	2	2	12	15	18	22	29	41	47
		4	4	27	15	16	17	21	27	45
	2	2	2	12	15	18	22	29	41	47
		4	4	10	13	15	19	25	36	43
4×4	3	4	4	9	11	15	20	26	31	34
8×8	3	4	4	10	13	17	25	30	40	52



(a) 4×4 checkerboard **(b)** 8×8 checkerboard

Figure 4.7: Checkerboard domain problem, $J = 6$: convergence history of the FGMRES iteration for various choices of smoother parameters. Continuous lines: $\hat{S}_J$ = MGCheb(n_S, k_S, m_S); dashed: $\hat{S}_J^{-1}$ = MGCG(n_S, k_S, m_S), cf. Table 4.13. $m_S = 1$.

10^{-8} starting from a random initial guess. In this test, we consider mesh with $J = 2$ and $J = 6$ levels with the selected sets of smoother orders. The required numbers of PCG iterations are shown in Table 4.14. Except for the case $k_S = 0$, the difference between the numbers of iterations on the coarse and the fine problems is insignificant. Also, the differences between 4×4 and 8×8 checkerboards are minor. This indicates that already for $k_S = 2$ the quality of Schur complement is satisfactory.

Spectrum analysis For further understanding of preconditioner properties, we numerically analyze the spectrum of preconditioned system matrix $\mathcal{P}_J\mathcal{M}_J$ with $\hat{S}$ = MGCheb(n_S, k_S, m_S) where $n_S = 1$, $m_S = 1$ and $k_S = 2$, and consider $k_A = 2$ and $k_A = 4$. We use the same grid as previously with a fixed number of levels $J = 6$. For As in single inclusion problem, we are interested in both ends of the spectrum. The eigenvalues were computed using ARPACK as in Sec. 4.3.1.

Figure 4.8a indicates that the great majority of eigenvalues are located close to 1, with distorted both ends of the spectrum. One can observe that the left end of the spectrum mostly depends on number of subdomains. The differences between parameters settings are minor except several smallest eigenvalues. In case where $k_A = 2$, $k_S = 0$ the smallest eigenvalue is positive close to zero. In case where $k_A = 4$, $k_S = 2$ the negative eigenvalues appear. The number of outlying eigenvalues in the right end of spectrum grows with the decrease of polynomial smoothers degree. For $k_A = 2$, $k_S = 0$ the real part of last 33 eigenvalues exceeds the plot range.

The imaginary part of eigenvalues (Figure 4.8b) is significantly greater in case of 8×8 checkerboard, especially in case of $k_A = 2$. In this cases the convergence of FGMRES was significantly improved by using

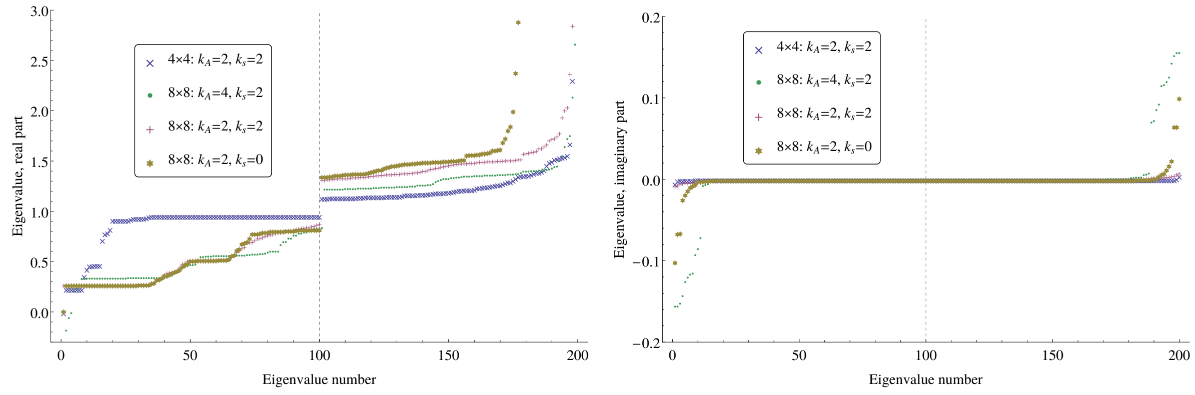
Table 4.14: Dependence of the number of iterations of PCG for $Sx = y$ with preconditioner MGCheb(k_S, 1) on the number of levels J in checkerboard problem for various k_A, k_S, $n_S = 1$.

(a) 4×4

k_S	k_A	J		
		2	6	8
0	0	15	26	30
	2	24	32	35
2	2	15	16	16
	4	17	18	18

(b) 8×8

k_S	k_A	J		
		2	6	8
0	0	16	28	33
	2	26	35	39
2	2	18	19	20
	4	20	21	22



(a) Real part **(b)** Imaginary part

Figure 4.8: Eigenvalue distribution of preconditioned matrix $\mathscr{M}$ for checkerboard domain problem: 100 smallest and 100 largest by magnitude eigenvalues. $J = 6$. The first 100 eigenvalues (numbered 1-100) and last 100 eigenvalues (numbered 101-200).

MGCG(n_S, k_S, 1) as Schur complement approximation (cf. Figure 4.7b). Also, the number of PCG iterations needed for convergence of Schur complement problem is largest — Table 4.14.

Multiple inclusions in 3D: a sinker problem

Let us consider the n-sinker problem, cf. [112], with sharp edges — in contrast to [111, 113] who consider smooth viscosity with large gradient transition layer between the inclusions. In the unit cube with uniform Cartesian coarse grid and element size $h = 2^{-3}$ we randomly pick n disconnected coarse cells and consider them as inclusions with viscosity $\mu_2 = 10^6$. We then uniformly refine the coarse mesh to J levels; in this way all discontinuities are resolved on all relevant grids. In this case, we consider two sets of boundary conditions: the first is the default one, while the second set we set uniform zero Dirichlet boundary condition. The second case requires additional constraint on the pressure, so we fix the first pressure unknown to 0.

With various numbers of levels $J = 2 \ldots 5$ and solver parameters $k_A = k_S = 2$, $n_s = 1$ we illustrate the solver convergence for the number of inclusions n equal to 8 or 16, reporting the number of iterations in Table 4.15. This test reveals the interesting property of the solver, the number of iterations seems to be independent of the number of inclusions and are identical as in the single inclusion case. Similarly to 2D case, the ChebCG is robust with respect to problem size and converges in a bit larger number of iterations than MGCG.

4.3.2 Implementation, performance and scalability

The multilevel preconditioner defined in Section 4.2 was designed to allow straightforward implementation within any matrix-free framework. In our case, we used the matrix-free toolbox [72] of `deal.II` library [73] to implement the method and all tests. This same library also provided its built-in implementation of Chebyshev smoothers used by the preconditioner, together with a generic multigrid framework. The implementation

Table 4.15: Dependence of the number of iterations on coefficient contrasts in 3D n-sinker problem with mixed Dirichlet-Neumann boundary conditions (Mixed) and pure Dirichlet boundary conditions (Uniform Dirichlet). Structured grid with J levels, with fixed parameters $k_A = k_S = 2$ and $n_s = 1$. Upper part: MGCG(n_S, k_S, m_S); lower part: ChebCG(n_S, k_S).

n	Boundary conditions	J			
		2	3	4	5
8	Mixed	8	8	8	8
	Uniform Dirichlet	8	8	8	8
16	Mixed	8	8	8	8
	Uniform Dirichlet	8	8	8	8
8	Mixed	11	11	11	10
16	Mixed	11	11	10	10

with 32-bit indexing of degrees of freedom limited the size of our test, the largest considered problem consisted of over 421 million unknowns.

The required diagonals of matrices A_j and $B_j(\text{diag}A_j)^{-1}B_j^T$ were precomputed once by using formulae

$$e_i^T A_j e_i = \sum_{c\in\mathcal{T}} e_i^T A^c e_i$$

$$e_i^T B_j(\text{diag}A_j)^{-1}B_j^T e_i = \sum_{c\in\mathcal{T}} e_i^T B_j^c(\text{diag}A_j)^{-1}(B_j^c)^T e_i,$$

where A_j^c (and similarly B_j^c) denotes contribution of cell $c \in \mathcal{T}_j$ on j-th level grid (so-called cell, or local, operators) to the global matrix A_j. Note the components of those sums are nonzero for the cell c only if i-th degree of freedom is associated with cell c. Thus, both diagonals may be computed by summing up each cell individual contribution.

To solve the problems on the coarsest level, we assembled the matrices corresponding to $\mathcal{T}_0$ grid and used the MUMPS [128] direct sparse solver. This was the only part of the implementation that required assembling matrices, of relatively small size.

Let us define the overall efficiency E of the method as

$$E = T/N,$$

where T is the wall clock time (in seconds) required by the solver to satisfy the stopping condition and N is the number of unknowns.

We carried efficiency tests of the method on a cluster using two nodes each one equipped with 2 Xeon 8160 processors at 2.10 GHz resulting in total 96 cores. We used 24 MPI ranks, so multi-threading, SIMD vectorization of loops and cache memory optimizations were among the main factors which influenced the timings, because the `deal.II` library takes advantage of both shared and distributed memory parallelization, when available. The single inclusion problem on a structured grid in 2D or 3D, described in detail Section 4.3.1, was used as a benchmark, with the remaining free preconditioner parameters set to $k_A = 2$, $k_S = 2$ and $n_S = 1$. Figure 4.9 indicates that solver's efficiency E levels off for sufficiently large problem size N. The reason that for smaller problems the performance suffers is probably related imbalance of `deal.II` mesh partitioning, a similar issue was present in [113]. The simple Schur approximation without inner MG ($\hat{S}^{-1}$ = ChebCG$(1, 2)$) outperforms the one with inner MG ($\hat{S}^{-1}$ = MGCG$(1, 2, 1)$). The speedup at the finest 3D mesh is over twice.

The same `deal.II` matrix-free framework was used by Clevenger and Heister [113] to implement block preconditioner for the Stokes problem ($\rho = 0$). The obtained timings could be directly compared, as the hardware configuration seems to be the same. Our solver with inner MG cycle seems to require roughly 30%–40% more time than GMG matrix-free solver from [113] to solve the system with the same number of unknowns. On the other hand, the robustness of the preconditioner used in [113] with respect to contrast is limited. The block preconditioner suitable for higher contrast case would be BFBT [111], that is significantly less efficient.

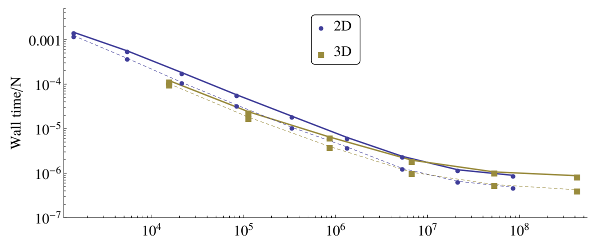


Figure 4.9: Efficiency of the solver, as a function of problem size. Single inclusion, structured grid, $k_A = 2, k_S = 2, n_s = 1$

Table 4.16: Peak memory usage in single inclusion problems

	2D		3D	
	Structured	Unstructured	Structured	Unstructured
N	84,965,379	51,914,755	53,070,468	17,265,796
Number of iterations	11	21	13	21
Memory used [KB]	59,664,668	58,433,252	33,693,900	30,217,736
$\frac{\text{Memory used}}{N}$ [B]	719.1	1153	650.1	1792

Let us finally say a few words about memory consumption requirements of the solver, as matrix-free computations allow extensive memory usage optimization. In our case, a workstation with 64 GB of RAM was capable to handle tests discussed above with the problems consisting of 84,965,379 or 53,070,468 degrees of freedom (in 2D or 3D, respectively), see Table 4.16 for peak memory requirements. For problem of fixed size, it shows direct dependence of the amount of memory required by the solver per degree of freedom on the number of iterations. This has to be attributed to FGMRES, whose auxiliary vectors storage requirements grow with the number of iterations.

4.4 Conclusions

The method presented in this chapter mixes multilevel and block solver approaches, resulting in a preconditioner which is matrix-free and capable of efficiently solving the Stokes problem with highly discontinuous viscosity. The design principle of the method does not depend on any special type of finite element discretization and is relatively simple to implement.

Numerical experiments show that for suitably chosen free parameteres of the method, the convergence rate degrades only weakly with the increase of the viscosity contrast, up to a threshold of order 10^{12} (depending on method parameters). Moreover, its convergence speed seems independent of the problem size on a structured grid, while on unstructured grids the number of iterations grows slowly with the number of unknowns.

The preconditioner presented here has been designed as a genuinely matrix–free method, thus enabling extensive memory and speed optimizations, including potential for the parallelization. For example, our implementation in `deal.II` is able to handle large parallel computations or solve routinely problems with more than 80 million unknowns on a PC with 64 GB of RAM. One of the key design choices which made the method robust with respect to viscosity contrast and mesh type and size, while preserving its matrix–free character, was the extensive use of the auxiliary Chebyshev smoothers (together with an inner multigrid iteration) inside a very efficient block smoother of [83].

For a single inclusion high contrast problem (so called "sinker problem") the convergence speed of the method is comparable to other methods, cf. [110, 120], and is much better than reported in [108]. Note however that further study would be required to make fair and detailed comparison of their convergence rate (due to variations in stopping criteria and other fine details) not to mention efficiency as defined in Section 4.3.2.

For multiple inclusions in 3D, our approach provides robust covergence rate with respect to number of inclusions, coefficient contrast, and problem size. Notably, while our method deals directly with sharp interfaces, many other methods [111, 113, 120] were tested assuming a blurred interface; in [120] it turned out that the performance of that method was sensitive to the thickness of the interface region.

Since the cost of our method strongly depends on the cost of application of $\tilde{S}^{-1}$ to a vector (which is relatively high, as compared to other methods mentioned above), it is crucial to use as small values of solver tuning parameters $m_S = k_S = n_S$ (see Section 4.2.2) as possible. Our experiments suggest that in most cases the break-even point between the precondtioner quality and iteration cost, resulting in lowest (or close to lowest) computation time corresponds to the following set of preconditioner parameters: $m = m_S = k_S = n_S = 2$ and $n = 1$. In 2D it was enough to take $k_A = 2$, while in 3D setting $k_A = 4$ if the mesh is not structured seemed a better choice. It is also worth mentioning that choosing MGCG to solve approximate Schur complement led to an improvement in the efficiency as compared to MGCheb. For a structured mesh, the cheaper variant of the operator $\tilde{S}^{-1}$, that is, ChebCG, can be used resulting in further speedup. In the case of the unstructured grid, the variant with ChebCG may also lead to a reduction of the total solution cost, but on the other hand it requires fine tuning of its parameters. This issue requires a further investigation.

A downside of the method which needs further research is that its performance in 2D depends moderately on the number of high contrast inclusions, which are responsible for a plateau phase in the convergence history. Numerical evidence indicates that the length of this phase is affected by the number of inclusions.

Towards solving the fluid-structure interaction problem 5

In this chapter we finally proceed to solving the FSI problem. We build the solver step-by-step and validate and illustrate every step by solving a benchmark problem. We begin with the demonstration of incompressible flow solver and structural solver separately. Then we discuss and compare two approaches to the mesh deformation, both of them were briefly mentioned in Section 2.2.1. Finally we test the application of both the biharmonic equation and the pseudo-elastic equation as a mesh deformation technique.

5.1 Preliminaries

In most of the tests, we will adhere to the benchmarks proposed by Turek and Hron in [8]. In all those tests a 2D channel with a rigid cylindrical obstacle is considered – as illustrated in Figure 5.1. The exact dimensions of the channel are described by several parameters, we place their values in Table 5.1. The domain Ω is partitioned into Ω_s – an elastic beam attached to the rigid cylinder and Ω_f – the remaining part of Ω. The left side of the channel is set as the inflow, the right side is set as the outflow, while the rest of boundaries are considered as walls. The point A, located at the midpoint of the right end of the beam, will be used in the test to compare results. We use the same coarse grid (depicted in Figure 5.2) for all those tests. The fine mesh is obtained by refining the coarse one, so that we obtain a multilevel structure with J levels. This setting allows us to apply the multigrid preconditioner described in the previous chapter. We use the variant with $\hat{S}^{-1} = \mathrm{MGCG}(n_S, k_S, m_S)$.

Table 5.2 lists the sizes of the discrete problems for various types of meshes and discretization levels J. In particular, the first column ($J = 1$) corresponds to the size of the coarsest problem, posed on $\mathcal{T}_0$.

