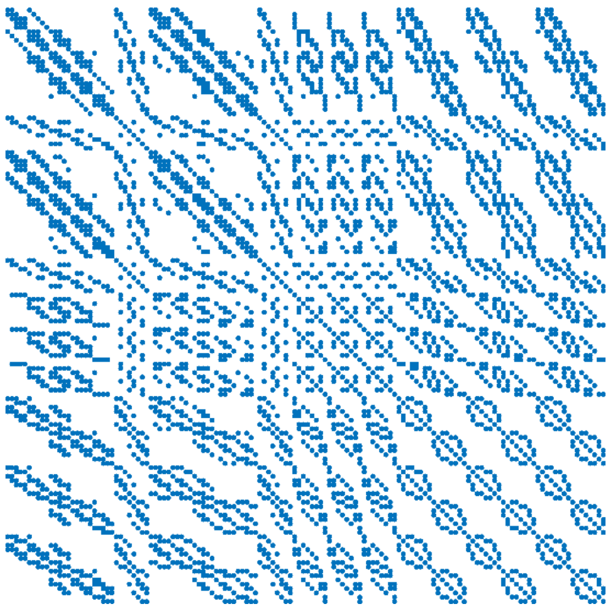


Parallel Overlapping Schwarz Preconditioners and Multiscale Discretizations with Applications to Fluid-Structure Interaction and Highly Heterogeneous Problems

INAUGURAL-DISSERTATION

zur
Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Alexander Heinlein
aus Essen

Köln, 2016

Berichterstatter: Prof. Dr. Axel Klawonn
(Universität zu Köln)

Prof. Dr. Oliver Rheinbach
(TU Bergakademie Freiberg)

Tag der mündlichen Prüfung: 22. April 2016

Abstract

Accurate simulations of transmural wall stresses in artherosclerotic coronary arteries may help to predict plaque rupture. Therefore, a robust and efficient numerical framework for Fluid-Structure Interaction (FSI) of the blood flow and the arterial wall has to be set up, and suitable material laws for the modeling of the fluid and the structural response have to be incorporated. In this thesis, monolithic coupling algorithms and corresponding monolithic preconditioners are used to simulate FSI using highly nonlinear anisotropic polyconvex hyperelastic and anisotropic viscoelastic material models for the arterial wall. An MPI-parallel FSI software from the LifeV library is coupled to the software FEAP in order to enable access to the structural material models implemented in FEAP. To define a benchmark test for highly nonlinear material models in FSI, a simple geometry corresponding to a section of an idealized coronary artery, suitable boundary conditions, and material parameters adapted to experimental data are used. In particular, the geometry is chosen to be non-symmetric to make effects due to the anisotropy of the structure visible. An initialization phase and several heartbeats are simulated, and systematical studies with meshes of increasing refinement and different space discretizations are carried out. The results indicate that, for the highly nonlinear material models, piecewise quadratic or F-bar element discretizations lead to significantly better results than piecewise linear shape functions. The results using piecewise linear shape functions are less accurate with respect to the displacements and, in particular, to the approximation of the stresses.

To improve the performance of the FSI simulations, a more robust preconditioner for the highly nonlinear structural material models has to be used. Therefore, a parallel implementation of the GDSW (Generalized Dryja-Smith-Widlund) preconditioner, which is a geometric two-level overlapping Schwarz preconditioner with energy-minimizing coarse space, is presented. The implementation, which is based on the software library Trilinos, is held flexible to make further extensions of the preconditioner possible. Even though the dimension of its coarse space is comparably large, parallel scalability for two and three dimensional scalar elliptic and linear elastic problems for thousands of cores is demonstrated. Also for unstructured domain decompositions and for a hybrid version of the preconditioner, convincing scalability is presented. When used as a preconditioner for the structure block in FSI simulations, the GDSW preconditioner shows excellent performance as well: scalability for up to 512 cores and a significant reduction of the simulation time and of the number of iterations with respect to the previously used preconditioner, IFPACK, are observed. IFPACK is an algebraic one-level overlapping Schwarz preconditioner.

Finally, highly heterogeneous (multiscale) problems are investigated. Since the GDSW coarse space is not robust for general problems of this type, spaces based on Approximate Component Mode Synthesis (ACMS) are considered. On the basis of the ACMS space, coarse spaces for overlapping Schwarz methods are constructed, and a parallel implementation of a special finite element method is presented. For the coarse spaces, preliminary results indicating numerical scalability and robustness are discussed. For the parallel implementation of the special finite element method, very good parallel weak scalability is observed with respect to the construction of the basis functions and to the solution of the resulting linear system using the FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) method.

Zusammenfassung

Präzise Simulationen der Wandspannungsverteilung in artherosklerotischen Koronararterien können ein Hilfsmittel zur Vorhersage von Rupturen des Plaques sein. Um diese zu ermöglichen, müssen die Fluid-Struktur Interaktion (FSI) von Blutfluss und Arterienwand sowie ein realistisches Materialverhalten von Blut und Arterienwand berücksichtigt werden. Dazu werden robuste, effiziente numerische Verfahren und geeignete Materialmodelle für Fluid und Struktur benötigt. In dieser Arbeit werden monolithische Kopplungsalgorithmen und Vorkonditionierer verwendet, um FSI unter Verwendung von stark nichtlinearen anisotropen polykonvexen hyperelastischen und anisotropen viskoelastischen Materialmodellen für die Arterienwand zu simulieren. Die zu diesem Zweck implementierte Software basiert auf der Kopplung eines MPI-parallelen FSI Codes aus der Softwarebibliothek LifeV mit der Strukturmechanik-Simulationssoftware FEAP. Dies ermöglicht die Verwendung aller Materialmodelle aus FEAP. Um einen Benchmark-Test für die Simulation von FSI unter der Verwendung von stark nichtlinearen Materialmodellen zu definieren, werden die Geometrie einer vereinfachten Koronararterie, geeignete Randbedingungen und realistische Materialparameter verwendet. Die Geometrie ist dabei asymmetrisch gewählt, damit Effekte infolge der Anisotropie des Strukturmaterials erkennbar sind. Die Simulationen beinhalten jeweils eine Initialisierungsphase und mehrere Herzschläge. Dabei werden Gitter unterschiedlicher Feinheit und verschiedene Raumdiskretisierungen verwendet. Die Ergebnisse der Simulationen zeigen unter anderem, dass stückweise lineare Formfunktionen nicht zur Diskretisierung der stark nichtlinearen Modelle geeignet sind. Qualitativ hochwertiger sind die Ergebnisse für stückweise quadratische und F-bar Elemente, vor allem bezüglich der Verschiebungen und insbesondere der Approximation der Spannungen.

Um die Performance der FSI Simulationen zu verbessern, muss ein robusterer Vorkonditionierer für die stark nichtlinearen Strukturmodelle verwendet werden. Daher wird eine parallele Implementierung des zweistufigen überlappenden Schwarz Vorkonditionierers GDSW (Generalized Dryja-Smith-Widlund), welcher geometrische Überlappung und einen energieminimalen Grobgitterraum verwendet, vorgestellt. Die Implementierung basiert auf der Softwarebibliothek Trilinos und ist flexibel gehalten, um weitere Verbesserungen zu ermöglichen. Trotz der hohen Dimension des Grobgitterraums kann parallele Skalierbarkeit für zwei und dreidimensionale skalare und linear elastische Probleme bis zu mehreren Tausend Rechenkernen gezeigt werden. Auch die Skalierbarkeit für unstruktutrierte Zerlegungen und für eine hybride Variante des GDSW Vorkonditionierers ist ausgezeichnet. Der GDSW Vorkonditionierer wird daher auch für den Strukturblock in FSI verwendet, mit exzellenten Ergebnissen: Starke Skalierbarkeit für bis zu 512 Rechenkernen und eine beachtliche Reduktion der Simulationszeit gegenüber dem zuvor verwendeten Vorkonditionierer, IFPACK, werden erreicht. IFPACK ist ein einstufiger überlappender Schwarz Vorkonditionierer mit algebraischer Überlappung.

Abschließend werden stark heterogene (Multiskalen-) Probleme behandelt. Der GDSW Vorkonditionierer ist, im Allgemeinen, nicht robust für Probleme dieser Art. Daher werden Grobgitterräume und eine parallele Implementierung einer speziellen Finite Elemente Methode, die auf Approximate Component Mode Synthesis (ACMS) basieren, untersucht. Bezüglich der Grobgitterräume für überlappende Schwarz Vorkonditionierer können erste vielversprechende Ergebnisse gezeigt werden. Für die parallele Implementierung der speziellen Finite Elemente Methode wird sehr gute parallele Skalierbarkeit für die Berechnung der Finite Elemente Funktionen und die Lösung mit Hilfe der FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) Methode präsentiert.