We note that in all simulations, we use the same implementation of GCE time integration scheme as described in Algorithm 1. The purpose for that was to test the correctness of the Navier-Stokes solver used in the FSI solver. The quantities in this and the next chapter specified without units are by default in SI units.

The implementation is based on `deal.II` 9.2 library[73]. As a coarse solver we use MUMPS [128] via Trilinos library[129]. The we use OpenMPI implementation of MPI standard, the program was compiled with g++ 7.3.

5.2 Solving the Navier-Stokes equation

Our first step towards building the FSI solver is solving transient Navier-Stokes problem. As in Chapter 2, we consider a domain Ω with boundary partitioned into Dirichlet Γ_D and Neumann Γ_N parts. The problem

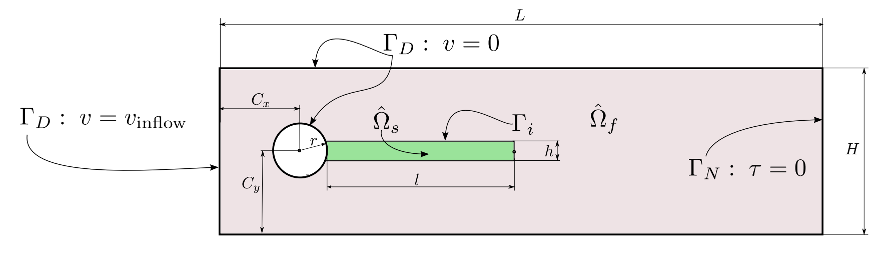


Figure 5.1: Geometrical setting for Turek benchmark problem [8].

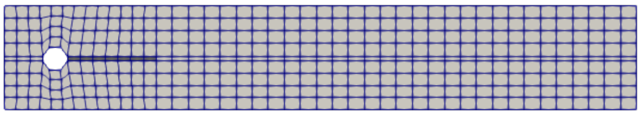


Figure 5.2: Coarse grid used in benchmark problems. The grid was provided by Prof. Luca Heltai.

Table 5.1: Geometry of Turek benchmark problem – list of parameters.

L	H	l	h	C_x	C_y	r
2.5	0.41	0.35	0.02	0.2	0.2	0.05

itself is formulated in the domain Ω:

$$\begin{cases} \rho \frac{\partial v}{\partial t} + \nabla v\ (v) - \operatorname{div}(\frac{1}{2}\,\mu\,(\nabla v + \nabla v^T)) + \nabla p = f & \text{in } \Omega, \\ \operatorname{div} v = 0 & \text{in } \Omega, \end{cases} \tag{5.1}$$

where v is the unknown fluid velocity, ρ is density, μ is viscosity and f is the external force. Additionally, v has to satisfy the boundary conditions:

$$\begin{cases} v & = v_D^* \text{ on } \Gamma_D, \\ \sigma \cdot n & = \tau \text{ on } \Gamma_N \end{cases} \tag{5.2}$$

with given v_D^* and τ. We solve the problem using the time integration scheme discussed in Chapter 2. Let us shortly resemble it in the simplified form, the with steps connected to deformation handling omitted.

5.2.1 Time integration scheme

In this benchmark we use the semi-implicit scheme without corrector. Assuming that the velocities v_{n-1} and v_{n-2} at the times t_{n-1} and t_{n-2} are known, we compute the velocity v_n at time t_n by solving the equation

$$\begin{cases} \rho \frac{3}{2\Delta t}(v_n - \frac{4}{3} v_{n-1} + \frac{1}{3} v_{n-2}) + \nabla v^*\ v^\circ - \operatorname{div}(\frac{1}{2}\,\mu\,(\nabla v + \nabla v^T)) + \nabla p = 0, & \text{in } \Omega \\ \operatorname{div} v_i = 0. & \text{in } \Omega \end{cases} \tag{5.3}$$

where $v^* = v_n$ and velocity v° is extrapolated from previous time steps

$$v^\circ = 2v_{n-1} - v_{n-2}.$$

In this case, we apply the GCE$_2$(1) scheme. Steps that were omitted are in this case trivial operations, but they were preformed in the simulation anyway.

5.2.2 Benchmark problem

As a benchmark problem we compute the flow around a cylinder in a channel. The flow is driven by boundary conditions and external force is not present, i.e. $f = 0$. This is known as the Schäfer–Turek benchmark [130] and was also suggested by Turek–Hron [8] (CFD2 and CFD3 tests) as a part of the test suite for FSI problems. The channel is presented in Figure 5.1 with detailed geometrical data placed in Table 6.1. For this benchmark, we ignore subdomains and consider whole the Ω as a fluid domain.

At the cylinder boundary, the lower and the upper wall of the channel we set the so-called no-slip boundary condition, i.e. zero Dirichlet boundary condition. The right end of the channel is considered as outflow, that

Table 5.2: Geometry of Turek–Hron benchmark problem – list of parameters.

J	1	4	5	8	9
N	3736	223,424	889,216	56,658,944	226,564,096

Table 5.3: Solver settings

Parameter		Value
Order of the Chebyshev smoother defining $\hat{A}^{-1}$	k_A	4
Number m of outer smoothing steps	m	2
Number of outer MG iterations	n	1
Order of the Chebyshev smoother defining $\tilde{S}_{k_S}^{-1}$	k_s	2
Number m_S of inner smoothing steps	m_s	1
Number of inner MG iterations	n_s	1

is modelled by imposing a zero Neumann condition there. Finally, the left side of the channel is considered as an inflow, modelled by setting the Dirichlet boundary condition:

$$v^*_{\text{inflow}} = \frac{3}{2} V_{in} \frac{4}{H^2} y(H - y), \tag{5.4}$$

where V_{in} is the mean inflow velocity and H is the height of the channel. This results in a parabolic velocity profile at the inlet. We use the Reynolds number computed using the mean velocity V and diameter of the cylindrical obstacle:

$$\text{Re} = \frac{\rho V\, 2r}{\mu},$$

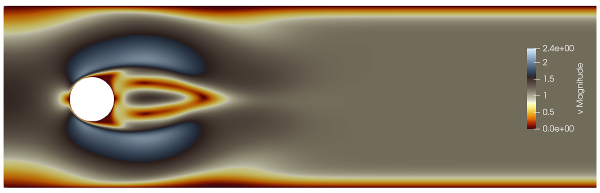
where r is the radius of the obstacle. In our tests, we set the fluid density $\rho = 10^3$, the viscosity $\mu = 1$ and consider various Reynolds numbers. As initial condition we set velocity to zero. We note, that the inflow boundary condition (5.4) enforces instantaneous acceleration of the fluid. This allows us to check the solver ability to cope with advection in the first few time steps. The main problem with rapid acceleration of the fluid is the artificial pressure jumps observed at the first few time steps. Originally in [8] the flow was slowly accelerated, and thus, the initial flow in our case is different then presented in [8].

We derive the triangulation of the domain Ω from the coarse grid, shown in Figure 5.2, by uniformly refining it $J - 1$ times, so that J mesh levels are created. The problem at each time step is discretized using the finite element method, as discussed in Section 3.2. The system of linear equations is solved using FGMRES with multigrid preconditioner described in Chapter 4. A minor modification of the smoother is required to handle a non-symmetric problem, the inner Schur complement operator MGCG is replaced by multigrid preconditioned BiCGStab[131]. The detailed list of smoother parameters is presented in Table 5.3, for detailed description of each one we refer to Section 4.2.2. The solver tolerance is set to $\epsilon = 10^{-6}$, i.e. we claim convergence if the L_2 norm of the residual drops below ϵ.

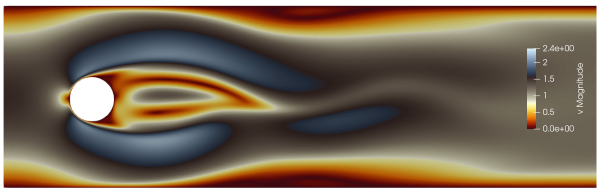
5.2.3 Results

For the experiments we use the fixed mesh with $J = 5$ levels, i.e. the coarse grid was uniformly refined 4 times. We run simulations with time step size $\Delta t = 0.01$ and Reynolds numbers equal to 200 (CFD3), 400 and 1000. The obtained velocity fields are illustrated in Figures 5.3, 5.4 and 5.5. The detailed quantitative comparison with the reference solution [8] requires computing lift and drag forces that have not been implemented and therefore we are unable to provide it. Our excuse is that we are mainly interested in the linear solver performance. Besides, the quantitative comparison will be done while demonstrating the complete FSI solver.

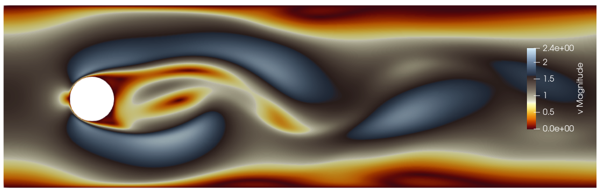
In contrast to the problem considered in Chapter 4, here the advection term is non-zero. The problem in no longer symmetric thus we are interested if the solver can deal with it efficiently. We illustrate the solver performance measured in the number of iterations required for convergence in Figure 5.6. The solver is clearly not robust with respect to the Reynold number, but the the numbers of iteration are still acceptable for us.



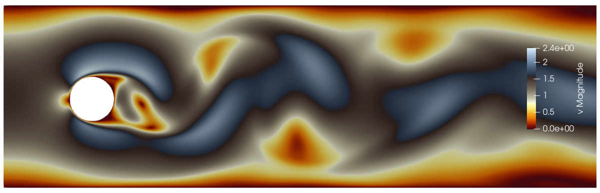
(a) $t = 0.5$



(b) $t = 1.0$

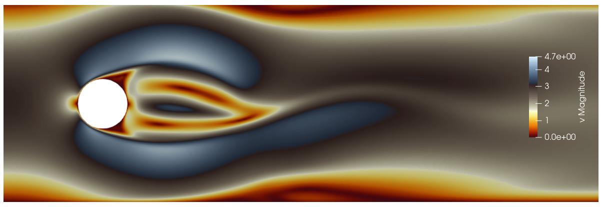


(c) $t = 1.5$

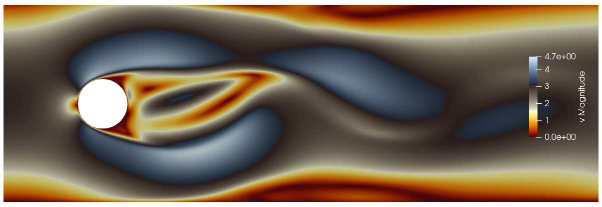


(d) $t = 2.0$

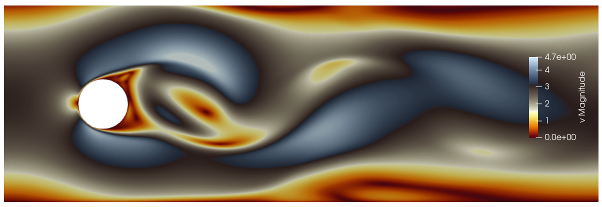
Figure 5.3: Velocity magnitude in Schäfer–Turek benchmark, Re = 200



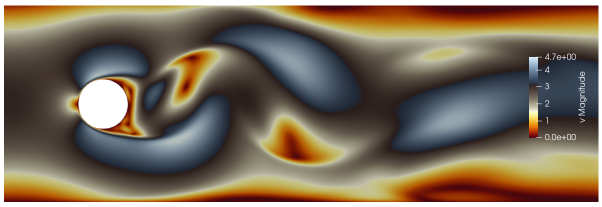
(a) $t = 0.5$



(b) $t = 0.65$

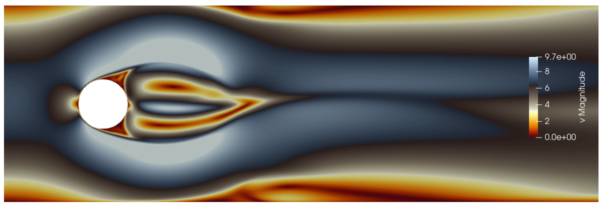


(c) $t = 0.75$

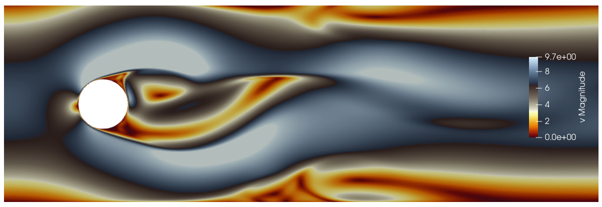


(d) $t = 0.85$

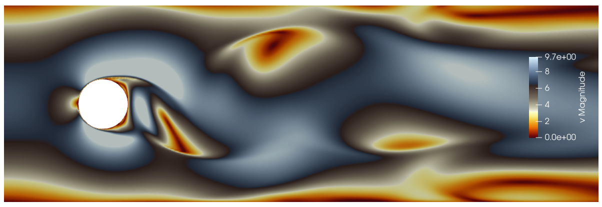
Figure 5.4: Velocity magnitude in Schäfer–Turek benchmark, Re = 400



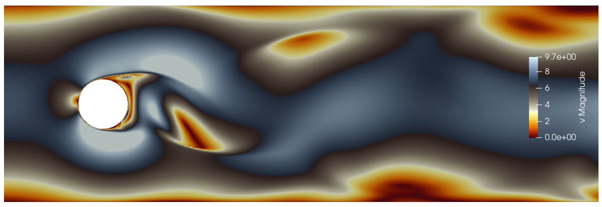
(a) $t = 0.25$



(b) $t = 0.375$



(c) $t = 0.5$



(d) $t = 0.625$

Figure 5.5: Velocity magnitude in Schäfer–Turek benchmark, Re = 1000

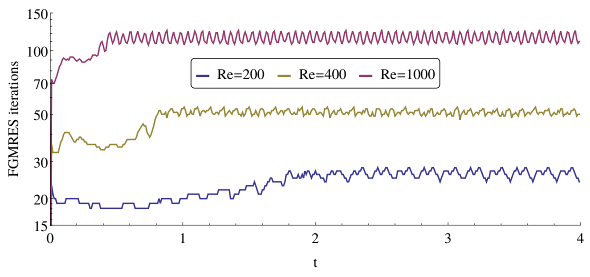


Figure 5.6: Number of iterations versus time in CFD benchmarks. Grid with $J = 5$ levels, time step size Δt =0.01

Figure 5.7: Grid with $J = 3$ levels used in CSM3 tests

5.3 Solving structural dynamics problem

By simulating dynamics of the solid we mean to solve the time dependent problem

$$\begin{cases} \rho \frac{\partial v}{\partial t} + \operatorname{div}(\frac{1}{2}\, \mu\, (\nabla u + \nabla u^T)) - \operatorname{div}\left((\nabla u)^T \nabla u\right) + \nabla p = f & \text{in } \Omega, \\ \operatorname{div} v = 0 & \text{in } \Omega, \\ \frac{\partial \bar{u}}{\partial t} = \bar{v} & \text{in } \bar{\Omega} \end{cases} \tag{5.5}$$

for the velocity v and the displacement $\hat{u}_s$. The solid displacement $\bar{u}$ also defines the transformation between undeformed domain $\bar{\Omega}$ and deformed (current) domain Ω, each point $\bar{x}$ in $\bar{\Omega}$ is transformed to point $x = \bar{x} + \bar{u}(\bar{x})$ in Ω. We approximately solve the problem by first introducing the time discretization, and then using the finite element method for the space discretization as in Section 3. Notice that in this case $\Omega_s = \Omega$, the extrapolation of solid displacement is a trivial operation, therefore the obtained results does not depend on used mesh deformation algorithm.

5.3.1 Benchmark problem

We verify the proposed method by providing a quantitative comparison with the reference solution of the freely oscillating beam – the CSM3 benchmark from [8]. The beam geometry is exactly the same as in the FSI benchmark problem, depicted in Figure 5.1. We fix the beam on the left side, while at the other boundaries we impose a zero Neumann boundary condition. At the initial time, the beam is undeformed and the motion is forced by the external force $f = [0,\ -g]^T$. We consider the incompressible Mooney-Rivlin solid (Section 2.4.2) with parameters $\rho_s = 10^3$ and $\mu_s = 0.5 \times 10^6$. The solid model is different from the referential one [8], that is the compressible St. Venant-Kirchhof solid with $\rho_s = 10^3$, $\mu_s = 0.5 \times 10^6$ and Poisson ration $\nu = 0.4$. The coarse grid used in this benchmark (Figure 5.7) is a solid part of the one used for the FSI problem (Figure 5.2). The finer grids are obtained via uniform refinement of the coarse one.