Acknowledgements

First of all, I would like to thank my advisor Axel Klawonn for the great opportunity to work on some most interesting topics during the last four years in his group. Even though, these years were filled with a lot of hard work and each of my projects required a lot of initial work, I look back at a very exciting time. I have learned many new things, and I had the chance to develop myself in many ways. This would not have been possible without the support and, especially in the last year, the encouragement of Axel Klawonn.

This thesis would also not have been possible without the help and support of Oliver Rheinbach. Even though having moved to Freiberg, he was still available for hours of fruitful discussions on the phone and during my stays in Freiberg. I am also thankful for sharing his knowledge on the implementation of parallel algorithms with me. In particular, I would like to thank him for the use of his FETI-DP implementation and his efforts in working on our publications.

Many thanks are also directed to all my former and current colleagues, namely, Andreas Fischle, Sabrina Gippert, Patrick Radtke, Martin Lanser, Martin Kühn, Stephanie Friedhoff, and Christian Hochmuth. I thank all of them, for sharing many, many hours in discussions about mathematical, numerical, and other important topics. I am grateful for their support, and glad about their understanding for my habits (especially, for not being communicative when trying to focus on implementation or writing). In particular, I appreciated the time being office mates with Andreas Fischle, helping und supporting me especially at the beginning of my PhD, Martin Lanser (twice actually) for helping out each other with all kinds of programming issues and sharing a lot of frustration while implementing C and C++ codes, and Patrick Radtke, for being a very calm and careful person. I would also like to thank Martin Kühn for being a good friend, especially outside the office, and Jascha Knepper, for our recent collaboration which continued the work of the first phase of my PhD.

Furthermore, I acknowledge all of my coworkers from the "CoDELartere" project for the collaboration and the results which would have not been possible without them, namely, Daniel Balzani, Simone Deparis, Simon Fausten, Davide Forti, and Jörg Schröder. Regarding our project, also the financial support by the German Science Foundation (DFG), project no. BA2823/9-1, KL2094/3-1, RH122/4-1, and SCHR570/15-1, and the Swiss National Foundation (SNF), project no. 140184 is gratefully appreciated.

I also thank Ulrich Hetmaniuk from the University of Washington for the chance to spend three months at the Department of Applied Mathematics at the University of Washington at the beginning of my PhD. I appreciate my stay in Seattle a lot, including the discussions with Ulrich and the chance to learn about the different culture in the U.S., and I feel bad for not having been able to come back during the time of my PhD.

Laying the ground for the implementations in my PhD thesis, I would also like to thank the `PETSc`, `Trilinos`, and `LifeV` teams. Many computers have been used to perform all the numerical experiments presented in this thesis (and many more). Therefore, I gratefully acknowledge the use of the CHEOPS parallel computer at Universität zu Köln, the Cray XT6m at the University of Duisburg-Essen, the Cluster at Technische Universität Bergakademie Freiberg, and the Oculus cluster at Universität Paderborn. I also gratefully acknowledge the Gauss Centre for Supercomputing (GCS) for providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS share of the supercomputer JUQUEEN [191] at Jülich Supercomputing Centre (JSC). GCS is the alliance of the three national supercomputing centres HLRS (Universität Stuttgart), JSC (Forschungszentrum Jülich), and LRZ (Bayerische Akademie der Wissenschaften), funded by the German Federal Ministry of Education and Research (BMBF) and the German State Ministries for Research of Baden-Württemberg (MWK), Bayern (StMWFK) and Nordrhein-Westfalen (MIWF).

Many other people who were not involved in the scientific part of my PhD made the completion of this thesis possible. First, I would like to thank my family: my parents, Renate and Hans, who always supported me in every possible way, and all my grandparents, Hanna, Hubert and Amalie, who sadly passed away and thus could not witness the completion of my thesis, and my grandfather Albert. All of them, in their own way, helped to finish this work by their support and encouragement.

Second, I thank my girlfriend Erika, for being encouraging und understanding, and for overlooking all the time I spent on my work instead of with her. I also thank her family and friends.

Third, I would like to thank all of my friends for being a distraction from my work and a great help in many difficult situations, and I am very glad about the fun time we spent together. It is great that they are still part of my life, even though, I could not spend much time with them during the last years. Therefore, I would like to mention some of them here: my best and oldest friend, Sebastian; my friends from high school, Andreas and Long; my friends from the studying time at Universität Duisburg-Essen; my old swimming team; my friends from Essen, Esther, Delia, and Marina; Mira, for many conversations on the phone; and my friends from the FH Südwestfalen, Jathessan and Damian, for several

Skype sessions. I would also like to thank those people I met in Seattle and in Cologne, for making life enjoyable after moving. In particular, many thanks go to Aymeric, Yerin, Rita, Francesco, Lynn, Susanna, Soo, Sandra, Cat, Sophia, Randy, Ben, Michelle and Thanh, Chengchen, Irina, Insa, Denise, Johanna, Stefan, and Maike.

Finally, I would like to thank especially those people who proofread this thesis within such a narrow time frame.

Contents

List of Tables

List of Figures

Introduction

Numerical simulations of the interaction of blood flow with arterial tissue in the cardiovascular system of human beings have become increasingly popular in the last decade. This is because, these days, cardiovascular diseases are the most frequent cause of death globally [205], and accurate realistic modeling of the cardiovascular system can be a useful tool for medical doctors to diagnose diseases or to predict the risks of consequential damages. The simulations may serve as a reference to decide if a and which surgical intervention may be necessary and also to accommodate optimizing medical methods.

In order to make realistic predictions from numerical simulations, medical measurements and expertise have to be combined with appropriate and reliable material models for the description of the blood flow and the mechanical response of the arterial wall. In addition to that, the Fluid-Structure Interaction (FSI) has to be taken into account, and robust and fast numerical methods are necessary for the solution of the arising linear and nonlinear systems.

As a prerequisite for the simulations, patient-specific geometries have to be measured and converted to 3D computer models, i.e., to CAD (Computer Aided Design) models or polygonal meshes (such as in Figure 0.2), and realistic material parameters for the material models (for the blood and the arterial tissue) have to be obtained from experiments. Typically, the 3D models for the simulations are constructed from magnetic resonance imaging (MRI), Computed Tomography (CT) scans, or catheter-based intravascular ultrasound (IVUS) measurements. From the grey-tone in the resulting images, the media-lumen- and the media-adventitia-interface can be identified and thus allowing the distinction of the individual layers, namely the intima, the media, the adventitia, and, in atherosclerotic arteries, the plaque; see Figure 0.1. Utilizing a series of 2D images and additional X-ray images describing the path of the catheter, a three-dimensional geometry of the artery, including the curvature of the vessel, can be reconstructed [20].

In large vessels, blood, which is in fact a suspension of cells and particles in plasma, can be approximated as a Newtonian fluid, whereas in smaller vessels and capillaries the shear-thinning behavior of the blood has to be taken into account [91]. In particular, on the scale of the cell-level, biochemical processes

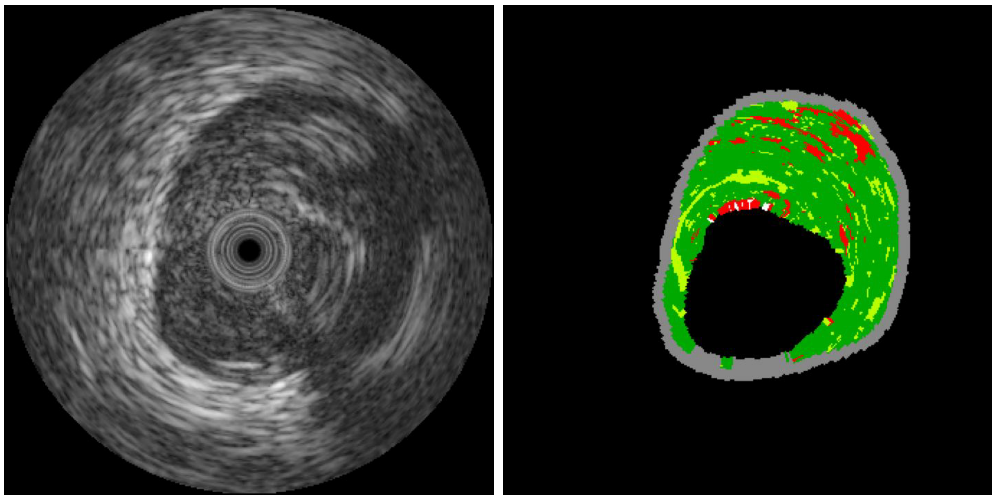


Figure 0.1: IVUS measurements to identify the layers of the arterial wall of a diseased artery, i.e., intima, media, adventitia, and plaque (left). The IVUS measurements do not provide the ability to identify between individual plaque components. Therefore virtual histology (IVUS-VH, property of Volcano Corperation, California, USA) analysis is carried out (right): the components are color-coded as fibrotic tissue (green), fibrofatty tissue (light green), necrotic core (red) and dense calcium (white); cf. [20].

are important to model the blood appropriately. Since larger coronary arteries are in the focus of this thesis, the approximation as a Newtonian fluid is reasonable. Coronary arteries form a network located on the outer layer of the heart wall and supply blood to the myocardium and other components of the heart [91].