We solve the linear system with FGMRES preconditioned with the multilevel method, the tolerance is $\epsilon = 10^{-6}$. In this test, we skip linear solver details as the problem solved at each step is a generalized Stokes equation on a structured grid with no coefficient jumps. Those details could be found in Chapter 4, where almost the same problem was considered.

5.3.2 Results

Using the mesh with $J = 3$ levels, we run a series of simulations with the second-order integration scheme GCE$_2$(1) with the time step sizes $\Delta t = 0.1,\ 0.03,\ 0.01$ and volumetric damping coefficient $\eta_V = 0.1$. We plot the displacement of the point A over time – Figure 5.8. The time integration is stable for all considered time

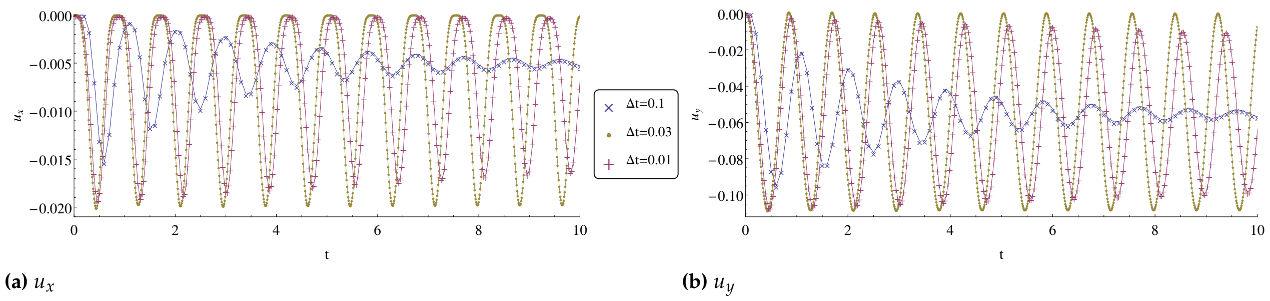


Figure 5.8: Displacement of the point A versus time in CSM3 test with various time step sizes. Grid with $J = 3$ levels, $GCE_2(1)$ scheme with $\eta_V = 1$

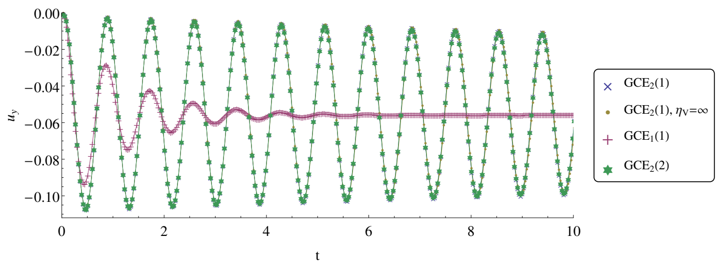


Figure 5.9: Displacement of the point A versus time in CSM3 test with various time schemes. Grid with $J = 3$ levels, $\Delta t = 0.03$

step sizes. The time scheme itself introduces an artificial damping that decreases with the time step size. For $\Delta t = 0.01$ the decrease of the beam amplitude after 10 second is unnoticeable.

For the time step size $\Delta t = 0.03$ we compare the results obtained with various schemes: $GCE_1(1)$, $GCE_2(1)$ and $GCE_2(2)$ with $\eta_V = 0.1$, and $GCE_2(2)$ with $\eta_V = \infty$. The displacement of point A over time is illustrated in Figure 5.9. The results for all second-order methods are nearly identical. In particular, we observe that the introduction of the volumetric damping ($\eta_V \neq \infty$) does not significantly affect the results. The benefits of the volumetric damping will be visible in the FSI benchmark problem where the solid beam is wiggling in the flow.

Next, we set the time step size to $\Delta t = 0.001$ and run the simulation using the grids with $J = 3, 4, 5$ levels. For the quantitative comparison with the reference solution, we note the motion parameters of the first oscillation of the point A: the mean value, amplitude and frequency. We take the maximum and minimum displacements $u_x(\mathsf{A})$ and compute the mean displacement and amplitude as follows

$$\begin{aligned} \text{mean} &= \frac{1}{2}(\max + \min), \\ \text{amplitude} &= \frac{1}{2}(\max - \min). \end{aligned}$$

The frequency is computed as the inverse of the time of the first oscillation. We report the results in Table 5.4 together with referential ones. Both frequency and amplitude indicate that our beam is stiffer than the one in [8]. That discrepancy could be expected since we used a different solid model. Overall, the obtained results are close enough to the referential ones for the purpose of this work.

Table 5.4: Results for CSM3 benchmark problem with time step size $\Delta t = 10^{-3}$

J	N	$u_x(\mathsf{A})$	$u_y(\mathsf{A})$	Frequency
3	1171	-10.216 ± 10.216	-54.649 ± 54.649	1.1961
4	4355	-10.257 ± 10.257	-54.774 ± 54.774	1.1933
5	16771	-10.266 ± 10.266	-54.801 ± 54.801	1.1933
Reference[8]		-14.305±14.305	-63.607 ± 65.160	1.0995

5.4 Handling domain deformation

The last part of the solver that has to be tested before proceeding to the FSI problem is the domain deformation handling method. The problem is to extend the solid displacement (or velocity) on the fluid domain so that the quality of deformed mesh will not deteriorate significantly. The choice of the extrapolation algorithm is crucial for the performance. If done incorrectly, the elements may become overly deformed, resulting in a reduced linear solver performance, or, in the worst case, negative volume of elements leading to meaningless results.

We consider two methods of computing extrapolation $\hat{u}_A$ of solid deformation: using the linear elasticity equation

$$\begin{cases} \operatorname{div}(\mu_A \epsilon(\hat{u}_A)) & = 0 \quad \text{in } \Omega_f, \\ \hat{u}_A & = \hat{u}_s \quad \text{in } \hat{\Omega}_s \\ \hat{u}_A & = 0 \quad \text{on } \partial\Omega, \end{cases} \tag{5.6}$$

with possibly non-constant coefficient μ_A, and the biharmonic equation

$$\begin{cases} \Delta^2 \hat{u}_{A,i} & = 0 \quad \text{in } \Omega_f, \\ \hat{u}_{A,i} & = \hat{u}_{s,i} \quad \text{on } \Gamma_i, \\ \hat{u}_{A,i} & = 0 \quad \text{on } \partial\Omega, \\ \frac{\partial \hat{u}_{A,i}}{\partial n} & = 0 \quad \text{on } \Gamma_i, \end{cases} \tag{5.7}$$

where n is outer normal of the domain Ω_f. The problem is solved for each individual component $\hat{u}_i$ so that extrapolation of $\hat{u}_S$ is obtained. We use the finite element method to solve both equations. The linear elasticity equation appears as a part of $a(\cdot,\cdot)$ on undeformed domain and its weak form was discussed in Section 3.2. The weak form of the biharmonic equation, a forth order problem, is not straightforward.

5.4.1 Solving the biharmonic problem

Let us briefly recall the $\mathscr{C}^0$ Interior Penalty Method ($\mathscr{C}^0$IP) proposed in [132]. The implementation is a modified *step-47* from `deal.II` library that presents a basic biharmonic solver. We improved it with adding handling of parallel computations. We refer to the library manual for further details.

We apply the method similar to the discontinuous Galerkin method, and thus we will need so-called jump:

$$\left\{\!\!\left\{ \frac{\partial^k v_h}{\partial n^k} \right\}\!\!\right\} = \left. \frac{\partial^k v_h|_{K_+}}{\partial n^k} \right|_e - \left. \frac{\partial^k v_h|_{K_-}}{\partial n^k} \right|_e,$$

and average

$$\left[\!\!\left[\frac{\partial^k v_h}{\partial n^k} \right]\!\!\right] = \frac{1}{2}\left(\left. \frac{\partial^k v_h|_{K_+}}{\partial n^k} \right|_e + \left. \frac{\partial^k v_h|_{K_-}}{\partial n^k} \right|_e \right)$$

operators for arbitrary scalar function v_h defined on domain Ω_f. The point e is at the interface between cells K_+ and K_- ($K_-, K_+ \in \mathcal{T}$), the normal n denotes a unit normal vector pointing from K_+ to K_-. The $\mathscr{C}^0$IP formulation of the biharmonic extrapolation problem is to find $\bar{u}_A$ such that $\hat{u}_A = \hat{u}_s \quad \text{on } \Gamma_i$, $\hat{u}_A = 0 \quad \text{on } \partial\Omega$, and

$$\mathscr{A}(v_i, \hat{u}_{A,i}) = 0 \quad \text{holds for all test functions } v \in \mathbb{V}, \tag{5.8}$$

for every component. The $\mathscr{C}^0$IP discrete biharmonic operator is defined as

$$\begin{aligned} \mathscr{A}(v_h, u_h) := & \sum_{K \in \mathbb{T}} \int_K D^2 v_h : D^2 u_h \, dx \\ & - \sum_{e \in \mathbb{F}} \int_e \left\{\!\!\left\{ \frac{\partial v_h}{\partial n} \right\}\!\!\right\} \left[\!\!\left[\frac{\partial^2 u_h}{\partial n^2} \right]\!\!\right] ds - \sum_{e \in \mathbb{F}} \int_e \left[\!\!\left[\frac{\partial^2 v_h}{\partial n^2} \right]\!\!\right] \left\{\!\!\left\{ \frac{\partial u_h}{\partial n} \right\}\!\!\right\} ds \\ & + \sum_{e \in \mathbb{F}} \frac{\gamma}{h_e} \int_e \left\{\!\!\left\{ \frac{\partial v_h}{\partial n} \right\}\!\!\right\} \left\{\!\!\left\{ \frac{\partial u_h}{\partial n} \right\}\!\!\right\} ds, \end{aligned}$$

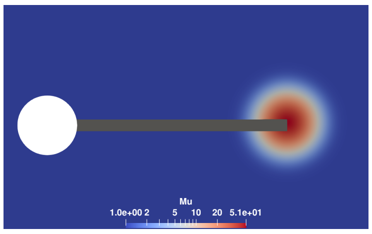


Figure 5.10: Distribution of coefficient μ_A. Solid marked in gray.

where $D^2 v_h$ $D^2 u_h$ are Hessian of v_h and u_h, respectively, and $\mathbb{F}$ is a set of all faces of cell inside. The penalty parameter γ is related to the degree of finite element, in our case $\gamma = 6$.

Iterative solver and preconditioner

The linear system behind the biharmonic problem is solved by the preconditioned CG method. Choosing the multilevel method as a preconditioner appears to be a natural choice, since we already have the multilevel structures. Let us note that the choice of the smoother is not straightforward, although the problem is symmetric and positive defined. For example, the Jacobi smoother is not suitable for this problem, mainly because the matrix is not diagonally dominating.

Developing an efficiencient biharmonic solver is out of scope of this thesis and we do not intend to apply an advanced method if not needed. Fortunately, the domain decomposition method turned out to work well enough as a smoother. The concept we came up with is similar to the one presented in [133]. The simple implementation was possible thanks to Trilinos library [129], especially the Ifpack2 package. In the implementation the `deal.II` wrapper of this preconditioner was used (PreconditionBlockwiseDirect) with the default settings.

5.4.2 Benchmark problem

We compare deformation extrapolation algorithms on the Turek-Hron benchmark problem [8]. As the solid displacement we use

$$\hat{u}_A = \begin{bmatrix} 0 \\ (x - 0.25)^2 \end{bmatrix}$$

that is intended to mimic the deformation of the beam while interacting with the flow. The deformation at the beam tip is $\hat{u}_A(\mathsf{A}) = 0.1232$ that is far greater than the maximum value we expect in FSI benchmarks. In the mesh deformation tests we use the mesh with $J = 4$ levels. This results in a fairly fine mesh, while individual elements are still visible and any problem could be easily spotted. We consider three methods of extrapolating the displacement: using the elastic equation method with both constant coefficient and a pre-cooked coefficient distribution, and using the biharmonic equation. The distribution of the coefficient for the elastic equation is given by the formula

$$\mu_A = 1 + 50\exp\left(-800((x - 0.6)^2 + (y - 0.205)^2)\right)$$

so that the mesh is stiffest at the tip the beam and gradually becomes elastic with a distance from point A.

5.4.3 Results

The deformed mesh is depicted in Figure 5.11 and Figure 5.12. Without varying coefficient the elastic equation provides an extrapolation with deteriorated elements at the tip of the beam. This issue is resolved by introducing a variable coefficient. The biharmonic equation method leads to the least deformed elements, while it is does not require problem-dependent tuning. On the other hand, the linear system is more computationally demanding. Moreover, the second order terms in $\mathscr{A}$ rise further problems, making matrix-free implementation significantly more challenging. In fact, at the time of writing this, the `deal.II` matrix

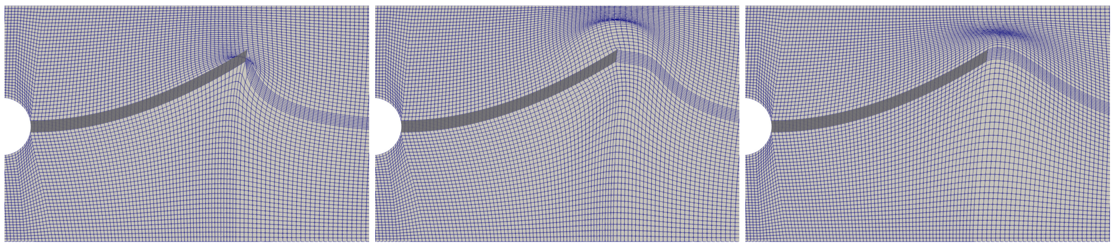

(a) Elastic, uniform coefficient **(b)** Elastic, variable coefficient **(c)** Biharmonic

Figure 5.11: Comparison of mesh deformation algorithms

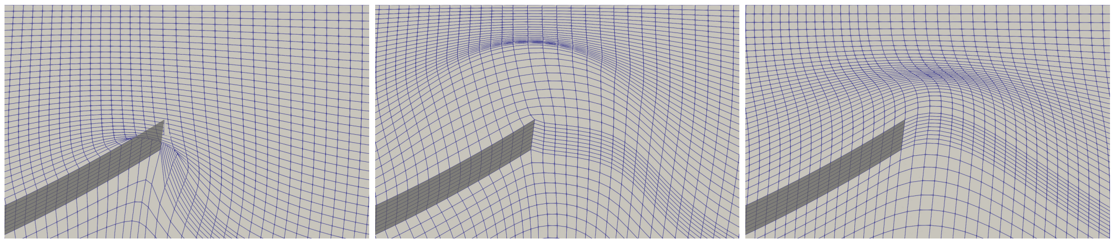

(a) Elastic, uniform coefficient **(b)** Elastic, variable coefficient **(c)** Biharmonic

Figure 5.12: Comparison of mesh deformation algorithms – zoom at the tip

free framework does not support the integration of second order terms. For those reasons, we decided to use the elastic equation based method with chosen a priori coefficient as a mesh deformation algorithm for FSI. Before moving on to the FSI problem, let us provide a bit of details of the linear solver for biharmonic equation.

Iterative solvers details

The construction of an efficient parallel iterative solver for a forth equation is a challenging problem itself, but it is definitely out of the scope of this thesis. However, the used solver is simple yet effective enough so let us provide a brief overview of its performance. The TrilinosWrappers::PreconditionBlockwiseDirect operator from `deal.II` library is a block additive Schwarz method with partitioning based on the parallel mesh partitioning. Thus, the number of MPI ranks is equal to the number of subdomains, that directly affects with the preconditioner performance. When ran in serial, the preconditioner is in fact a direct solver, while in parallel it represents a block-Jacobi operator with block size equal to the local matrix size. We combine it with the Chebyshev iteration of order 4 and use it as a smoother.

To solve the elastic problem we use the CG method with multilevel preconditioner. As a smoother, we choose Chebyshev operator of order 4, at each level we perform 1 smoothing step. Instead of solving the coarse system directly, we perform 1 Chebyshev iteration of order 40. We estimate the eigenvalues using 20 CG iterations and set the smoothing range to $[1.2\frac{\lambda_{\max}}{100}, 1.2\lambda_{\max}]$. This is a standard procedure for the Laplace problem as demonstrated in `deal.II` [73] tutorial *step-37*.