Since, e.g., plaque rupture is a high threat in atherosclerotic arteries, reliable computations of the transmural stresses in the arterial wall, which are often considered as the main origin of plaque rupture, are necessary. To model the mechanical response and the stresses of the arterial tissue accurately, sophisticated material models, which take into account the incompressibility, the anisotropy induced by the fibers in the arterial wall, and the viscoelastic behavior of the arterial tissue, have to be utilized; cf. [19, 41, 20, 23, 24]. Therefore, suitable space discretizations have to be considered for the discretization of the arterial wall.

In Chapter 1, the numerical framework and the MPI-parallel (Message Passing Interface) software environment which enable FSI simulations in coronary arteries are introduced and carefully investigated with respect to the material

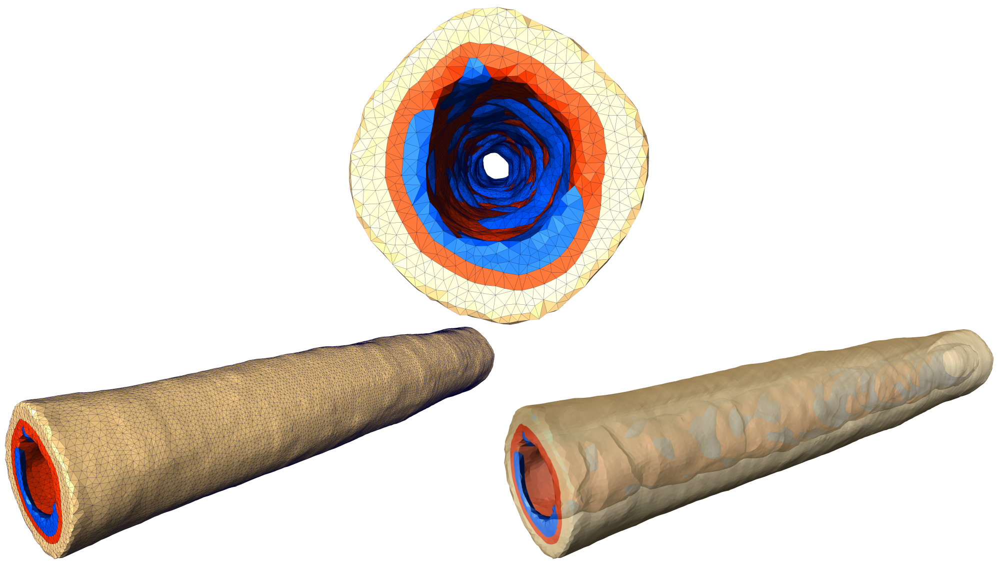


Figure 0.2: Three-dimensional geometry of a diseased artery without curvature used in [136, 20]. The plaque is colored in blue, the media in red, and the adventitia in beige.

models, boundary conditions, and the spatial discretizations. Therefore, we define a benchmark test such that the simulations can be carried out under controlled settings, where we use a section of an idealized coronary artery as the geometry. It is chosen to be simple, but nonsymmetric, to make effects due to the anisotropy of the structure visible. The implementation of our FSI software is based on the coupling of the software libraries `LifeV` [90, 92] and `FEAP` [193], which enables the use of a variety of material models for the arterial wall.

We consider a fully-coupled monolithic approach using a Convective Explicit (CE) time discretization, which is based on the linearization of the convective term in the Navier-Stokes equations; see [24]. The simulations include an initialization phase and several consecutive heartbeats. As a preconditioner for the monolithic tangent matrix, we use a composed Dirichlet-Neumann preconditioner which is based on a block factorization of an approximation of the tangent matrix. The inverse matrices of the fluid, structure, and geometry blocks are approximated by parallel preconditioners.

However, the computation time of our simulations turns out to be comparably large, even if utilizing up to 250 processor cores: the simulation of three heartbeats takes up to several weeks, depending on the refinement of the mesh. For the software framework described in Chapter 1, this time cannot be reduced significantly by using more computer processors in parallel. This is due to the fact

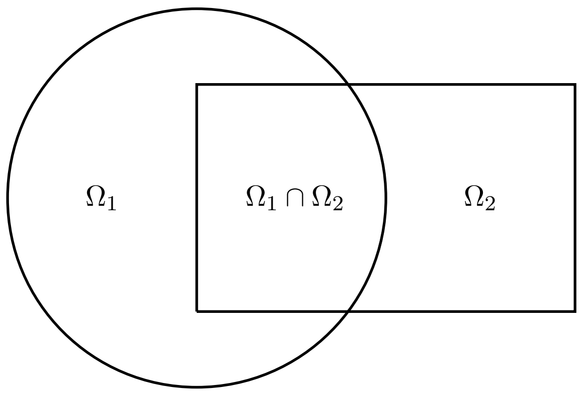


Figure 0.3: Classical setting of the alternating Schwarz method by H. A. Schwarz [180]. The overlap of the subdomains Ω_1 and Ω_2 is $\Omega_1 \cap \Omega_2$.

that, even for the largest mesh used, the local subdomains are already comparably small when using 250 cores (subdomains). Even more importantly, a very fine temporal resolution has to be used, and for the sophisticated (anisotropic, almost incompressible, polyconvex, nonlinear) material models considered in this thesis, the numbers of Newton and Krylov iterations grow very large. On the one hand, the parallel preconditioner, which is used in our framework, i.e., the one-level algebraic Schwarz preconditioner `IFPACK` [173] from the software library `Trilinos` [109], is not scalable. On the other hand, `IFPACK` is not very robust for the nonlinear material models considered for the arterial wall used in our setting. In this thesis, we therefore focus on the improvements of the performance of our FSI software by the use of a better suited parallel preconditioner. Numerical experiments show that this can reduce the simulation time significantly.

Overlapping Schwarz preconditioners [186], which make use of an overlapping Domain Decomposition (DD) of the computational domain, cf. Figure 0.3, are frequently used in the field of FSI in biomechanics, e.g., in [56, 30, 207, 156]. Unfortunately, the `IFPACK` preconditioner lacks robustness and scalability, caused by the algebraic overlap and the missing coarse level. However, a major advantage of this preconditioner is that it can be constructed using only the fully-assembled global stiffness matrix. Therefore, we consider the GDSW (Generalized Dryja-Smith-Widlund) preconditioner, which combines the strengths of `IFPACK` with a geometric overlap and a robust coarse space. The GDSW preconditioner was introduced by Dohrmann, Klawonn, and Widlund in 2008 [70] and is a two-level overlapping Schwarz preconditioner with an energy-minimizing coarse space. As the `IFPACK` preconditioner, the GDSW preconditioner can be computed from the fully-assembled global stiffness matrix, and, even for unstructured domain decompositions, no additional coarse triangulation is required to construct the coarse space. Therefore, it fits well into our FSI framework. The GDSW coarse space is robust for almost incompressible elasticity (cf. [72]), and it is related to the coarse space of the FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) method [82, 83]. This is of advantage since the FETI-DP method was already shown to be robust and scalable for the anisotropic polyconvex hyperelastic material model which is used to describe the mechanical behavior of the arterial wall in this thesis; see [20, 42, 89]. However, in our monolithic context, applying the FETI-DP method is difficult since it does not operate on the original variables but on Lagrange multipliers. In addition to that, it cannot be constructed from the global matrix since the local subdomain matrices are required.