We test the biharmonic solver and the elastic solver in the setting from the previous subsection on the serie of grids with various number of levels and report the numbers of iterations and time required for convergence in Table 5.5. The biharmonic solver was ran with 4 and 8 MPI ranks to illustrate its dependence from the number of subdomains. The computations have been done on a PC with Intel i7-3770K CPU @ 3.50GHz. The program also takes advantage of multi-threading so in all cases the number of floating point operations per second is comparable.

Although the numbers of iterations in the biharmonic solver are comparable to the ones in the linear elasticity solver, the timings are significantly worse. Moreover, the solve wall time does not include assembly of the matrix and initialization of the preconditioner. The second one requires computing LU decomposition of

Table 5.5: Performance of linear solvers inside the mesh deformation solvers. Numbers of CG iterations and wall times

J	N (per component)	Linear elasticity	Biharmonic	
			4 MPI ranks	8 MPI ranks
2	6,368	13 (0.5253s)	7 (0.229s)	12 (0.464s)
3	25,024	15 (2.617s)	8 (1.652)	14 (2.346s)
4	99,200	16 (6.485s)	9 (10.11s)	17 (17.31s)
5	395,008	17 (18.86s)	12 (55.55s)	20 (81.59s)
6	1,576,448	17 (66.25s)	14 (329.9s)	27 (546.8s)

the block owned by each processor. As the result, the initialization is even more time consuming than the solve itself. In case the FSI problem this is not an issue, since the initialization could be done only once per simulation.

For those reasons, we have decided to use only the pseudo-elastic problem to compute the extension.

The fluid-structure interaction solver 6

With all building blocks tested we may finally proceed to test the FSI solver on the problems it was intended to handle. We will first verify if it can provide a stable integration with the results close enough to the referential ones. We will be comparing ourselves with the Turek benchmark problem with Re =100 (FSI2) and Re =200 (FSI3). We do not expect a perfect match since our solid model is different than the one used in [8]. We use the incompressible solid instead of a compressible one, thus we expect that the results will be different, as we observed in CSM tests (Section 5.3). We test various time-stepping schemes and introduce a simple modification to the predictor-corrector scheme exploiting the properties of the iterative solver. We extend the results with the simulations of higher Reynolds numbers Re = 400 and Re = 1000 to demonstrate the stability of the method. We then proceed to the main focus of this thesis – the linear solver behind the momentum sub-step. We illustrate the performance measured in the number of iterations and time required to solve the system.

6.1 The benchmark problem

The individual parts of the benchmark problem were present in the previous sections. Here we test the fully coupled problem, i.e. the elastic beam attached to the cylindrical obstacle inside the channel is interacting with the flow. We use the geometry described in Section 5.1 illustrated in Figure 5.1, with detailed geometrical data presented in Table 5.1.

The boundary and initial conditions are the same as in Section 5.2. The cylinder, upper and lower sides of the channel are considered as rigid walls and we impose no-slip boundary conditions there. At the right end of the channel we set a zero Neumann boundary condition, the left side of the channel is considered as inflow with parabolic velocity profile – Equation (5.4). As a parameter defining the flow we use the average velocity at the inlet V_{in} and the Reynolds number computed with respect to the obstacle diameter. In the benchmark test, we consider $V_{in} = 1$ or $V_{in} = 2$ that corresponds to Re = 100 and Re = 200, respectively. In the test withRe = 400 (FSI400) Re = 1000 (FSI1k) we set the viscosity to $\mu_s = 0.5$ and velocity to $V_{in} = 2$ or $V_{in} = 5$, respectively.

At the initial time, the velocity $v = 0$ and the beam is undeformed. As in Section 5.2, due to boundary conditions the flow is accelerated instantaneously while in [8] the flow gradually speeds up. This change allows us to check the solver capabilities to handle a fully developed flow in the nearly undeformed configuration. This provides a fixed geometry for the solver performance comparison that is especially important when considering various time step sizes, as the solution at any fixed time might change with the time step size due to method accuracy. On the other hand, it rises a problem with the artificial pressure jumps observed in the first few time steps. From our observations, it follows that the pressure oscillations are damped in a few initial time steps and have a negligible impact on the results.

Table 6.1: Parameters used in FSI benchmarks

		FSI2	FSI3	FSI400	FSI1k
ρ_s	$[\frac{\text{kg}}{\text{m}^3}]$	10^4	10^3	10^3	10^3
ρ_f	$[\frac{\text{kg}}{\text{m}^3}]$	10^3	10^3	10^3	10^3
μ_s	$[\frac{\text{kg}}{\text{ms}^2}]$	0.5×10^6	2×10^6	2×10^6	2×10^6
μ_f	$[\frac{\text{kg}}{\text{ms}}]$	1	1	0.5	0.5
V_{in}	$[\frac{\text{m}}{\text{s}}]$	1	2	2	5
Re		100	200	400	1000

Table 6.2: Solver parameters in FSI benchmarks

Parameter		Value	
		Predictor	Corrector
Order of the Chebyshev smoother defining $\hat{A}^{-1}$	k_A	4	6
Number m of outer smoothing steps	m	2	2
Number of outer MG iterations	n	1	1
Order of the Chebyshev smoother defining $\tilde{S}_{k_s}^{-1}$	k_s	4	2
Number m_S of inner smoothing steps	m_s	1	1
Number of inner MG iterations	n_s	1	1
Solver tolerance		$\epsilon_p = 10^{-6}$	$\epsilon_c = 10^{-6}$

6.2 The linear solver settings

The implicit advection scheme results in the non-symmetric linear problem at the momentum step. We solve the system with the FGMRES method preconditioned by the operator discussed in Chapter 4. Unlike the Navier-Stokes problem from Section 5.2 and similarly to Section 4.3 here we deal with the problem with strongly variable viscosity, that appears to be more challenging for an iterative solver. Because of that in the FSI tests we will use a higher smoother orders, the specific values are provided in Table 6.2. The chosen values are probably an overkill, and thus could be optimized for better timings. As in the previous chapter, we use the Schur complement approximation $\hat{S}^{-1} = \text{MGCG}(n_S, k_S, m_S)$ (we replace CG with BiCG in case of non-symmetric problem). We note that the variant without inner multigrid cycle also provides robust convergence, but we came up with it too late to incorporate it in this chapter. Nonetheless, we present some details in Appendix A. We claim the system is solved if the L_2 norm of residual is less than ϵ_p for predictor or ϵ_c for corrector.

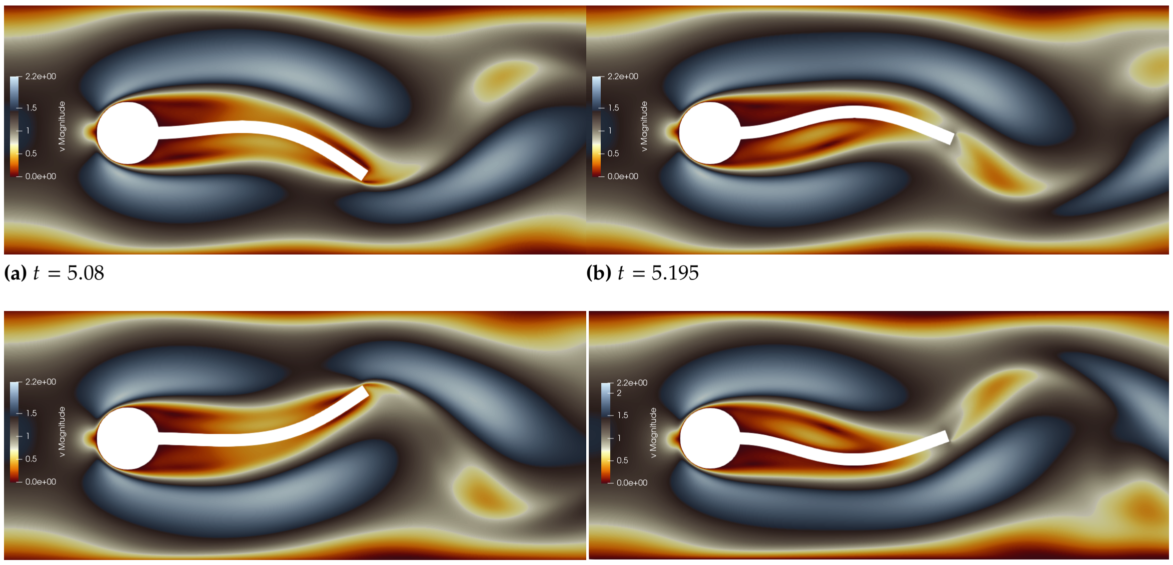
6.3 Comparison with the referential results

As the first test, we solve the benchmark problems FSI2 and FSI3 on the fixed grid with $J = 5$ levels. We use the second-order implicit time integration GCE$_2$(2) with time step size $\Delta t = 5 \times 10^{-3}$. The lower time step size than in Section 5.2 was chosen for accuracy, we were not able to capture the beam oscillations in the FSI3 benchmark with $\Delta t = 0.01$. We use the volumetric damping coefficient $\eta_V = 0.1$, same as in 5.3. As a mesh deformation handler we choose elastic equation method with the coefficient distribution from the previous chapter.

To visualize the results we plot the velocity magnitude at various times, Figure 6.1 and Figure 6.2 for FSI2 and FSI3, respectively. In both cases during the initial period the flow slowly develops building up the vortices behind the obstacle while the beam is moving slightly. Then, due to vortex shedding, several unsteady oscillation occur before the motion stabilizes and becomes periodic. This is best illustrated by plotting the position of point A over time — Figure 6.3.

We report the amplitude, the average displacement and the period of the last computed oscillation in Table 6.3 and Table 6.4 together with the referential values [8]. The difference is almost imperceptible in the FSI2 benchmark, that is surprising if taking into account the increased stiffness of the beam observed in the CSM test (Section 5.3). The only noticeable gap can be spotted in the oscillation frequency in the FSI3 benchmark. To confirm our results we run another simulations with the time step $\Delta t = 0.001$. This time as a starting point we choose the solution obtained in the previous tests at time $t = 7$. We compute another 2 seconds of the solution and complement Table 6.3 and Table 6.4 with data from the last oscillation. This time the amplitude in the FSI2 benchmark is a bit lower than in referential results, confirming the intuitive guess.

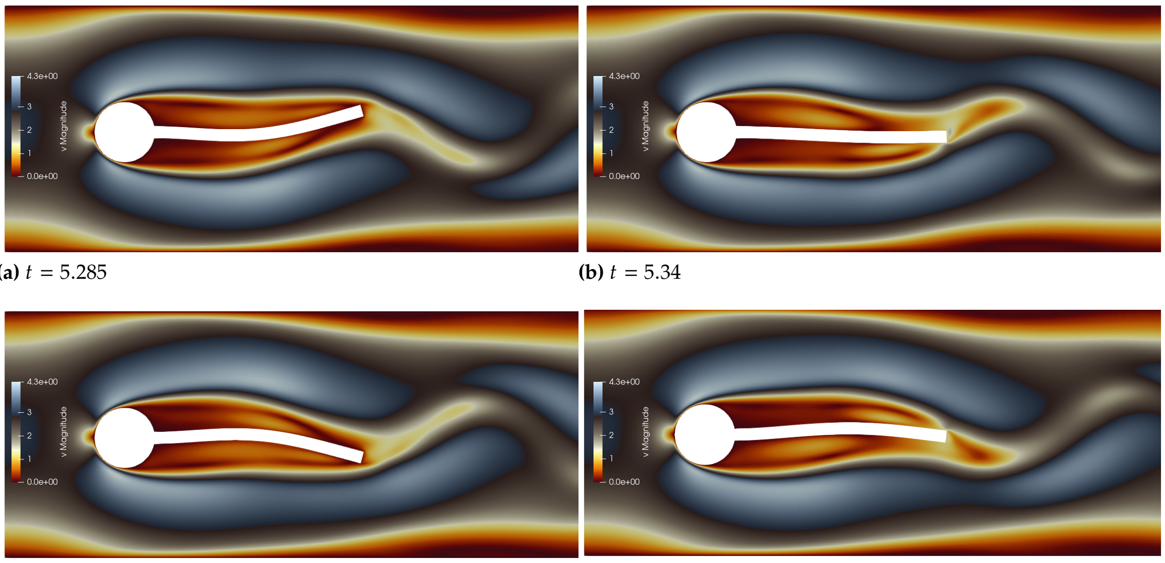
In the FSI3 benchmark we observe 10% higher frequency than in reference results. We guess that in this case the dominating factor defining the flow is the beam stiffness. Compared to FSI2 benchmark we have higher



(a) $t = 5.08$ **(b)** $t = 5.195$

(c) $t = 5.315$ **(d)** $t = 5.455$

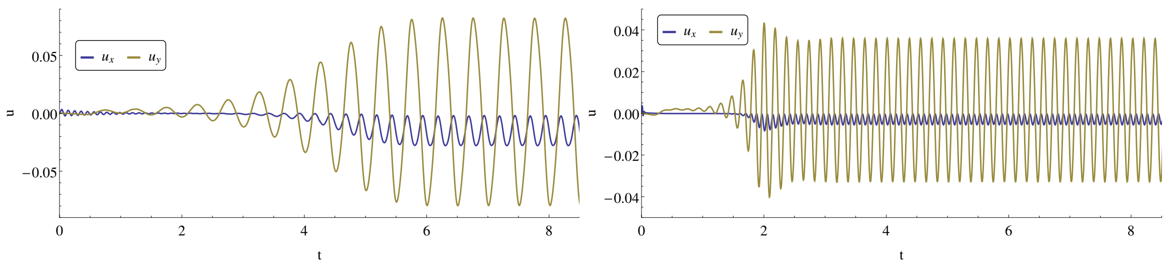
Figure 6.1: Velocity magnitude in the FSI2 benchmark problem



(a) $t = 5.285$ **(b)** $t = 5.34$

(c) $t = 5.375$ **(d)** $t = 5.42$

Figure 6.2: Velocity magnitude in the FSI3 benchmark problem



(a) FSI2 **(b)** FSI3

Figure 6.3: Displacement of the point A versus time in the FSI benchmark problems. Grid with $J = 5$ levels, GCE$_2$(2) scheme with $\eta_V = 0.1$

Table 6.3: Comparison of displacements and frequency in the FSI2 benchmark

	$u_x(\mathsf{A}) \times 10^{-3}$	$u_y(\mathsf{A}) \times 10^{-3}$	Frequency
$J = 5, \Delta t = 0.005,$	-14.85 ± 12.89	1.25 ± 80.8	2.00
$J = 5, \Delta t = 0.001$	-14.34 ± 12.19	1.10 ± 78.5	1.99
Reference	-14.58 ± 12.44	1.23 ± 80.6	2.0

Table 6.4: Comparison of displacements and frequency in the FSI3 benchmark

	$u_x(\mathsf{A}) \times 10^{-3}$	$u_y(\mathsf{A}) \times 10^{-3}$	Frequency
$J = 5, \Delta t = 0.005$	-2.79 ± 2.46	1.47 ± 34.38	5.40
$J = 5, \Delta t = 0.001$	-2.73 ± 2.57	1.55 ± 34.68	5.52
Reference	-2.69 ± 2.53	1.48 ± 34.38	5.3

stiffness to density ratio and thus the eigenfrequency of the beam is closer to the vortex shedding frequency. Some near-resonant phenomena may also come into play in this case.

Having in mind that we use a different solid model, those results are satisfactory enough to claim correctness of the FSI solver implementation. Unfortunately, we are unaware of any results for Turek benchmark problem with an incompressible solid to compare with.

6.3.1 Influence of volumetric damping

In Section 5.3 we claimed that the volumetric damping does not have negative impact on the model, let us now show that this newly introduced feature of the time integration scheme actually improves it. For that, we repeat the FSI3 benchmark with ($\eta_V = 0.1$) and without ($\eta_V = \infty$) the volumetric damping. We set the time step size to $\Delta t = 0.005$ and run the simulations using the second-order time integration scheme using grid with $J = 5$ levels.

To illustrate the results we plot the deformation at point A — Figure 6.4. The horizontal displacement u_x reveals that the point A is slowly creeping (Figure 6.4a) and the motion of the beam is no longer periodic. To visualize this effect we plot the deformed beam obtained without volumetric damping and compare it to the one at the similar phase of oscillation obtained in the referential simulation with $\eta_A = 0.1$ — Figure 6.5. The analogous effect also occurs in case of the first-order time integration scheme where the beam is gaining volume. The introduction of volumetric damping clearly resolves this issue.