Therefore, a parallel implementation of the GDSW preconditioner based on `Trilinos` is presented in Chapter 2. The implementation is designed such that it can be used in our FSI software without significantly modifications of existing code and without introducing additional software dependencies. Although our implementation is held flexible with respect to the implementation of any two-level overlapping Schwarz preconditioner, we restrict ourselves to the parallel implementation of the GDSW preconditioner here. We test the parallel scalability for Laplacian and linear elastic model problems in two and three dimensions, before we apply the GDSW preconditioner as a block preconditioner for the structure in FSI simulations in Chapter 3. For the Laplacian and the linear elastic model problems, the performance of the GDSW preconditioner is remarkable, for both structured and unstructured domain decompositions. In addition, we investigate possibilities to reduce the dimension of the coarse space and a hybrid version of the GDSW preconditioner. With the parallel implementation of the GDSW preconditioner, the simulation of FSI using nonlinear material models for the structure requires much less Krylov iterations. Also, when using larger time steps, the use of the GDSW preconditioner reduces the number of Krylov iterations compared to the `IFPACK` preconditioner. Both factors help to improve the parallel scalability and reduce the total simulation time significantly.

However, for some types of problems, such as highly heterogeneous (multiscale) problems, the GDSW coarse space is not sufficient; see Chapter 5. In particular, for the discretization or the preconditioning of such problems, additional treatment may be necessary. Multiscale problems appear in virtually all areas of modern science and engineering, e.g., composite materials, porous

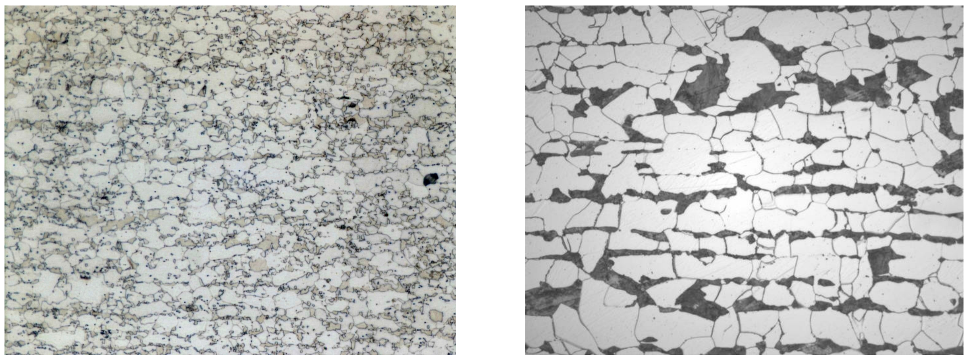


Figure 0.4: Images of the microstructures of dual phase steels obtained from electron backscatter diffraction (EBSD/FIB). Courtesy of Jörg Schröder, University of Duisburg-Essen, Germany, originating from a cooperation with ThyssenKrupp Steel.

media, and turbulent transport in high Reynolds number flow; see [77]. The heterogeneity can occur in many of the properties under consideration, e.g., multiscale fluctuations on the permeability (hydraulic conductivity) of the media when analyzing groundwater transport, or fluctuations in thermal, electrical, or elastic properties at the phase boundaries of composite materials. The microstructure of dual phase steels is a typical example for a multiscale configuration; cf. the images depicted in Figure 0.4. Multiscale models are also used in the field of biomechanics on the cellular, tissue, and organ levels. However, multiscale problems in biomechanics are not object of this thesis.

Among others, the Multiscale Finite Element Method (MsFEM) [77, 114] and Component Mode Synthesis (CMS) method [54, 117, 118] are suitable for the discretization of such problems. In Chapter 4 of this thesis, we consider a special finite element method [111] which was constructed by Hetmaniuk and Lehoucq in 2010 as an Approximation of the CMS method and is therefore named ACMS. However, not being very popular yet, the method is a trade-off between the very good approximation properties of the CMS method and locally supported basis functions, such as in the MsFEM. Both methods, CMS and ACMS, are based on a domain decomposition approach and the computation of eigenfunctions by solving generalized eigenvalue problems, which are global in the CMS method and local in the ACMS method. We present a parallel implementation of the ACMS-based special finite element method in two dimensions, which enables us to test the approximation properties of the discretization for large problems; cf. Chapter 4. In addition to that, we show the application of the FETI-DP method to such kind of special finite element methods. As a result, we observe a quadratic-logarithmical behavior of the condition number which is also typical for the application of the FETI-DP method to standard Lagrangian finite element functions and excellent scalability results.

As a logical completion of this thesis, we use ACMS basis functions for the construction of a coarse space for a two-level overlapping Schwarz preconditioner in Chapter 5. In particular, we select those basis functions which are related to the interface of the decomposition, and present preliminary results which indicate scalability and robustness. Similar to so-called adaptive coarse spaces, some of the basis functions on the interface are computed on the basis of local generalized eigenvalue problems.

Finally, in Chapter 4, we introduce a heuristic strategy to reconstruct the ACMS coarse space using the coefficient function or, alternatively, certain entries of the stiffness matrix. Thus, we are able to avoid the solution of the generalized eigenvalue problems. For this approach, which requires only algebraic information, we also obtain promising preliminary results.

1 Fluid-Structure Interaction in Coronary Arteries

Computational simulations of diseased arteries represent a novel approach in clinical diagnosis and treatment assistance, provided that accurate predictions of the mechanical behavior are available. They may not only help to optimize medical methods of treatment but also enable a more precise assessment whether the decision for a surgical intervention is justified or not. In particular, transmural wall stresses are expected to provide important information for an estimation whether, e.g., an atherosclerotic plaque is likely to rupture if the artery is not treated. Reliable material models as well as robust numerical methods are necessary to provide simulations that allow for a realistic prediction of stresses. In this chapter, which is based on the work in [23] and [24] (Copyright © 2015 John Wiley & Sons, Ltd.), sophisticated nonlinear models for fluid as well as for the structure are combined with appropriate space and time discretizations and efficient parallel solution methods to enable the computation of transmural stresses. This combination of sophisticated nonlinear models for fluid and for structure is not common in Fluid-Structure Interaction (FSI), yet.

We consider a fully-coupled monolithic approach to solve the FSI problem for the geometry of an idealized artery. The fluid dynamics are modeled by the Navier-Stokes equations in Arbitrary-Lagrangian Eulerian (ALE) coordinates [33, 56, 91] and the structure by nonlinear material laws in a Lagrangian frame of reference. The FSI problem is composed of three subproblems, i.e., a fluid, a structure, and a geometry problem. At the fluid-structure interface, we enforce the geometry adherence between the fluid and structure displacement fields, the continuity of the velocities, and the equilibrium of the normal stresses. The system of equations describing the FSI problem is treated as a single system involving all the state variables in a monolithic fashion. In general, this system is nonlinear because of the convective term in the Navier-Stokes equations, the nonlinearity of the constitutive law of the structure, and the moving fluid domain. An overview of FSI in biomechanics and the full FSI model are presented in Section 1.1. Therein, also descriptions of some strongly coupled segregated and some monolithic coupling algorithms are given. In addition to that, reviewing the results of [66], we compare some of these coupling algorithms with respect to their performance and parallel scalability. As a result, the monolithic algorithm seems to be more performant. This motivates the use of a monolithic Convective Explicit (CE) monolithic algorithm for our numerical tests presented in Section 1.5. In the Convective Explicit approach, the convective term of the fluid momentum is linearized by temporal extrapolation and used for solving the fully coupled FSI problem [13, 56]. The resulting discretized nonlinear FSI problem is solved by Newton's method wherein, at each nonlinear iteration, the linearized FSI system is solved in parallel by the GMRES method, preconditioned by an approximated monolithic Dirichlet-Neumann preconditioner [56].

Additionally, we introduce the monolithic Fully-Implicit (FI) time discretization algorithm. In contrast to the CE approach, in the FI approach, the convective term is not linearized. We also describe the FaCSI[1] preconditioner, which was introduced in [65] and can be viewed as a extension of the composed Dirichlet-Neumann preconditioner. Both, the FI algorithm and the FaCSI preconditioner, are used in Chapter 3, where different overlapping Schwarz preconditioners for the structural block are compared. The FaCSI preconditioner involves also the use of a SIMPLE preconditioner [160] for a fluid subproblem on the interior degrees of freedom.

The modeling of the arterial wall by sophisticated material models plays an important role in our discussion. We consider anisotropy as well as the viscoelastic effects accounting for the elastin-rich ground substance with embedded collagen and smooth muscle cells; see also [26, 112]. In Section 1.2, the structure mechanical context is introduced first, followed by the definitions of all material models used for the arterial wall in this thesis.

Aside from the mechanical behavior of the arterial tissue itself, the stresses therein strongly depend on the interaction with the blood flow imposing a complex and inhomogeneous shear stress and pressure distribution on the inner vessel wall surface. Therefore, the fluid-structure interaction is important to be considered in numerical computations. Recently, in [198], FSI in idealized healthy cerebral arteries with both, nonlinear isotropic and anisotropic material constitutive laws, was investigated, particularly highlighting the role of the fibers. In [198], P1 finite elements were used to represent the structure displacements for a non-polyconvex hyperelastic energy, but a grid convergence analysis was not reported. Here, we rather follow a systematic approach with a special focus on the structural side of the simulation.

[1]According to [65] **FaCSI** is an abbreviation for **Fa**ctorization, **C**ondensation, and **SI**MPLE.

With respect to the modeling of the structure, we consider polyconvex hyperelastic energies. We report on the investigations from [24], which is a continuation of our earlier work started in the proceedings article [23], where, to the best of the author's knowledge, for the first time an extended polyconvex anisotropic hyperelastic energy incorporating all mixed invariants was used in FSI. The article [24] presents our framework and our findings in a very detailed way, extending [23], among other experiments, by results of viscoelasticity in FSI.

Beyond simple P1 finite elements, we consider P2 as well as $\bar{\mathrm{F}}$ finite elements for the space discretization of the structure. The latter are based on a three-field formulation [183] to avoid locking caused by the incompressibility constraint; see Section 1.2.6 for a description of $\bar{\mathrm{F}}$ elements and Section 1.4.4 for all combinations of space discretizations (fluid, structure, and geometry) for FSI used in this thesis. The choice of an appropriate space discretization is important to obtain a good estimate of the stresses in the arterial wall.

For our numerical tests, we define a benchmark problem of sufficient complexity to show that our approach is viable and efficient. However, the geometry is chosen rather simple to make an interpretation of the results possible, whereas it is chosen to be nonsymmetric to reveal effects arising from the nonlinearities and the anisotropy of the material models. The settings of our simulations, i.e., the geometry, the material parameters, the space and time discretizations, the meshes, and the boundary conditions, are described in Section 1.4. In particular, we take special care to use a suitable absorbing boundary condition at the outflow; see Section 1.4.6 for the description of the absorbing outflow boundary condition and Section 1.5.1.5 for the corresponding discussion with respect to the numerical results.

The inflow conditions are the driving force of our simulations. We use a ramp phase before applying one or several heartbeats. This ramp can be regarded as an initialization phase and is used only to incorporate the prestretch from a physiological blood pressure; cf. Section 1.4.6 and [24]. While, in principle, any shape can be chosen for the ramp, certain choices may encourage (decaying) oscillations, which can be observed, e.g., in the pressure. Therefore, we discuss the shape of the ramp and possible sources of these oscillations.

We perform mesh convergence studies using all three different combinations of space discretizations (cf. Section 1.4.4), for both phases of the simulations (ramp phase and heartbeat phase). Our smallest mesh has only 30 000 total degrees of freedom, and the largest one has over a million total degrees of freedom; cf. Section 1.4.4. The discussion of our most important numerical results is given in Section 1.5; a collection of all results using the settings of our benchmark problem can be found at the end of the section.

The software environment used in the simulations, including a description of the implementation of the coupling of the software libraries `LifeV` and `FEAP`, is described in Section 1.3.

1.1 Monolithic Fluid-Structure Interaction

This section starts with an overview of the extensive work which has been carried out on the development of algorithms for the solution of time-dependent FSI problems in biomechanics. Then, we describe the framework which is used in this thesis.

The approaches for handling of the coupling in FSI problems are typically categorized either as segregated or as monolithic schemes. However, the distinction is not always straight-forward. Segregated schemes can range from loosely coupled iterative schemes, such as simple, possibly accelerated, fixed point iterations, to schemes with a much stronger coupling still using separate solvers for fluid and structure. Monolithic schemes range from block preconditioners for the fully coupled problem constructed from segregated solvers to preconditioners which are not constructed from separate solvers.

Many researchers have been working on segregated coupling algorithms. For instance, Causin et al. studied loosely coupled FSI algorithms in [47], in contrast to Deparis et al., who studied segregated methods based on Schur complement approaches in [63]. Whereas in the algorithms in [63], the nonlinear Steklov-Poincaré operators are linearized, the Schur complements are computed after linearization by Fernández et al. in [86]. The approach in [208] by Yang and Zulehner is also based on the Steklov-Poincaré operator on the FSI interface. Solvers using inexact factorizations were considered by Badia, Quaini, and Quarteroni in [13]. Gerbeau and Vidrascu studied a quasi-Newton approach on the fixed point formulation using a finite difference approximation of the Jacobian or reduced models in [99], resulting in Newton-like methods. The Interface Quasi-Newton method (IQN) was considered by Degroote et. al in [60, 61]; see also [147].

Monolithic algorithms were investigated, e.g., by Hron and Turek [115], by Gee, Küttler, and Wall in [97] and Küttler et al. in [143], by Bazilevs et al. [32], by Barker and Cai [29, 30, 206, 207], and by Crosetto, Deparis, Fourestey, and Quarteroni in [56]. In [66], strongly coupled segregated Dirichlet-Neumann, Neumann-Dirichlet, and Neumann-Neumann coupling algorithms have been compared with the monolithic Dirichlet-Neumann preconditioner used in this thesis and in [24]. Results presented in [66], and reviewed in Section 1.1.2.1, indicate that the monolithic approach is the fastest and that its parallel scalability is superior in our biomechanical context.

Parallel Algebraic Multigrid (AMG) preconditioners have recently been applied to fully monolithic ALE formulations of FSI problems in the setting of biomechanics, see, e.g., Gee, Küttler, and Wall in [97] and Bazilev et al. [32].

Overlapping Schwarz methods within monolithic approaches were studied in different regimes of severity of the added-mass effect in [56], confirming successful results for 2D obtained already by Barker and Cai [30].

In the context of overlapping Schwarz preconditioners in FSI, we present a parallel framework for two-level overlapping Schwarz preconditioners based on the software library `Trilinos` in Chapter 2. Therein, also the parallel efficiency of the two-level Schwarz GDSW preconditioner for Laplacian and linear elastic model problems is reported. The preconditioner is then also applied to FSI problems in Chapter 3. As a preconditioner for the structural block in FSI problems, good strong scalability and robustness with respect to sophisticated material models are observed.

In this thesis, as well as in [115] and [168], the ALE mapping is obtained as the solution of a Laplace equation (cf. Equation (1.2)), but it is also possible to use the solution of, e.g., an elasticity problem instead; see [172]. There are alternative approaches to the ALE framework. Among these are XFEM approaches; see [202] and references therein. In [166, 170, 75], a fully Eulerian formulation of FSI is used in 2D, to avoid the degeneration of the ALE mapping and to facilitate adaptivity. Other alternatives are space-time finite element methods, cf., e.g., [195, 116, 32], and Eulerian level set formulations; see [53] or [203]. The immersed boundary method can also be applied to FSI problems [162]. Comparisons of different time stepping schemes for FSI problems in ALE-formulation are also known; see, e.g., [168] by Razzaq, Hron, and Turek.

Constructing preconditioners for Navier-Stokes equations is challenging by itself. Among the many approaches are preconditioners based on the SIMPLE method [160] or on approximate factorization of the Navier-Stokes equations [164]. The Pressure Convection-Diffusion (PCD) preconditioner is based on a factorization which converges in at most two GMRES iterations [157]. Another approximation of the Schur complement leads to the Least-Squares Commutator (LSC) preconditioner [78, 80], which is compared to the PCD preconditioner by Elman et al. in [79]. Benzi et al. [34, 35] have introduced the Augmented Lagrangian (AL) preconditioners which are based on an augmented Lagrangian formulation of the corresponding saddle point problem. A comparison between the PCD, the LSC and the AL approaches has been discussed in [201].

Many publications on the mechanical interaction of biological surrounding structures with an interior blood flow focus on the qualitatively correct physiological simulation of the hemodynamics. Here, even simple structural models, e.g., linear elasticity, as in [56], or simple discretizations, e.g., P1, as in [198], can be sufficient.

One of our main objectives, however, is to compute realistic transmural stress distributions resulting from the interior blood flow in an artery. This requires a realistic, i.e., nonlinear and anisotropic, model of the wall structure including eigenstresses; see [85]. As a result of the almost incompressibility, a suitable discretization is also necessary; it is to be expected that simple P1 finite elements will not be sufficient. In [143], Küttler et al. applied nonlinear Saint Venant-Kirchhoff and Neo-Hooke material laws to FSI of biological tissues in different strong coupling schemes. To the best of the author's knowledge, the first approach using nonlinear, polyconvex, anisotropic structural models in the context of FSI has been considered in [23].