6.3.2 Importance of the corrector step

To test if the corrector step is crucial for the accuracy of the method we again run the FSI2 benchmark problem using the mesh with $J = 5$ levels and $\eta_V = 0.1$. This time we apply the GCE$_2$(1) scheme with semi-implicit advection. We choose the FSI2 benchmark for two reasons: the beam and thus also the domain deforms much more than in the FSI3 benchmark, and secondly, the Reynolds number is lower reducing the time required to

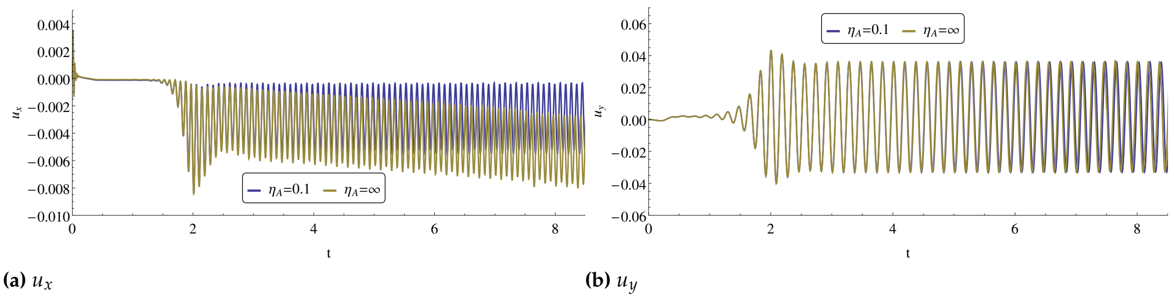
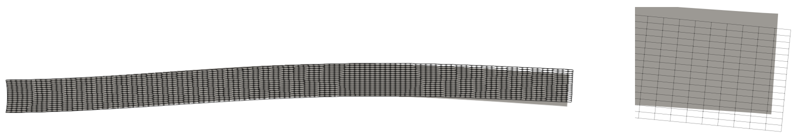
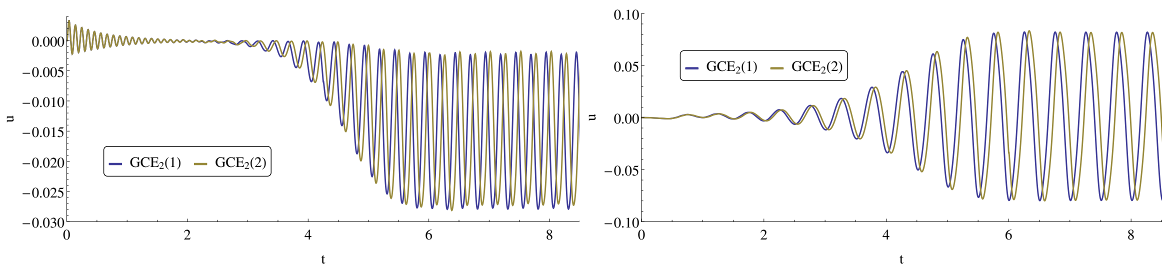


Figure 6.4: Displacement of the point A versus time in the FSI3 benchmark problems. Grid with $J = 5$ levels, GCE$_2$(2) scheme.



(a) Beam **(b)** Zoom at the tip

Figure 6.5: Comparison of beam deformation with (grey) and without (black wireframe) volumetric damping at time $t = 7.53$ and $t = 7.5$ respectively.



(a) u_x **(b)** u_y

Figure 6.6: Comparison of GCE$_2$(1) and GCE$_2$(2) schemes. Displacement of the point A versus time in the FSI2 benchmark problem. Grid with $J = 5$ levels.

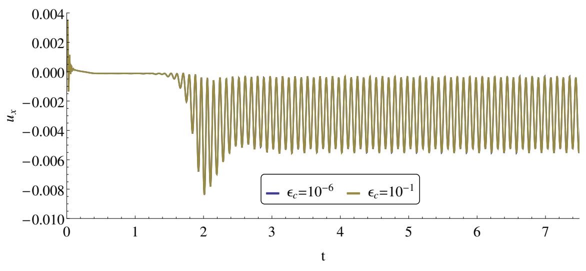
obtain the results. As previously, we compare the results by plotting displacement of point A – Figure 6.6. The results are fairly similar, but the solution obtained without the corrector seems to be falling behind the ones obtained using the predictor-corrector scheme.

6.4 Modified predictor-corrector scheme

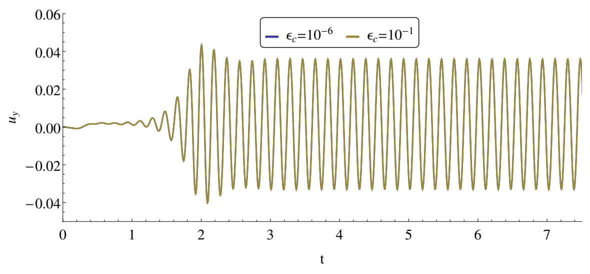
In our test we prefer the implicit time integration method for stability reasons. The time step size $\Delta t = 5\times10^{-3}$ is not small enough for the explicit time integration scheme to be stable, even in the FSI2 benchmark problem (Re =100). From our experiments it follows that the largest time step size ensuring stable explicit integration appears to be $\Delta t = 5 \times 10^{-4}$ for the FSI2. For comparison, the implicit one is stable with the time step size exceeding $\Delta t = 10^{-2}$ in both FSI2 and FSI3 benchmarks. With each implicit time step being more expensive than the explicit one (details in the next section), the overall performance of the implicit scheme is significantly better. With increasing Reynolds number the stability limit of the explicit time scheme deteriorates and usage of the implicit time scheme becomes even more preferable. This, however, has its price, as the number of FGMRES iterations also increases with Reynolds number.

On the other hand, the explicit time scheme could be used to compute the prediction of the solution at the next time step. This provides a better approximation of velocity v° so that our implicit scheme could be closer to the fully implicit one. In this setting, the implicit step is used as a corrector, and thus, the scheme becomes a predictor-corrector scheme. Notice that the predicted velocity may be used as a starting point for preconditioned FGMRES. In these settings, the iterative solver provides a gradual transition between the explicit and implicit scheme. We exploit further this property by playing with the corrector solver parameters. We keep the high accuracy of the predictor while we accept the solution with a higher residual norm in the corrector. In this way, we hope to obtain a scheme with stability significantly better than the explicit one, while the overall cost could be reduced.

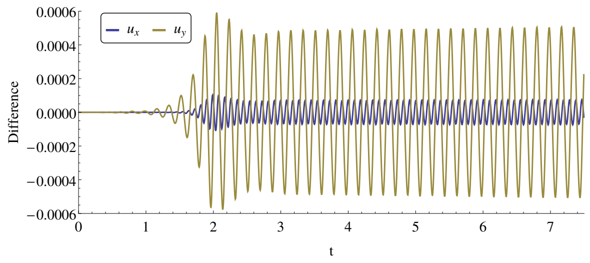
Following this idea, we test the predictor-corrector scheme with tuned iterative solvers inside the momentum steps. In the explicit predictor step we set the solver tolerance to $\epsilon_p = 10^{-6}$ while in the implicit corrector step we set the tolerance to $\epsilon_c = 0.1$. We test our hypothesis on the FSI3 benchmark using grid with $J = 4$ levels, and compare the obtained displacement $u_{0.1}(\mathsf{A})$ to $u_{10^{-6}}(\mathsf{A})$, obtained using $\epsilon_c = 10^{-6}$ — Figure 6.7. Since the



(a) u_x



(b) u_y



(c) Difference $u_{10^{-6}}(\mathsf{A}) - u_{0.1}(\mathsf{A})$

Figure 6.7: Comparison of the schemes with various corrector tolerances. Displacement of the point A versus time in the FSI3 benchmark problem. Grid with $J = 4$ levels, GCE$_2$(2) scheme with $\eta_V = 0.1$

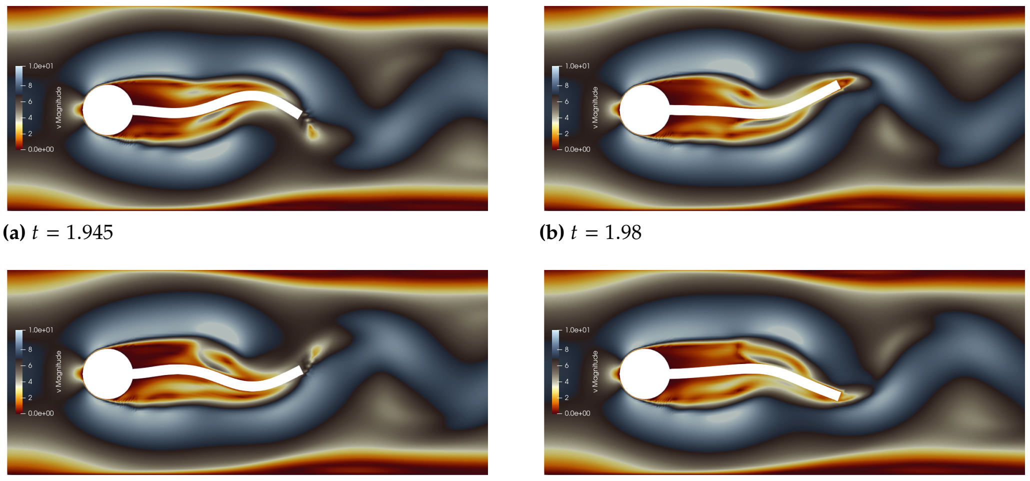
differences are barely visible we also plot the difference — Figure 6.7c. It appears that the difference in not growing over time and the displacement $u_{0.1}(\mathsf{A})$ lags behind $u_{10^{-6}}(\mathsf{A})$ by a small fraction of the time step.

Using the modified method we run the FSI1k test with Reynolds number Re = 1000 with $\Delta t = 0.005$ ($\epsilon_c = 10^{-2}$) and $\Delta t = 0.0025$ ($\epsilon_c = 10^{-1}$) to demonstrate its stability. We note that a larger time step size required a higher precision inside the corrector step for stability. We close this section by presenting visually-pleasing pictures of the obtained flow in a selected time steps — Figure 6.8. Notice the small wrinkles in velocity field near the tip of the beam or downwind side of the cylinder. This type of artefact is a result of the low corrector accuracy combined with a relatively large time step size, especially in comparison to Reynolds number. We note that in this case less than 150 time-steps were done per one oscillation of the beam. For integrating higher Reynolds number problem, either the time step size or the FGMRES convergence threshold should be decreased.

6.5 The linear solver performance

Finally, we share selected insights into the multilevel preconditioner for the FGMRES. Majority of computations were done using several nodes of Marconi supercomputer, each one equipped with 2 Xeon 8160 CPU@2.4GHz (24 cores each) and 196 GB of RAM. All presented timings in this section were obtained using this machine. The program utilizes both distributed (MPI) and shared memory (multi-threading) computing. Additionally, as in Section 4.3.2, the underlying `deal.II` implementation of the matrix-free framework takes advantage of SIMD vectorization.

First, we present a detailed breakdown of the computational time spent at the first time step of the FSI2 benchmark with $\Delta t = 0.005$ using 2 nodes (96 cores total, 24 MPI ranks) — Table 6.5. We have chosen the FSI2 benchmark because the number of FGMRES iterations, in this case, does not vary with mesh refinement. As we will later show, for higher Reynolds numbers the FGMRES surprisingly becomes more efficient after exceeding a certain number of mesh levels. Note that Table 6.5 does not show differences in timings between the predictor and the corrector that arise with Reynolds numbers. We will illustrate that using the number of FGMRES iterations.



(a) t = 1.945 **(b)** t = 1.98 **(c)** t = 2.05 **(d)** t = 2.035

Figure 6.8: Velocity magnitude in the FSI1k problem

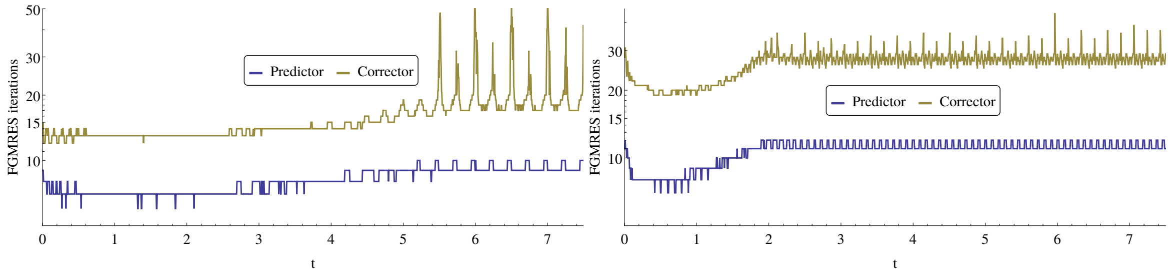
Table 6.5: Breakdown of wall time spent at the first time step of the FSI2 benchmark problem

J =	4	5	6	7	8
N =	223k	889k	3.5M	14.1M	56.6M
Predictor					
Implicit step	44%	45%	44%	46%	48%
Extension	0.45%	0.93%	1.5%	1.7%	1.6%
Corrector					
Implicit step	43%	45%	42%	40%	38%
Extension	0.63%	1.3%	1.7%	1.7%	1.7%
Output results	11%	5.2%	7.7%	7.9%	7%
Total	4.67s	8.68s	23.4s	85s	346s

Some less time-consuming operations such as right-hand side assembly have been omitted, thus the time percentages do not sum up to 100%. Among the non-listed part of the time step is the computation of the new geometry that is done by updating geometrical mapping stored inside matrix-free operators.

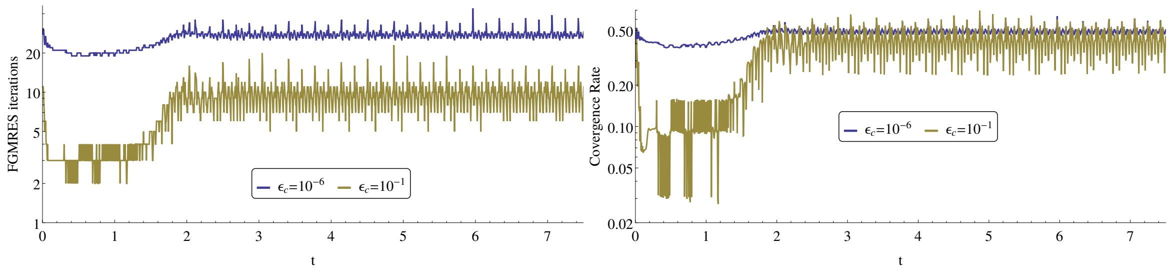
Our results also indicate that around 0.5 s of every implicit step is consumed on assembling the coarse Schur complement matrix. This is because the assembly is done by multiplying unit vectors by coarse Schur complement operator (implemented as a matrix-free object). This mainly impacts performance if the number of levels is relatively low, the process is surely inefficient and should be optimized. Replacing it by a proper assembly using matrix multiplication would greatly reduce this timing. This was not done since in the near future we plan to improve the Stokes solver so that it would not require the inner MG cycle. A significant part of the computational time is consumed on computing eigenvalues required for the Chebyshev smoothers. This part could be also greatly improved by refreshing eigenvalues once per several time steps.

The most time consuming part is the solution of the linear system associated with the implicit part of the time step, as it covers over 80% of the total computation time and the percentage increases with mesh refinement. This is also the only part of the algorithm the computational cost of which may vary during the simulation. Therefore we will focus on this step.



(a) FSI2 **(b)** FSI3

Figure 6.9: Number of FGMRES iterations versus time in FSI benchmark problems. Grid with $J = 5$ levels, $\Delta t = 0.05$



(a) Number of iterations **(b)** Covergence rates

Figure 6.10: Comparison of solver performance: modified vs original scheme. FSI3 benchmark, $\Delta t = 0.005$, $J = 5$

6.5.1 Preconditioned FGMRES performance

As a measure of performance of the preconditioned FGMRES solver we use the number of iterations n required to meet the stopping criterion, and convergence rate defined as

$$\rho = \left(\frac{||r_n||_2}{||r_0||_2} \right)^{\frac{1}{n}} \tag{6.1}$$

where r_i is the residual after i-th iteration. In case of the corrector step that uses the semi-implicit advection the number of iterations may vary in time as in Section 5.2.3. Additionally the variable geometry may also induce some changes in convergence of FGMRES inside the predictor step.

We present the number of iterations by plotting it against time in Figure 6.9a (FSI2) and Figure 6.9b (FSI3). The number of iterations in the predictor step is pretty much constant throughout the remaining part of the simulation while in the corrector step we observe a clear time dependence. There are several reasons behind this effect: the variable geometry, variable advection velocity and variable starting point of the solver. We believe that two lasts are the main contributors to the effect since it appears also in the FSI3 benchmark where the domain deformation is tamer. The rapid spikes of the number of iterations occur when the tip of the beam is at the turning point. We suspect that the dynamically changing flow is the main contributor to this effect. Besides we would also expect notable changes in the convergence of the predictor solver while it is clearly not an issue here.

Modified predictor-corrector scheme

We introduced the modified time integration scheme to workaround the reduced linear solver performance in the corrector step. Let us demonstrate the profit coming out of this by plotting the number of corrector iterations versus time — Figure 6.10a. The numbers of iterations in the predictor step are unchanged (thus we skip plotting them) while the corrector step requires much less effort.

Apart from stopping the FGMRES sooner, with the modified scheme we also exploit the relatively good convergence rate in the first few iterations. To illustrate this, we compare the corrector convergence rates in

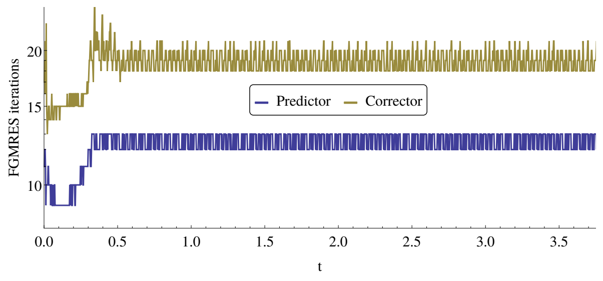


Figure 6.11: Number of iterations in FSI1k problem with $\Delta t = 0.0025$, $J = 5$

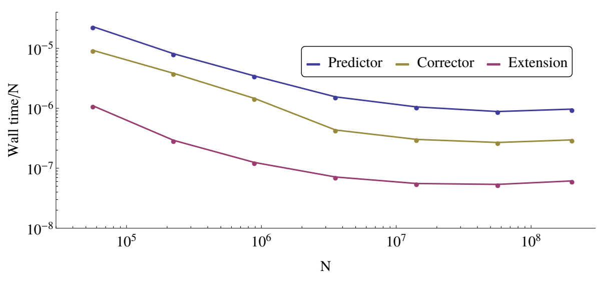


Figure 6.12: Efficiency of the solver, as a function of problem size N. FSI2 benchmark,$\epsilon_p = 10^{-6}$,$\epsilon_c = 10^{-1}$

Table 6.6: Number of FGMRES iterations required for convergence ($\epsilon = 10^{-6}$) at the first time step for varying number of levels J with fixed $\Delta t = 0.005$.

	$J =$ $N =$	4 223k	5 889k	6 3.5M	7 14.1M	8 56.6M	9 226M
FSI2	Predictor	8	9	9	10	10	10
	Corrector	15	13	11	10	9	9
FSI3	Predictor	11	11	12	12	12	12
	Corrector	26	28	23	20	19	18
FSI400	Predictor	11	11	11	12	12	12
	Corrector	28	33	32	30	25	23
FSI1k	Predictor	12	12	12	12	13	13
	Corrector	43	51	168	174	82	52

the FSI3 benchmark problem – Figure 6.10b. The difference is conspicuous in the initial period when the flow calmly builds up, hence the solution does not vary significantly between time steps.

We additionally include the numbers of iterations obtained in FSI1k test with $\Delta t = 0.0025$ to demonstrate the solver ability to cope with higher Reynolds numbers – Figure 6.11.

6.5.2 Performance and scalability

To illustrate robustness with respect to problem size we measure the computational cost of solving the linear system in a number of FGMRES iterations required to reduce the residual below the threshold $\epsilon = 10^{-6}$, the obtained results are presented in Table 6.6. The number of iterations in the predictor step resembles the results obtained in Section 4.3, as it grows insignificantly with problem size.

An intriguing effect is observed concerning the number of iterations in the corrector step. The ameliorating convergence rate may seem counter-intuitive at first glance, however, it can be explained. Notice that the cell Pecelet number (cf Section 3.2.1, equation (3.58)) abates at finer grids, therefore at the cell level the problem shifts towards a diffusion dominated one. The point where this effect begins to be noticeable depends on the Reynolds number that we observe in the obtained results. Unfavorably, the solver performance deteriorates with increasing Reynolds number.

We also measure of solver efficiency, defined as wall time required for the solver to converge divided by number of unknowns. We set the thresholds $\epsilon_p = 10^{-6}$ and $\epsilon_c = 10^{-1}$ and solve the first time step of the FSI2 benchmark. As Figure 6.12 reveals, the presented method is more efficient at larger problems, that resembles the results obtained in Chapter 4. We refer to Appendix A for performance details of the solver without the inner multigrid cycle.

7 Final remarks and conclusion

Let us shortly present our conclusions, observations and essential findings of this thesis. Here we focus on the FSI solver as a whole, for the discussion of the generalized Stokes solver we refer to Section 4.4.

7.1 Possible extensions and improvements

Although the results reported in the thesis seem quite satisfactory, there is a lot of things that could be improved. We already have some ideas on how to speed it up or to resolve some of the issues. Let us here briefly discuss our vision of further advancement.

7.1.1 Improved Stokes solver

Following the timings provided in Chapter 6 we claim with no doubt that the variable coefficient generalized Stokes solver is the most time consuming part of our implementation. When computing velocity extension we solve a linear system with a comparable number of unknowns using the same matrix-free framework, but the entire operation takes around 50 times less time than the implicit step. It is hard to compare the linear elasticity equation with the generalized Stokes problem but we believe that a great majority of computational time could be saved. Moreover, we have an idea of possible improvements.

The main difference between the smoother used in the generalized Stokes problem and the one in elasticity equation is the inner multigrid cycle. This is probably the most time-consuming part of the algorithm, hence avoiding it should greatly increase the performance. One way to achieve that is to consider the discontinuous Galerkin formulation [134] and exploit its properties. For instance, the smoother presented in [114] falls into the class of operators defined by Zulehner [83] with the exact inverse of Schur complement as in [82]. Therefore, following our result, we expect at least similar robustness of this smoother with respect to coefficient jumps. The efficient application of the preconditioner is challenging – the operator relies on applying inverses of the local matrix to cells sharing one vertex (so-called vertex-patch). Combining it with a matrix-free approach is far from straightforward, while the standard matrix-based implementation is significantly less efficient than the solver we used in this work. We have a limited, but promising evidence that the matrix-free implementation of the preconditioner based on vertex-patch smoothing could be implemented in a matrix-free way. Our initial estimates suggest that the improved solver may greatly outperform the current one. Still, a lot of work has to be done to achieve that.

We also consider replacing finite element method with a non-conventional method like Lattice-Boltzman Method [135]. Since the most time consuming part of the scheme is similar to the one appearing in simulations of non-Newtonian fluids, we believe it may be possible.

7.1.2 Known issues

Our implementation, written solely by the author of this thesis have been carefully tested. However, it is surely long and complex enough to include some very special but unintended features, also known as "bugs". Location of some of them is known to us, about the rest we will probably learn via later surprises. In this section, we humbly admit the ones we know of.

Implementation of neo-Hooken solid

In Section 2.4.2 we discussed the neo-Hookean solid since it is a popular model. Initially, we intended to also include this model in experiments, however it leads to form $a_v(\cdot,\cdot)$ with some integrals over undeformed solid domain $\hat{\Omega}_s$. This could be done with some manipulations of our implementation, but we failed to do that correctly. As a result, we obtained expected beam oscillations in CMS3 benchmark only with absurdly small time step size. Since the Mooney-Rivlin model is equivalent to neo-Hookean in 2D, and additionally, does not require integration over $\hat{\Omega}_s$ we decided to give up this idea.

Scalability limits

We have tested the implementation over 1000 MPI ranks. The first problem we have encountered was associated with the direct solver at the coarse level. At around 1500 MPI ranks MUMPS refused to solve the coarse problem, throwing an exception and crashing the program. Honestly to say, for now, we have no idea how to trace the problem occurring only with so big numbers of MPI ranks. Thus, we are not even sure if the problem is inside our implementation or its MUMPS internal issue. The 1000 MPI ranks, together with mutithreading allows us to use 48,000 cores (one MPI rank per node), that is far more than we have currently available at our disposal.

7.2 Conclusions

This thesis presents a complete monolithic FSI solver, that is capable of solving problems with at least hundreds millions of unknowns. To achieve that, we employed the ALE formulation of the problem involving interaction between incompressible hyperelastic solid and incompressible fluid. We then introduced a predictor-corrector time integration scheme by improving the Geometry-Convective Explicit scheme. The key ingredient to ensure the constant volume of the solid is volumetric damping proposed in this work. Our semi-implicit scheme uses a formulation where velocity and pressure are the main unknowns, while the other fields are computed as dependent variables. To solve the problem at each time step we used the Finite Element Method in spaces satisfying the LBB condition. Moreover, the proposed time integration scheme is relatively simple to implement.

The most time-consuming part of the algorithm is computing the solution of the generalized Stokes problem with variable coefficients. This has to be done one or two times per each time step, depending on the variant of the predictor-corrector scheme. To solve such a problem we used a preconditioned Krylov subspace method with a dedicated preconditioner. We proposed the new preconditioner that mixes multilevel and block solver approaches, which is capable of efficiently solving the Stokes problem with a discontinuous viscosity. According to numerical experiments, the solver is robust with respect to both mesh size and coefficient jumps. In the case of semi-implicit advection, the performance decreases with growing Reynolds number.

We tested our FSI solver on the benchmark problem. Although our solid model was different than the one used in [8], we obtained a surprisingly close match. Unfortunately, we are not aware of any referential results computed using a similar solid model to compare with.

The maximum time step size of the proposed scheme depends on the treatment of advection. We did not encounter any stability issues if semi-implicit formulae were used but, in case of explicit advection, the time step size is limited. We believe that the instabilities are not associated with the explicit geometry treatment, since the maximum time step size depends mostly on Reynolds number and we did not observe any geometry-related problems.

The entire FSI solver presented in this thesis was designed and implemented genuinely matrix-free, adding our humble contribution to unleash the power of modern high-performance computers for solving FSI problems. The main bottleneck of the program is solving the generalized Stokes problem. However, our implementation is modular and any part could be easily replaced if any better substitute will be available.

In this work, we did not consider any 3D FSI problem. Its lack may be disappointing, but we have several reasons for that. First, the most commonly used 3D benchmark problem is a flow in an elastic cylindrical tube. The problem is intended to mimic the dynamics of a blood vessel and requires pressure boundary conditions at the inflow. This feature was not yet implemented. Secondly, even though the solver may seem efficient, 3D computations require much more computing power than 2D ones. We note, that we did not encountered any problem when testing our solver on 3D problems.

7.2.1 Original contributions of the thesis

- A new monolithic matrix-free FSI solver was developed
- The solver was implemented within the framework of the finite element library `deal.II`, exploiting its ability to handle parallel computations
- Geometry-Convective Explicit scheme was extended to include incompressible hyperelastic Mooney-Rivlin solid
- A new preconditioner for the generalized Stokes problem with discontinuous coefficients was proposed
- A volumetric damping was introduced in the solid part to improve conservation of volume
- A direct connection between the generalized Stokes problem with discontinuous coefficients and the FSI problem was shown

Improved Stokes solver A

At the final stage of preparing this thesis, we have found out that our multilevel preconditioner does not require nested multigrid for Schur complement approximation. Instead, it is enough to perform 1 CG iteration (BiCG in case of a nonsymmetric problem) preconditioned by a Chebyshev iteration. More precisely, we use $\hat{S} = \tilde{S}$ (cf. Equation 4.15). This choice requires higher-order Chebyshev operator in case of the unstructured grid.

In the meantime, we also replaced the eigenvalue estimation method: the maximum eigenvalue approximate is now computed by 40 iterations of the power method. This allows us to use a better approximation of the maximum eigenvalue, therefore we can lower the safety factor to $s_1 = 1.05$ (cf. Section 4.2.2) expecting an improved solver performance. This solution, however, comes at a higher cost of eigenvalue estimation.

Below we tested the new variant of the solver in Section 4.3, here we demonstrate the differences by solving the first time step of FSI2 benchmark problem using the settings from Table 6.2 and a reduced corrector accuracy $\epsilon_c = 10^{-1}$. We plot the efficiency versus the number of degrees of freedom – Figure A.1. For comparison, we also place the timings from Section 6.5.1 (dashed lines). For all considered numbers of levels $J = 2, \ldots, 8$ the predictor converged in 10 iterations while the corrector converged in 3 iterations. We note that the initial residual norm at the corrector step is over 10^4 thus the solver reduced the residual by 5 orders of magnitude. The speedup is between 2 to 4 times, depending on the mesh size. The possibility to skip the inner multigrid cycle will also greatly reduce work required to implement the support for adaptively refined grids.

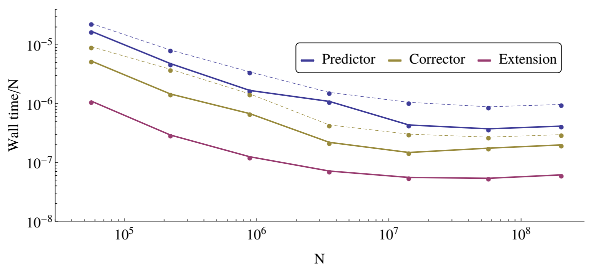


Figure A.1: Efficiency of the solver, as a function of problem size N. FSI2 benchmark,$\epsilon_p = 10^{-6}$,$\epsilon_c = 10^{-1}$.

Bibliography

Here are the references in citation order.

[1] R. Samannan et al. 'Effect of Face Masks on Gas Exchange in Healthy Persons and Patients with COPD'. In: *Annals of the American Thoracic Society* ja (2020) (cited on page 1).

[2] J.C. Bailar et al. 'Reusability of facemasks during an influenza pandemic'. In: *Institute of Medicine of the National Academies* (2006) (cited on page 1).

[3] E. Priest. *Solar magnetohydrodynamics*. Vol. 21. Springer Science & Business Media, 2012 (cited on page 1).

[4] R. Betti and J.P. Freidberg. 'Radial discontinuities in tokamak magnetohydrodynamic equilibria with poloidal flow'. In: *Physics of Plasmas* 7.6 (2000), pp. 2439–2448 (cited on page 1).

[5] E. Frederick. 'Some effects of electrostatic charges in fabric filtration'. In: *Journal of the Air Pollution Control Association* 24.12 (1974), pp. 1164–1168 (cited on page 1).

[6] F. Ding and A. Kareem. 'Tall buildings with dynamic facade under winds'. In: *Engineering* (2020) (cited on page 1).

[7] D.-Y. Chen et al. 'Suppression of vortex-induced vibrations of a flexible riser by adding helical strakes'. In: *Journal of Hydrodynamics* 31.3 (2019), pp. 622–631 (cited on page 1).

[8] S. Turek and J. Hron. *Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow*. Springer, 2006 (cited on pages 1, 49–51, 53, 54, 56, 59, 60, 70).

[9] M.-C. Hsu, I. Akkerman, and Y. Bazilevs. 'High-performance computing of wind turbine aerodynamics using isogeometric analysis'. In: *Computers & Fluids* 49.1 (2011), pp. 93–100 (cited on page 2).

[10] Y. Bazilevs et al. 'Aerodynamic and FSI analysis of wind turbines with the ALE-VMS and ST-VMS methods'. In: *Archives of Computational Methods in Engineering* 21.4 (2014), pp. 359–398 (cited on pages 2, 3).

[11] A. Korobenko et al. 'FSI simulation of two back-to-back wind turbines in atmospheric boundary layer flow'. In: *Computers & Fluids* 158 (2017), pp. 167–175 (cited on pages 2, 3).

[12] T. Terahara et al. 'Heart valve isogeometric sequentially-coupled FSI analysis with the space–time topology change method'. In: *Computational Mechanics* (2020), pp. 1–21 (cited on page 2).

[13] M. Hsu et al. 'Fluid–structure interaction analysis of bioprosthetic heart valves: significance of arterial wall deformation'. In: *Computational mechanics* 54.4 (2014), pp. 1055–1071 (cited on page 2).