Gee, Förster, and Wall proposed methods to compute prestresses in the isotropic large deformation setting and compared them using a 3D model of an abdominal aortic aneurysm recovered from patient-specific CT geometry data in [96]. They also have reported that FSI simulations lead to unrealistic wall deformations unless the prestress is accounted for.

1.1.1 Model Description

We now introduce the fluid-structure interaction problem. Let Ω^f and Ω^s be the domains occupied by the fluid and the solid in their undeformed reference configuration. We denote by $\Gamma = \partial\Omega^f \cap \partial\Omega^s$ the fluid-structure interface in the reference configuration. At any time $t > 0$, the domain occupied by the fluid Ω_t^f can be retrieved from Ω^f by the Arbitrary Lagrangian Eulerian (ALE) mapping,

$$\begin{aligned} \mathcal{A}_t : \quad \Omega^f &\to \Omega_t^f \\ \mathbf{X} &\mapsto \mathcal{A}_t(\mathbf{X}) = \mathbf{X} + \mathbf{d}_f(\mathbf{X}), \end{aligned} \tag{1.1}$$

where $\mathbf{d}_f$ represents the displacement of the computational fluid domain. The use of the ALE formulation allows an arbitrary reconstruction of the volumetric finite element grid in the fluid domain Ω_t^f from the displacement on its boundary $\partial\Omega_t^f$. For the sake of computation, this reconstruction operates directly in the reference configuration. More precisely, since the structural displacement $\mathbf{d}_s$ and the displacement of the fluid domain $\mathbf{d}_f$ coincide on the fluid-structure interface Γ, to obtain $\mathbf{d}_f$, we extend $\mathbf{d}_s$ on Γ to the interior of the reference fluid domain Ω^f by means of a harmonic extension:

$$\begin{cases} -\Delta \mathbf{d}_f = \mathbf{0} & \text{in } \Omega^f, \\ \mathbf{d}_f = \mathbf{d}_s & \text{on } \Gamma, \\ \mathbf{d}_f \cdot \mathbf{n}_f = 0 & \text{on } \partial\Omega^f \backslash \Gamma, \end{cases} \tag{1.2}$$

where $\mathbf{n}_f$ is the outward unit normal to the reference fluid domain boundary. Since the structural displacement $\mathbf{d}_s$ changes in time, the harmonic extension (1.2) allows defining the current configuration of the fluid domain, $\Omega_t^f = \mathcal{A}_t(\Omega^f)$, using the ALE map parametrization (1.1).

In our FSI model, we consider the fluid dynamics governed by the incompressible Navier-Stokes equations written in the ALE frame of reference [33, 91],

$$\begin{cases} \rho_f \left(\left.\dfrac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{X}} + ((\mathbf{u} - \mathbf{w}) \cdot \nabla)\mathbf{u} \right) - \nabla \cdot \boldsymbol{\sigma}_f(\mathbf{u}, p) = 0 & \text{in } \Omega_t^f \times (0, T], \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega_t^f \times (0, T]. \end{cases} \tag{1.3}$$

In (1.3), the term $\frac{\partial}{\partial t}|_{\mathbf{X}} = \frac{\partial}{\partial t} + \mathbf{w} \cdot \nabla$ is the ALE derivative and $\mathbf{X}$ corresponds to the fluid coordinates in reference configuration, ρ_f is the fluid density, $\mathbf{u}$ and p are the fluid velocity and pressure, respectively, and $\boldsymbol{\sigma}_f(\mathbf{u}, p) = 2\mu_f \boldsymbol{\epsilon}(\mathbf{u}) - pI$ is the Cauchy stress tensor (I is the identity matrix). We denote by $\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right)$ the strain rate tensor and by μ_f the dynamic viscosity of the fluid. Furthermore, $\mathbf{w}$ is the fluid mesh velocity

$$\mathbf{w} = \left.\frac{\partial \mathbf{d}_f}{\partial t}\right|_{\mathbf{X}}.$$

The final time of the time interval is T.

We formulate the structure problem in a purely Lagrangian frame of reference. The conservation of momentum for the structure reads

$$\rho_s \frac{\partial^2 \mathbf{d}_s}{\partial t^2} - \nabla \cdot \mathbf{P} = 0 \quad \text{in } \Omega^s \times (0, T], \tag{1.4}$$

where ρ_s is the density of the structure and $\mathbf{P}$ are the first Piola-Kirchhoff stresses of any material model which could be used for the arterial wall, cf., e.g. Section 1.2.2 for linear elasticity, Section 1.2.3 for Neo-Hooke, Section 1.2.4 for a nonlinear anisotropic hyperelastic material model, and Section 1.2.5 for a nonlinear anisotropic viscoelastic material model. For more details on the continuum mechanical context and the corresponding notation, see also Section 1.2.1.

The coupling between the geometry, fluid and structure subproblems is expressed by the coupling conditions

$$\mathbf{d}_f = \mathbf{d}_s \qquad \text{on } \Gamma, \tag{1.5}$$

$$\frac{\partial \mathbf{d}_s}{\partial t} = \mathbf{u} \circ \mathcal{A}_t \qquad \text{on } \Gamma, \tag{1.6}$$

$$(\det[\mathbf{F}])^{-1}\mathbf{F}^{-T}\,\boldsymbol{\sigma}_f\,\mathbf{n}_f \circ \mathcal{A}_t + (\mathbf{F}\mathbf{S})\,\mathbf{n}_s = 0 \qquad \text{on } \Gamma, \tag{1.7}$$

whereas (1.5) expresses the geometric adherence, which already appeared in the definition of the geometry problem (1.2), (1.6) the continuity of the velocities (kinematic condition), and (1.7) the continuity of the stresses (dynamic condition) on Γ. Here, $\mathbf{n}_f$ and $\mathbf{n}_s$ are the outer normal vectors of the fluid and the structural domain, respectively, and $\mathbf{F}$ is the deformation gradient.

The resulting system of equations describing the FSI problem is nonlinear due to the moving fluid domain, the convective term in the fluid momentum equation, and the possible nonlinearity of the structural material model. In this thesis, we use indeed highly nonlinear structural material models formulated in a finite strain framework.

1.1.2 Coupling Algorithms

We use finite differences to approximate the time derivatives of the fluid as well as the structure equations, and the finite element method for the space discretization. We choose conforming fluid and structure meshes at the interface. Specifically, we consider three different combinations of discretizations (fluid, structure, and geometry) for the full FSI problem that we refer to as "P1", "P2", and "$\bar{\text{F}}$"; see Section 1.4.4 and, in particular, Table 1.7.[2]

After space discretization, there are mainly two different possible ways to handle the coupling of the fluid and the structural problem, i.e., by segregated or by monolithic coupling algorithms. In this thesis, monolithic algorithms are used for our FSI framework.

To motivate this choice, we give a short overview of segregated and monolithic approaches of interest. In particular, we focus on strongly coupled segregated algorithms based on Steklov-Poincaré operators and discuss their performance compared to a related monolithic algorithm in Section 1.1.2.1. The monolithic algorithms which are used in the numerical experiments in Section 1.5 and in Chapter 3 are then introduced in Section 1.1.2.2.

[2]The names "P1", "P2", and "$\bar{\text{F}}$" correspond to the space discretization of the structure, and the discretizations for the fluid and the geometry problems are chosen accordingly. For the description of the $\bar{\text{F}}$ discretization for the structure, see Section 1.2.6.

1.1.2.1 Strongly Coupled Segregated Algorithms

Following [64, 63, 62, 66], we consider the equilibrium of forces,

$$S_s\left(d_s\right) + S_f\left(d_f\right) = 0 \tag{1.8}$$

at the FSI interface Γ with Dirichlet-to-Neumann operators S_s and S_f which map the structural and the fluid displacement, respectively, to the corresponding normal stresses. The Dirichlet-to-Neumann operators correspond to the, possibly nonlinear, Schur complements where the variables which do not lie on the FSI interface have been eliminated. These operators are also known as *Steklov-Poincaré operators*, and thus, equation (1.8) is also called the Steklov-Poincaré formulation of the FSI problem.