[14] WHO. *Cardiovascular Diseases*. https://www.who.int/health-topics/cardiovascular-diseases accessed October 7, 2020 (cited on page 2).

[15] J. Lengiewicz, M. Wichrowski, and S. Stupkiewicz. 'Mixed formulation and finite element treatment of the mass-conserving cavitation model'. In: *Tribology International* 72 (2014), pp. 143–155 (cited on page 2).

[16] D. Dowson. 'Elastohydrodynamic and micro-elastohydrodynamic lubrication'. In: *Wear* 190.2 (1995), pp. 125–138 (cited on page 2).

[17] M. Ekiel-Jeżewska et al. 'Stokes velocity generated by a point force in various geometries'. In: *The European Physical Journal E* 41.10 (2018), p. 120 (cited on page 2).

[18] M. Bukowicki and M. Ekiel-Jeżewska. 'Sedimenting pairs of elastic microfilaments'. In: *Soft Matter* 15.46 (2019), pp. 9405–9417 (cited on page 2).

[19] S. Pawłowska, T. Kowalewski, and F. Pierini. 'Fibrous polymer nanomaterials for biomedical applications and their transport by fluids: An overview'. In: *Soft Matter* 14.42 (2018), pp. 8421–8444 (cited on page 2).

[20] H.-J. Bungartz and M. Schäfer. *Fluid-structure interaction: modelling, simulation, optimisation*. Vol. 53. Springer Science & Business Media, 2006 (cited on page 2).

[21] S. Basting et al. *Fluid-Structure Interaction: Modeling, Adaptive Discretisations and Solvers*. Vol. 20. Walter de Gruyter GmbH & Co KG, 2017 (cited on page 2).

[22] H.-J. Bungartz M. Mehl M. Schäfer. *Fluid Structure Interaction II: Modelling, Simulation, Optimization*. Vol. 73. Springer-Verlag Berlin Heidelberg, 2010 (cited on page 2).

[23] Y. Bazilevs, K. Takizawa, and T. Tezduyar. *Computational fluid-structure interaction: methods and applications*. John Wiley & Sons, 2013 (cited on pages 2, 3, 17).

[24] T. Richter. *Fluid-structure interactions: models, analysis and finite elements*. Vol. 118. Springer, 2017 (cited on pages 2, 9, 10, 12, 20).

[25] C. Taylor and P. Hood. 'A numerical solution of the Navier-Stokes equations using the finite element technique'. In: *Computers & Fluids* 1.1 (1973), pp. 73–100 (cited on pages 2, 31, 36).

[26] M. Gruca, M. Bukowicki, and M. Ekiel-Jeżewska. 'Periodic and quasiperiodic motions of many particles falling in a viscous fluid'. In: *Physical Review E* 92.2 (2015), p. 023026 (cited on page 2).

[27] Th. Dunne. 'An Eulerian approach to fluid–structure interaction and goal-oriented mesh adaptation'. In: *International Journal for Numerical Methods in Fluids* 51.9-10 (2006), pp. 1017–1039 (cited on pages 2, 11).

[28] R. Glowinski, T.-W. Pan, and J. Periaux. 'A fictitious domain method for external incompressible viscous flow modeled by Navier-Stokes equations'. In: *Computer methods in applied mechanics and engineering* 112.1-4 (1994), pp. 133–148 (cited on page 3).

[29] A. Gerstenberger and W. Wall. 'An extended finite element method/Lagrange multiplier based approach for fluid–structure interaction'. In: *Computer Methods in Applied Mechanics and Engineering* 197.19-20 (2008), pp. 1699–1714 (cited on page 3).

[30] F. Nobile and C. Vergara. 'An effective fluid-structure interaction formulation for vascular dynamics by generalized Robin conditions'. In: *SIAM Journal on Scientific Computing* 30.2 (2008), pp. 731–763 (cited on page 3).

[31] R. Mittal and G. Iaccarino. 'Immersed boundary methods'. In: *Annu. Rev. Fluid Mech.* 37 (2005), pp. 239–261 (cited on page 3).

[32] L. Boilevin-Kayl, M. A. Fernández, and J.F. Gerbeau. 'Numerical methods for immersed FSI with thin-walled structures'. In: *Computers & Fluids* 179 (2019), pp. 744–763 (cited on page 3).

[33] L. Heltai and A. Caiazzo. 'Multiscale modeling of vascularized tissues via nonmatching immersed methods'. In: *International Journal for Numerical Methods in Biomedical Engineering* 35.12 (2019), e3264 (cited on page 3).

[34] E. Hachem et al. 'Immersed stress method for fluid–structure interaction using anisotropic mesh adaptation'. In: *International journal for numerical methods in engineering* 94.9 (2013), pp. 805–825 (cited on page 3).

[35] H. Casquero et al. 'A hybrid variational-collocation immersed method for fluid-structure interaction using unstructured T-splines'. In: *International Journal for Numerical Methods in Engineering* 105.11 (2016), pp. 855–880 (cited on page 3).

[36] M. Uhlmann. 'An immersed boundary method with direct forcing for the simulation of particulate flows'. In: *Journal of Computational Physics* 209.2 (2005), pp. 448–476 (cited on page 3).

[37] T. Richter. 'A Monolithic Geometric Multigrid Solver for Fluid-Structure Interactions in ALE formulation'. In: *International Journal for Numerical Methods in Engineering, doi:10.1002/nme.4943* (2015). DOI: `10.1002/nme.4943` (cited on pages 3, 5, 21).

[38] T. Richter and T. Wick. 'Finite elements for fluid–structure interaction in ALE and fully Eulerian coordinates'. In: *Computer Methods in Applied Mechanics and Engineering* 199.41 (2010), pp. 2633–2642 (cited on pages 3, 11).

[39] M. Souli, A. Ouahsine, and L. Lewin. 'ALE formulation for fluid–structure interaction problems'. In: *Computer methods in applied mechanics and engineering* 190.5-7 (2000), pp. 659–675 (cited on page 3).

[40] J. Donea, S. Giuliani, and J.-P. Halleux. 'An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions'. In: *Computer methods in applied mechanics and engineering* 33.1-3 (1982), pp. 689–723 (cited on page 3).

[41] J. Donea et al. 'Arbitrary L agrangian–E ulerian Methods'. In: *Encyclopedia of Computational Mechanics Second Edition* (2017), pp. 1–23 (cited on page 3).

[42] S.R. Idelsohn, E. Onate, and F. Del Pin. 'A Lagrangian meshless finite element method applied to fluid–structure interaction problems'. In: *Computers & structures* 81.8 (2003), pp. 655–671 (cited on page 3).

[43] K. Takizawa, T. E. Tezduyar, and N. Kostov. 'Sequentially-coupled space–time FSI analysis of bio-inspired flapping-wing aerodynamics of an MAV'. In: *Computational Mechanics* 54.2 (2014), pp. 213–233 (cited on page 3).

[44] P. Moireau et al. 'External tissue support and fluid–structure simulation in blood flows'. In: *Biomechanics and modeling in mechanobiology* 11.1-2 (2012), pp. 1–18 (cited on page 3).

[45] P. Crosetto et al. 'Fluid–structure interaction simulation of aortic blood flow'. In: *Computers & Fluids* 43.1 (2011), pp. 46–57 (cited on page 3).

[46] J.-F. Gerbeau, M. Vidrascu, and P. Frey. 'Fluid–structure interaction in blood flows on geometries based on medical imaging'. In: *Computers & Structures* 83.2 (2005), pp. 155–165 (cited on pages 3, 12).

[47] W. A. Wall and T. Rabczuk. 'Fluid–structure interaction in lower airways of CT-based lung geometries'. In: *International Journal for Numerical Methods in Fluids* 57.5 (2008), pp. 653–675 (cited on pages 3, 12).

[48] A. Korobenko et al. 'Recent advances in ALE-VMS and ST-VMS computational aerodynamic and FSI analysis of wind turbines'. In: *Frontiers in Computational Fluid-Structure Interaction and Flow Simulation*. Springer, 2018, pp. 253–336 (cited on page 3).

[49] J. Yan et al. 'Computational free-surface fluid–structure interaction with application to floating offshore wind turbines'. In: *Computers & Fluids* 141 (2016), pp. 155–174 (cited on page 3).

[50] B. Helenbrook. 'Mesh deformation using the biharmonic operator'. In: *International journal for numerical methods in engineering* 56.7 (2003), pp. 1007–1021 (cited on pages 3, 12).

[51] M. Selim, R. Koomullil, et al. 'Mesh deformation approaches–a survey'. In: *Journal of Physical Mathematics* 7.2 (2016) (cited on page 3).

[52] T. Wick. 'Variational-monolithic ale fluid-structure interaction: Comparison of computational cost and mesh regularity using different mesh motion techniques'. In: *Modeling, Simulation and Optimization of Complex Processes HPSC 2015*. Springer, 2017, pp. 261–275 (cited on page 3).

[53] P. Causin, J.-F. Gerbeau, and F. Nobile. 'Added-mass effect in the design of partitioned algorithms for fluid–structure problems'. In: *Computer methods in applied mechanics and engineering* 194.42 (2005), pp. 4506–4527 (cited on page 4).

[54] M. Á. Fernández and M. Moubachir. 'A Newton method using exact Jacobians for solving fluid–structure coupling'. In: *Computers & Structures* 83.2 (2005), pp. 127–142 (cited on page 4).

[55] P. Crosetto et al. 'Parallel algorithms for fluid-structure interaction problems in haemodynamics'. In: *SIAM Journal on Scientific Computing* 33.4 (2011), pp. 1598–1622 (cited on pages 4, 20, 21).

[56] C. Murea and S. Sy. 'Updated Lagrangian/Arbitrary Lagrangian–Eulerian framework for interaction between a compressible neo-Hookean structure and an incompressible fluid'. In: *International Journal for Numerical Methods in Engineering* 109.8 (2017), pp. 1067–1084 (cited on pages 4, 6, 17, 23).

[57] S. Badia, A. Quaini, and A. Quarteroni. 'Modular vs. non-modular preconditioners for fluid–structure systems with large added-mass effect'. In: *Computer Methods in Applied Mechanics and Engineering* 197.49-50 (2008), pp. 4216–4232 (cited on pages 4, 21).

[58] A. Lozovskiy et al. 'An unconditionally stable semi-implicit FSI finite element method'. In: *Computer Methods in Applied Mechanics and Engineering* 297 (2015), pp. 437–454 (cited on pages 4, 6, 21).

[59] J. Degroote, K. Bathe, and J. Vierendeels. ‘Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction’. In: *Computers & Structures* 87.11-12 (2009), pp. 793–801 (cited on page 4).

[60] M. Heil, A. L. Hazel, and J. Boyle. ‘Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches’. In: *Computational Mechanics* 43.1 (2008), pp. 91–101 (cited on pages 4, 21).

[61] U. Langer and H. Yang. ‘Numerical simulation of fluid–structure interaction problems with hyperelastic models: A monolithic approach’. In: *Mathematics and Computers in Simulation* 145 (2018), pp. 186–208 (cited on pages 4, 6, 21).

[62] J. Xu and Y. Zhu. ‘Uniform convergent multigrid methods for elliptic problems with strongly discontinuous coefficients’. In: *Mathematical Models and Methods in Applied Sciences* 18.01 (2008), pp. 77–105 (cited on pages 4, 30, 31, 34, 39).

[63] J. Mandel and M. Brezina. ‘Balancing domain decomposition for problems with large jumps in coefficients’. In: *Mathematics of Computation of the American Mathematical Society* 65.216 (1996), pp. 1387–1401 (cited on page 4).

[64] B. Janssen and Th Wick. ‘Block preconditioning with Schur complements for monolithic fluid-structure interactions’. In: *ECCOMAS CFD*. 2010 (cited on page 4).

[65] S. Badia, A. Quaini, and A. Quarteroni. ‘Splitting methods based on algebraic factorization for fluid-structure interaction’. In: *SIAM Journal on Scientific Computing* 30.4 (2008), pp. 1778–1805 (cited on page 4).

[66] J. Xu and K. Yang. ‘Well-posedness and robust preconditioners for discretized fluid–structure interaction systems’. In: *Computer Methods in Applied Mechanics and Engineering* 292 (2015), pp. 69–91 (cited on pages 4–6, 17, 20, 21, 23).

[67] M. Benzi, M. A. Olshanskii, and Z. Wang. ‘Modified augmented Lagrangian preconditioners for the incompressible Navier–Stokes equations’. In: *International Journal for Numerical Methods in Fluids* 66.4 (2011), pp. 486–508 (cited on page 4).

[68] Y. Wu and X-Ch Cai. ‘A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation’. In: *Journal of Computational Physics* 258 (2014), pp. 524–537 (cited on page 4).

[69] M. W. Gee, U. Küttler, and W. A. Wall. ‘Truly monolithic algebraic multigrid for fluid–structure interaction’. In: *International Journal for Numerical Methods in Engineering* 85.8 (2011), pp. 987–1016 (cited on page 5).

[70] U. Langer and H. Yang. ‘Recent development of robust monolithic fluid-structure interaction solvers’. In: *Fluid-Structure Interactions. Modeling, Adaptive Discretization and Solvers, Radon Series on Computational and Applied Mathematics* 20 (2016) (cited on page 5).

[71] K. Yang et al. ‘Modeling and Numerical Studies for Fluid-Structure Interaction Involving an Elastic Rotor’. In: *Computer Methods in Applied Mechanics and Engineering* 311 (2016), pp. 788–814 (cited on pages 5, 12).

[72] M. Kronbichler and Kormann K. ‘A generic interface for parallel cell-based finite element operator application’. In: *Computers & Fluids* 63 (2012), pp. 135–147. DOI: `https://doi.org/10.1016/j.compfluid.2012.04.012` (cited on pages 5, 34–36, 44).

[73] A. Arndt et al. ‘The `deal.II` Library, Version 9.2’. In: *Journal of Numerical Mathematics* 28.3 (2020), pp. 131–146. DOI: `10.1515/jnma-2020-0043` (cited on pages 5, 6, 35, 36, 44, 49, 57).

[74] M. Adams et al. ‘Parallel multigrid smoothing: polynomial versus Gauss–Seidel’. In: *Journal of Computational Physics* 188.2 (2003), pp. 593–610. DOI: `https://doi.org/10.1016/S0021-9991(03)00194-3` (cited on pages 5, 30, 34).

[75] J. Brannick et al. ‘Local Fourier analysis of multigrid methods with polynomial smoothers and aggressive coarsening’. In: *arXiv preprint arXiv:1310.8385* (2013) (cited on page 5).

[76] O. F. Oxtoby and A. Malan. 'A matrix-free, implicit, incompressible fractional-step algorithm for fluid–structure interaction applications'. In: *Journal of Computational Physics* 231.16 (2012), pp. 5389–5405 (cited on pages 5, 6).

[77] X. Lv et al. 'A matrix-free implicit unstructured multigrid finite volume method for simulating structural dynamics and fluid–structure interaction'. In: *Journal of Computational Physics* 225.1 (2007), pp. 120–144 (cited on pages 5, 6).

[78] T. He, T. Wang, and H. Zhang. 'The use of artificial compressibility to improve partitioned semi-implicit FSI coupling within the classical Chorin–Témam projection framework'. In: *Computers & Fluids* 166 (2018), pp. 64–77 (cited on pages 5, 6).

[79] J.R. Cash. 'On the integration of stiff systems of ODEs using extended backward differentiation formulae'. In: *Numerische Mathematik* 34.3 (1980), pp. 235–246 (cited on pages 6, 20).

[80] A. Arndt et al. 'The deal.II finite element library: Design, features, and insights'. In: *Computers & Mathematics with Applications* 81 (2021), pp. 407–422. DOI: `10.1016/j.camwa.2020.02.022` (cited on page 6).

[81] T. Wick. 'Solving monolithic fluid-structure interaction problems in arbitrary Lagrangian Eulerian coordinates with the deal. ii library'. In: *Archive of Numerical Software* 1.1 (2013), pp. 1–19 (cited on pages 6, 17).

[82] D. Braess and R. Sarazin. 'An efficient smoother for the Stokes problem'. In: *Applied Numerical Mathematics* 23.1 (1997), pp. 3–19 (cited on pages 7, 28, 30, 32, 33, 37, 69).

[83] W. Zulehner. 'A class of smoothers for saddle point problems'. In: *Computing* 65.3 (2000), pp. 227–246 (cited on pages 7, 28, 30, 32, 33, 35, 37–39, 46, 69).

[84] D. Acheson. *Elementary fluid dynamics*. 1991 (cited on page 9).