The geometric adherence condition (1.5) yields the existence of a common displacement, $\boldsymbol{\lambda} = d_s = d_f$, at the FSI interface, such that the Steklov-Poincaré formulation can be written as

$$S_s\left(\boldsymbol{\lambda}\right) + S_f\left(\boldsymbol{\lambda}\right) = 0. \tag{1.9}$$

The Steklov-Poincaré formulation (1.9) can be solved, e.g., by means of a fixed-point iteration on the equation

$$S_s^{-1}\left(-S_f\left(\boldsymbol{\lambda}\right)\right) = \boldsymbol{\lambda}, \tag{1.10}$$

or Newton's method on

$$S_s^{-1}\left(-S_f\left(\boldsymbol{\lambda}\right)\right) - \boldsymbol{\lambda} = 0. \tag{1.11}$$

Equations (1.10) and (1.11) are, however, equivalent to

$$S_f^{-1}\left(-S_s\left(\boldsymbol{\lambda}\right)\right) = \boldsymbol{\lambda} \quad \text{and} \quad S_f^{-1}\left(-S_s\left(\boldsymbol{\lambda}\right)\right) - \boldsymbol{\lambda} = 0, \tag{1.12}$$

respectively, which can also be solved by using a fixed-point or a Newton iteration. The systems (1.10) and (1.11) can be seen as preconditioned by the operator S_s^{-1}, whereas the systems in (1.12) can be seen as preconditioned by the operator S_f^{-1}. The involvement of the inverse S_s^{-1} is particularly favorable when the structural inverse can be computed or approximated easily, e.g., for linear elasticity, where S_s would be linear.

In contrast to weakly coupled segregated algorithms, where the fluid and the structural problems are solved separately in each time step (or for several time steps), the approaches presented here involve the solution of one common nonlinear interface problem. Therefore, they are denoted as strongly coupled segregated algorithms. They are segregated in the sense that, for the solution of the interface problem, the linearized fluid and structural subproblems are solved separately.

Alternatively, (1.8) can be solved directly using Newton's method, which leads to another strongly coupled segregated algorithm. The Newton linearized system reads

$$\left(S_s'\left(\boldsymbol{\lambda}^k\right)+S_f'\left(\boldsymbol{\lambda}^k\right)\right)\delta\boldsymbol{\lambda}^k=-\left(S_s\left(\boldsymbol{\lambda}^k\right)+S_f\left(\boldsymbol{\lambda}^k\right)\right) \tag{1.13}$$

in the k-th Newton step. However, the corresponding Jacobian $S_s'+S_f'$ is typically ill-conditioned. Three choices of preconditioners for (1.13), which are motivated from Domain Decomposition Methods (DDM) [197], are the *Dirichlet-Neumann* $(S_s')^{-1}$, the *Neumann-Dirichlet* $(S_f')^{-1}$, and the *Neumann-Neumann* $(\alpha_s(S_s')^{-1}+\alpha_f(S_f')^{-1})$ preconditioners. The weights for the Neumann-Neumann preconditioner are chosen such that $\alpha_s+\alpha_f=1$, with $0\le\alpha_s,\alpha_f\le 1$. For $\alpha_s=1$ and $\alpha_f=0$, the Neumann-Neumann preconditioner is equal to the Dirichlet-Neumann preconditioner and, on the contrary, for $\alpha_s=0$ and $\alpha_f=1$ it is equal to the Neumann-Dirichlet preconditioner.

In [66], the parallel performance of this strongly coupled segregated approach was compared to the GCE monolithic algorithm, which is described in the next section, using the Dirichlet-Neumann preconditioner for the monolithic algorithm; cf. Section 1.1.3. For the space discretization, P1 elements have been used for the structure, P1-P1 elements for the fluid velocity and pressure (stabilized by interior penalty), and P1 elements for the discretization of the geometry problem. This corresponds to the combination of space discretizations "P1" for the full FSI problem; see Table 1.7 in Section 1.4.4. For all methods, the time domain is discretized by an implicit Euler scheme with time step $\Delta t=10^{-4}\,$s; see also [66, 24] and Section 1.4 for more detailed descriptions of the settings of the FSI benchmark problem which has been used in the simulations.

It can be observed that, for the settings of our FSI benchmark problem, the monolithic approach shows much better performance and parallel scalability than the segregated algorithms under consideration. Regarding the preconditioners for the segregated coupling algorithm (1.13), the Dirichlet-Neumann preconditioner outperforms the Neumann-Dirichlet and the Neumann-Neumann preconditioners, both, in terms of the iteration count and in CPU times. We also observe that the Neumann-Neumann preconditioner performs best when

	Time per time step			GMRES iterations		
Algorithm	**1 CPU**	**2 CPUs**	**4 CPUs**	**1 CPU**	**2 CPUs**	**4 CPUs**
SP - DN	31 s	26 s	22 s	15	15	15
SP - ND	866 s	729 s	635 s	567	567	567
SP - NN ($\alpha_s = 0.5$, $\alpha_f = 0.5$)	590 s	501 s	434 s	274	274	274
SP - NN ($\alpha_s = 0.999$, $\alpha_f = 0.001$)	40 s	34 s	30 s	20	21	21
SP - NN ($\alpha_s = 0.9999$, $\alpha_f = 0.0001$)	32 s	28 s	24 s	15	15	15
Monolithic GCE with composed DN	12 s	8 s	5 s	11	25	50

Table 1.1: Performance of various coupling algorithms for different numbers of CPUs utilized. We compare the time to perform a single time step and the number of GMRES iterations averaged on the first ten time steps performed. Here, DN refers to the Dirichlet-Neumann algorithm, ND to the Neumann-Dirichlet algorithm, and NN to the Neumann-Neumann method. SP refers to the Steklov-Poincaré formulation. In the monolithic GCE scheme the blocks are not inverted but Overlapping Schwarz preconditioners are used. Taken from [66]; courtesy of Deparis, Forti, Quarteroni.

it is close to the Dirichlet-Neumann preconditioner, i.e., when α_s is close to 1 and α_f is close to 0; see also Table 1.1.

1.1.2.2 Monolithic Algorithms

Since the monolithic approach seems to be more performant, especially in the context of hemodynamics, we choose this coupling algorithm for our framework. Thus, we give a short, incomplete review on monolithic FSI algorithms here.

A monolithic FSI approach coupling nonlinear hyperelastic solid models with the Navier-Stokes equations for the fluid is presented by Hron and Turek in [115], considering the incompressible case for the solid. They take a systematic approach starting from 2D; see also Turek et al. [168]. For the solution of the linear saddle point systems, a sparse direct solver, an ILU preconditioner, and a geometric multigrid method with a Vanka-type smoother are considered. A block preconditioner with Schur complements for the monolithic system is presented in [120]. In [200], a brain aneurysm in 2D is discussed, using a Neo-Hookean material for the structure.

A scalable monolithic solver for an FSI problem coupling blood flow with a conforming arterial wall in 2D is presented by Barker in [29] as well as by Barker and Cai in [30]. They apply a Newton scheme with an explicitly computed Jacobian; see also [87], [32], and [30]. For the solution of the arising linearized systems Barker and Cai use a hybrid multilevel Schwarz preconditioner which uses restricted additive Schwarz on the fine level and multiplicative Schwarz on the coarse level. The parallel Newton-Krylov-Schwarz approach for the monolithic system is extended to three dimensions in Wu and Cai [207], and scalability is shown for up to three thousand processors. The solution approach is related to ours, with the difference that we apply Schwarz methods on the blocks of a monolithic Dirichlet-Neumann preconditioner instead of the whole monolithic system.

In [97], Gee, Küttler, and Wall use a monolithic ALE approach to couple a nonlinear Saint Venant-Kirchhoff model with a Navier-Stokes fluid in 3D and solve the arising equations using a Newton scheme based on an exact Jacobian. The authors consider block preconditioners for the monolithic system, i.e., Block-Gauss-Seidel, using AMG for the blocks, as well as a new AMG scheme using Block-Gauss-Seidel smoothing on all levels. The block-AMG approaches have already been compared with partitioned approaches in [143]. Recently, in [145, 144], Langer and Yang considered a Dirichlet-Neumann method for FSI problems in biomechanics using a Mooney-Rivlin model for the structure and a straight tube. Mayr, Klöppel, Wall, and Gee present a monolithic FSI approach using dual mortars in [154]. In our approach, we are able to handle nonmatching grids by using radial basis functions, cf. [67]. Razzaq, Damanik, Hron, Ouazzi, and Turek used an isotropic Neo-Hookean material model in [167] to model the arterial wall in FSI in an aneurysm and Q2P1disc finite elements.