[85] T. Wick. 'Flapping and contact FSI computations with the fluid–solid interface-tracking/interface-capturing technique and mesh adaptivity'. In: *Computational Mechanics* 53.1 (2014), pp. 29–43 (cited on page 11).

[86] T. Wick. 'Fully Eulerian fluid–structure interaction for time-dependent problems'. In: *Computer Methods in Applied Mechanics and Engineering* 255 (2013), pp. 14–26 (cited on page 11).

[87] T. Richter. 'A fully Eulerian formulation for fluid–structure-interaction problems'. In: *Journal of Computational Physics* 233 (2013), pp. 227–240 (cited on page 11).

[88] R. W. Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997 (cited on page 14).

[89] R. W. Ogden. 'Large deformation isotropic elasticity-on the correlation of theory and experiment for incompressible rubberlike solids'. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 326. The Royal Society. 1972, pp. 565–584 (cited on page 14).

[90] G. Dahlquist. 'A special stability problem for linear multistep methods'. In: *BIT Numerical Mathematics* 3.1 (1963), pp. 27–43 (cited on page 20).

[91] J.J.I.M. van Kan. 'A second-order accurate pressure-correction scheme for viscous incompressible flow'. In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 870–891 (cited on page 23).

[92] S. Turek. 'A comparative study of time-stepping techniques for the incompressible Navier-Stokes equations: from fully implicit non-linear schemes to semi-implicit projection methods'. In: *International Journal for Numerical Methods in Fluids* 22.10 (1996), pp. 987–1011 (cited on page 23).

[93] S. Dong and J. Shen. 'An unconditionally stable rotational velocity-correction scheme for incompressible flows'. In: *Journal of Computational Physics* 229.19 (2010), pp. 7013–7029 (cited on page 23).

[94] I. Babuška and R. Narasimhan. 'The Babuška-Brezzi condition and the patch test: an example'. In: *Computer methods in applied mechanics and engineering* 140.1-2 (1997), pp. 183–199 (cited on page 25).

[95] A. N. Brooks and T. Hughes. 'Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations'. In: *Computer methods in applied mechanics and engineering* 32.1-3 (1982), pp. 199–259 (cited on pages 26, 27).

[96] V. John and P. Knobloch. 'On discontinuity—capturing methods for convection—diffusion equations'. In: *Numerical mathematics and advanced applications*. Springer, 2006, pp. 336–344 (cited on page 27).

[97] M. Kronbichler, T. Heister, and W. Bangerth. 'High accuracy mantle convection simulation through modern numerical methods'. In: *Geophysical Journal International* 191.1 (2012), pp. 12–29 (cited on page 29).

[98] J. Rudi et al. 'An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle'. In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 2015, pp. 1–12 (cited on page 29).

[99] E. Olsson and G. Kreiss. 'A conservative level set method for two phase flow'. In: *Journal of Computational Physics* 210.1 (2005), pp. 225–246 (cited on page 29).

[100] H. A. Barnes. 'The yield stress—a review or '$\pi\alpha\nu\tau\alpha\rho\epsilon\iota$'—everything flows?' In: *Journal of Non-Newtonian Fluid Mechanics* 81.1 (1999), pp. 133–178 (cited on page 29).

[101] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Vol. 15. Springer Science & Business Media, 2012 (cited on pages 29, 31).

[102] D. Silvester and A. Wathen. 'Fast iterative solution of stabilised Stokes systems part II: using general block preconditioners'. In: *SIAM Journal on Numerical Analysis* 31.5 (1994), pp. 1352–1367 (cited on page 29).

[103] P. Krzyzanowski. 'Block preconditioners for saddle point problems resulting from discretizations of partial differential equations'. In: *Efficient Preconditioned Solution Methods for Elliptic Partial Differential Equations* (2011), p. 44 (cited on page 29).

[104] K.-A. Mardal and R. Winther. 'Preconditioning discretizations of systems of partial differential equations'. In: *Numerical Linear Algebra with Applications* 18.1 (2011), pp. 1–40 (cited on page 29).

[105] D. Drzisga et al. 'On the analysis of block smoothers for saddle point problems'. In: *SIAM Journal on Matrix Analysis and Applications* 39.2 (2018), pp. 932–960 (cited on page 29).

[106] A. Toselli and O. Widlund. *Domain decomposition methods-algorithms and theory*. Vol. 34. Springer Science & Business Media, 2006 (cited on page 29).

[107] W. Hackbusch. *Multi-grid methods and applications*. Vol. 4. Springer Science & Business Media, 2013 (cited on page 29).

[108] M. A. Olshanskii and A. Reusken. 'Analysis of a Stokes interface problem'. In: *Numerische Mathematik* 103.1 (2006), pp. 129–149 (cited on pages 29, 46).

[109] M. A. Olshanskii, J. Peters, and A. Reusken. 'Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations'. In: *Numerische Mathematik* 105.1 (2006), pp. 159–191 (cited on page 29).

[110] B. Aksoylu and Z. Unlu. 'Robust preconditioners for the high-contrast Stokes equation'. In: *Journal of Computational and Applied Mathematics* 259 (2014), pp. 944–954 (cited on pages 29, 46).

[111] J. Rudi, G. Stadler, and O. Ghattas. 'Weighted BFBT preconditioner for Stokes flow problems with highly heterogeneous viscosity'. In: *SIAM Journal on Scientific Computing* 39.5 (2017), S272–S297 (cited on pages 29, 30, 44, 45, 47).

[112] D. May, J. Brown, and L. Le Pourhiet. 'A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow'. In: *Computer methods in applied mechanics and engineering* 290 (2015), pp. 496–523 (cited on pages 30, 44).

[113] T. Clevenger and T. Heister. 'Comparison Between Algebraic and Matrix-free Geometric Multigrid for a Stokes Problem on Adaptive Meshes with Variable Viscosity'. In: *arXiv preprint arXiv:1907.06696* (2019) (cited on pages 30, 44, 45, 47).

[114] Q. Hong et al. 'A robust multigrid method for discontinuous Galerkin discretizations of Stokes and linear elasticity equations'. In: *Numerische Mathematik* 132.1 (2016), pp. 23–49 (cited on pages 30, 69).

[115] J. Schöberl. 'Multigrid methods for a parameter dependent problem in primal variables'. In: *Numerische Mathematik* 84.1 (1999), pp. 97–119 (cited on page 30).

[116] Ch. Wieners. 'Robust multigrid methods for nearly incompressible elasticity'. In: *Computing* 64.4 (2000), pp. 289–306 (cited on page 30).

[117] Ch. Long. 'Multigrid methods for saddle point systems using constrained smoothers'. In: *Computers & Mathematics with Applications* 70.12 (2015), pp. 2854–2866 (cited on pages 30, 31).

[118] M. Olshanskii. 'Multigrid analysis for the time dependent Stokes problem'. In: *Mathematics of Computation* 81.277 (2012), pp. 57–79 (cited on page 30).

[119] S. Brenner, H. Li, and L.-Y. Sung. 'Multigrid methods for saddle point problems: Stokes and Lamé systems'. In: *Numerische Mathematik* 128.2 (2014), pp. 193–216 (cited on page 30).

[120] D. Borzacchiello et al. 'Box-relaxation based multigrid solvers for the variable viscosity Stokes problem'. In: *Computers & Fluids* 156 (2017), pp. 515–525 (cited on pages 30, 46, 47).

[121] S. P. Vanka. 'Block-implicit multigrid solution of Navier-Stokes equations in primitive variables'. In: *Journal of Computational Physics* 65.1 (1986), pp. 138–158 (cited on page 30).

[122] S. Bauer et al. 'Large-scale simulation of mantle convection based on a new matrix-free approach'. In: *Journal of Computational Science* 31 (2019), pp. 60–76 (cited on page 30).

[123] B. Gmeiner et al. 'A quantitative performance study for Stokes solvers at the extreme scale'. In: *Journal of Computational Science* 17 (2016), pp. 509–521 (cited on page 30).

[124] Y. Saad and M. H. Schultz. 'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems'. In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869 (cited on pages 30, 31).

[125] Y. Saad. 'A flexible inner-outer preconditioned GMRES algorithm'. In: *SIAM Journal on Scientific Computing* 14.2 (1993), pp. 461–469 (cited on pages 30, 31).

[126] R. S. Varga. *Matrix iterative analysis*. Vol. 27. Springer Science & Business Media, 2009 (cited on page 34).

[127] R. B. Lehoucq, D. C. Sorensen, and Ch. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Vol. 6. Siam, 1998 (cited on page 41).

[128] P. Amestoy et al. 'MUMPS: a general purpose distributed memory sparse solver'. In: *International Workshop on Applied Parallel Computing*. 2000, pp. 121–130 (cited on pages 45, 49).

[129] M. Heroux et al. 'An overview of the Trilinos project'. In: *ACM Transactions on Mathematical Software (TOMS)* 31.3 (2005), pp. 397–423 (cited on pages 49, 56).

[130] M. Schäfer et al. 'Benchmark computations of laminar flow around a cylinder'. In: *Flow simulation with high-performance computers II*. Springer, 1996, pp. 547–566 (cited on page 50).

[131] H. A. van der Vorst. 'Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems'. In: *SIAM Journal on scientific and Statistical Computing* 13.2 (1992), pp. 631–644 (cited on page 51).

[132] S. Brenner and L. Y. Sung. 'C 0 interior penalty methods for fourth order elliptic boundary value problems on polygonal domains'. In: *Journal of Scientific Computing* 22.1-3 (2005), pp. 83–118 (cited on page 55).

[133] X. Zhang. 'Multilevel Schwarz methods for the biharmonic Dirichlet problem'. In: *SIAM Journal on Scientific Computing* 15.3 (1994), pp. 621–644 (cited on page 56).

[134] O.C. Zienkiewicz et al. 'On discontinuous Galerkin methods'. In: *International journal for numerical methods in engineering* 58.8 (2003), pp. 1119–1148 (cited on page 69).

[135] W. Regulski et al. 'Pressure drop in flow across ceramic foams—A numerical and experimental study'. In: *Chemical Engineering Science* 137 (2015), pp. 320–337 (cited on page 69).

Department of Mechanics of Materials
Institute of Fundamental Technological Research
Polish Academy of Sciences
Warsaw, Poland

Fluid-structure interaction problems: velocity-based formulation and monolithic computational methods

Michał Wichrowski

Supervisor:
Prof. Stanisław Stupkiewicz
Co-supervisor:
Dr. Piotr Krzyżanowski

Abstract

The solution of fluid-structure interaction (FSI) problems is required in vast applications, ranging from the study of micro-scale biomechanics to design of offshore platforms. The complexity of phenomena requires accurate computational methods, resulting in a growing problem size. The resulting problems may involve billions of unknowns which can only be handled on massively parallel, distributed memory supercomputers. For handling such large-scale problems special algorithms are required.
In this work, we develop a new method of solving time-dependent FSI problems using the Finite Element Method in Arbitrary Lagrangian-Eulerian frame of reference. We derive the monolithic predictor-corrector time integration scheme by adopting the Geometry-Convective Explicit scheme for the problem involving interaction between incompressible hyperelastic solid and incompressible fluid. To improve conservation of the volume of solid we modify the mass conservation equation by introducing volumetric damping. The proposed algorithm consists of several sub-steps at each time step. Among them, the most time-consuming is the solution of the generalized Stokes problem with discontinuous variable coefficients. This has to be done one or two times per each time step, depending on the variant of the predictor-corrector scheme. We introduce a new multilevel preconditioner, that is robust with respect to both problem size and coefficient jumps. We test our implementation on the Turek-Hron benchmark problem. The design of all building blocks of the solver is matrix-free, allowing both speed and memory optimizations. The implementation, based on `deal.II` library supports parallel computations, was tested on problem problems involving over 200 million unknowns.

Keywords: fluid-structure interaction, FSI, multigrid, matrix-free, monolithic, ALE, HPC.

Zakład Mechaniki Materiałów
Instyt Podstawowych Problemów Techniki
Polskiej Akademi Nauk
Warszawa, Polska

Zagadnienia oddziaływania płyn-ciało stałe: sformułowanie przędkościowe i monolityczne metody obliczeniowe

Michał Wichrowski

Promotor:
prof. Stanisław Stupkiewicz
Promotor pomocniczy:
dr Piotr Krzyżanowski

Streszczenie

Rozwiązanie zagadnień oddziaływania płyn-ciało stałe (ang. Fluid-Structure Interaction, FSI) jest wymagane w rozległym spektrum zastosowań, począwszy od badań biomechaniki w nanoskali po projektowanie platform wiertniczych. Złożoność tych zjawisk wymaga dokładnych metod obliczeniowych, co wiąże się z rosnącym rozmiarem zadania. Nie są rzadkością zadania z milionami czy nawet miliardami niewiadomych, w przypadku których samo rozwiązanie jest zbyt duże, by je przechowywać w pamięci pojedynczego komputera. Do obsługi takich zadań na dużą skalę potrzebne są specjalne algorytmy, które można zaimplementować na maszynach masowo równoległych.
W tej pracy rozwijamy nową metodę rozwiązywania zależnych od czasu zagadnień FSI sformułowanych w arbitralnym układzie odniesienia (ALE). Monolityczny schemat całkowania po czasie wyprowadzamy przez adaptację schematu GCE(geometry-convective explicit) dla zagadnienia interakcji między nieściśliwym hipersprężystym ciałem stałym i nieściśliwym płynem. Aby zapewnić stałą objętość ciała stałego, modyfikujemy równanie zachowania masy, wprowadzając tłumienie objętościowe. Zadanie na każdym kroku czasowym rozwiązywane jest za pomocą metody elementów skończonych. Zaproponowany algorytm w każdym kroku czasowym wymaga jednego lub dwóch (w zależności od wariantu) rozwiązań uogólnionego zagadnienia Stokesa o silnie zmiennych współczynnikach. Następnie wprowadzamy nowy wielopoziomowy operator ściskający (preconditioner), który zapewnia zbieżność niezależnie od rozmiaru zadania, jak i skoków współczynników. Testujemy naszą implementację na przykładzie zaproponowanym przez Turka. Konstrukcja wszystkich elementów składowych metody nie wymaga przechowywania macierzy (matrix-free), co umożliwia optymalizację czasu jak i użycia pamięci. Implementacja, oparta na bibliotece `deal.II`, umożliwia obliczenia równoległe i rozwiązywanie zadań z ponad 200 milionami niewiadomych.

Słowa kluczowe: Oddziaływanie płyn-ciało stałe, FSI,multigrid, metoda wielosiatkowa, matrix-free, monolityczny schemat obliczeniowy, ALE, HPC

Original contribution of the thesis

- A new monolithic matrix-free FSI solver was developed
- The solver was implemented within the framework of the finite element library `deal.II`, exploiting its ability to handle parallel computations
- Geometry-Convective Explicit scheme was extended to include incompressible hyperelastic Mooney-Rivlin solid
- A new preconditioner for the generalized Stokes problem with discontinuous coefficients was proposed
- A volumetric damping was introduced in the solid part to improve conservation of volume
- A direct connection between the generalized Stokes problem with discontinuous coefficients and the FSI problem was shown

Abstract

The solution of fluid-structure interaction (FSI) problems is required in vast applications, ranging from the study of micro-scale biomechanics to design of offshore platforms. The complexity of phenomena requires accurate computational methods, resulting in a growing problem size. The resulting problems may involve billions of unknowns which can only be handled on massively parallel, distributed memory supercomputers. For handling such large-scale problems special algorithms are required.
In this work, we develop a new method of solving time-dependent FSI problems using the Finite Element Method in Arbitrary Lagrangian-Eulerian frame of reference. We derive the monolithic predictor-corrector time integration scheme by adopting the Geometry-Convective Explicit scheme for the problem involving interaction between incompressible hyperelastic solid and incompressible fluid. To improve conservation of the volume of solid we modify the mass conservation equation by introducing volumetric damping. The proposed algorithm consists of several sub-steps at each time step. Among them, the most time-consuming is the solution of the generalized Stokes problem with discontinuous variable coefficients. This has to be done one or two times per each time step, depending on the variant of the predictor-corrector scheme. We introduce a new multilevel preconditioner, that is robust with respect to both problem size and coefficient jumps. We test our implementation on the Turek-Hron benchmark problem. The design of all building blocks of the solver is matrix-free, allowing both speed and memory optimizations. The implementation, based on `deal.II` library supports parallel computations, was tested on problem problems involving over 200 million unknowns.