Here, we consider two different monolithic time discretization approaches, i.e., the *Fully Implicit* (FI) and the *Convective Explicit* (CE) time discretizations, cf. [13, 56, 65]. Whereas, in the FI case, we treat all subproblems by an implicit time discretization scheme, in the CE case, we treat only the structure and the geometry subproblems fully implicitly. In the fluid subproblem the convective term of the fluid momentum is linearized as follows:

$$((\mathbf{u}^{n+1} - \mathbf{w}^{n+1}) \cdot \nabla)\mathbf{u}^{n+1} \approx ((\mathbf{u}^* - \mathbf{w}^*) \cdot \nabla)\mathbf{u}^{n+1}, \tag{1.14}$$

with $\mathbf{u}^*$ and $\mathbf{w}^*$ represent temporal extrapolations of the fluid velocity and of the fluid domain velocity, respectively. This choice is suitable when the Reynolds number that characterizes the fluid flow is not high, namely for laminar flows. This condition is typically fulfilled in the problems at hand.

The related *Geometry-Convective Explicit* (GCE) time discretization approach is proposed in [12]. In the GCE approach, the geometry problem is decoupled by using the mesh from the previous time step in the fluid problem. The convective term is treated explicitly, again. As a result, the only remaining nonlinearity occurs in the structural equation (if the material model is nonlinear); see also [57] for detailed discussion of the time discretizations and corresponding preconditioners. We do not use this approach for any further simulations; however, it has been used to compute the results in Table 1.1. Note that, when using a linear elastic material for the structure and the GCE time discretization approach, the resulting monolithic FSI problem is linear in each time step.

After a space-time discretization, the fully coupled nonlinear FSI system reads

$$\begin{pmatrix} F(\mathbf{u}_f^{n+1}, p^{n+1}, \mathbf{d}_f^{n+1}) & + & 0 & + & C_1^T\,\boldsymbol{\lambda}^{n+1} & + & 0 \\ 0 & + & S\,(\mathbf{d}_s^{n+1}) & + & C_3^T\,\boldsymbol{\lambda}^{n+1} & + & 0 \\ C_1\,\mathbf{u}_f^{n+1} & + & C_2\,\mathbf{d}_s^{n+1} & + & 0 & + & 0 \\ 0 & + & C_4\mathbf{d}_s^{n+1} & + & 0 & + & H\,\mathbf{d}_f^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{b}_s \\ C_2\,\mathbf{d}_s^n \\ 0 \end{pmatrix} \tag{1.15}$$

for both, the Fully Implicit and the Convective Explicit case. We denote by $\boldsymbol{\lambda}$ the vector containing the Lagrange multipliers enforcing the balance of normal stresses across Γ. In system (1.15), the fluid subproblem F is nonlinear due to the moving fluid domain and, in case of the FI time discretization, also due to the convective term. The solid subproblem S is nonlinear depending on the material law and finite strain setting used to model the structure deformations. In contrast, the geometry subproblem H is linear. The matrices C_1 and C_2 account for the continuity of the velocity on Γ, the transposed matrices C_1^T and C_3^T account for the balance of normal stresses (imposed weakly), whereas C_4 accounts for the geometric adherence. Assuming conforming meshes and conforming discretizations at the fluid-structure interface yields

$$C_1|_\Gamma = I|_\Gamma, \quad C_2|_\Gamma = 1/\Delta t\, C_3, \quad C_3|_\Gamma = -I|_\Gamma, \text{ and } \quad C_4|_\Gamma = I|_\Gamma, \tag{1.16}$$

where $I|_\Gamma$ is the identity matrix defined on the degrees of freedom on the fluid-structure interface Γ.

When using non-conforming discretizations, the coupling operators C_1, C_1^T, C_2, and C_3^T have to be defined accordingly, e.g., using radial basis functions; see, e.g., [67].

1.1.3 Linearization and Parallel Preconditioner

We solve the nonlinear problem (1.15) by means of the Newton method. At each time step, the Newton algorithm yields the following linear system

$$J_{\mathrm{M}}(\mathbf{x}_k^{n+1})\boldsymbol{\delta}_{k+1} = -\mathbf{r}(\mathbf{x}_k^{n+1}), \tag{1.17}$$

where k denotes the index of the Newton iterations, $J_{\mathrm{M}}(\mathbf{x}_k^{n+1})$ is the tangent matrix associated to the linearized FSI problem, $\mathbf{r}(\mathbf{x}_k^{n+1})$ is the residual, $\boldsymbol{\delta}_{k+1}$ denotes the Newton increment, and $\mathbf{x}_k^{n+1} = (\mathbf{u}, p, \mathbf{d}_s, \boldsymbol{\lambda}, \mathbf{d}_f)$ is the solution vector.

The tangent associated to the FSI problem (1.17) reads

$$J_{\mathrm{M}} = \begin{pmatrix} D_{(\mathbf{u}_f,\mathbf{p})}F & 0 & C_1^T & D_{\mathbf{d}_f}F \\ 0 & D_{\mathbf{d}_s}S & C_3^T & 0 \\ C_1 & C_2 & 0 & 0 \\ 0 & C_4 & 0 & H \end{pmatrix}, \tag{1.18}$$

where $D_{(\mathbf{u}_f,\mathbf{p})}F$ denotes the linearization of the fluid part, $D_{\mathbf{d}_f}F$ are the shape derivatives corresponding to the fluid mesh movement, cf. [88], and $D_{\mathbf{d}_s}S$ denotes the linearization of the solid part.

For each k, i.e., in each Newton iteration, we solve (1.17) using the GMRES method preconditioned by an approximated monolithic Dirichlet-Neumann preconditioner; cf. [56, 57]. This preconditioner is constructed from the Jacobian of the monolithic system J_{M} by neglecting the coupling block C_3^T, resulting in the approximate Jacobian

$$P_{DN} = \begin{pmatrix} D_{(\mathbf{u}_f,\mathbf{p})}F & 0 & C_1^T & D_{\mathbf{d}_f}F \\ 0 & D_{\mathbf{d}_s}S & \mathbf{0} & 0 \\ C_1 & C_2 & 0 & 0 \\ 0 & C_4 & 0 & H \end{pmatrix}. \tag{1.19}$$

As suggested by the name, the Dirichlet-Neumann preconditioner for the monolithic system is related to the Dirichlet-Neumann preconditioner for the segregated coupling algorithms. This can be easily seen by considering the preconditioned system matrix $P_{DN}^{-1}J_{\mathrm{M}}$ neglecting the geometry problem,

$$\begin{aligned} P_{DN}^{-1}J_{\mathrm{M}} &= \begin{pmatrix} D_{(\mathbf{u}_f,\mathbf{p})}F & 0 & C_1^T \\ 0 & D_{\mathbf{d}_s}S & \mathbf{0} \\ C_1 & C_2 & 0 \end{pmatrix}^{-1} \begin{pmatrix} D_{(\mathbf{u}_f,\mathbf{p})}F & 0 & C_1^T \\ 0 & D_{\mathbf{d}_s}S & C_3^T \\ C_1 & C_2 & 0 \end{pmatrix} \\ &= \begin{pmatrix} I & 0 & -(D_{(\mathbf{u}_f,\mathbf{p})}F)^{-T}C_1^T \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & (D_{(\mathbf{u}_f,\mathbf{p})}F)^{-1}C_1^T \\ 0 & I & (D_{\mathbf{d}_s}S)^{-1}C_3^T \\ 0 & 0 & S'_f\left((S'_f)^{-1} + (S'_s)^{-1}\right) \end{pmatrix}, \end{aligned}$$

using the definitions of C_1, C_2, and C_3 for conforming meshes in (1.16). The bottom diagonal block of the right factor,

$$S'_f\left((S'_f)^{-1} + (S'_s)^{-1}\right) = I + S'_f(S'_s)^{-1}, \tag{1.20}$$

relates the Dirichlet-Neumann preconditioner for the monolithic system with the Dirichlet-Neumann preconditioned system matrix for the strongly coupled segregated approach in (1.13),

$$\left(S'_s\right)^{-1}\left(S'_s + S'_f\right) = I + \left(S'_s\right)^{-1}S'_f. \tag{1.21}$$

Note that S'_f operates on the displacement of the computational fluid domain $\mathbf{d}_f$ rather than on the fluid velocity u; cf. Section 1.1.2.1. Therefore, the fluid Schur complement has to be scaled by a factor of Δt to obtain S'_f in the monolithic context.

Instead of applying the inverse P_{DN}^{-1} directly, we apply an approximated Dirichlet-Neumann preconditioner. The *composed Dirichlet-Neumann* preconditioner [57] is obtained by approximating the inverse matrices of the structural block $(D_{\mathbf{d}_s}S)$, the geometry block (H), and the fluid block $((D_{(\mathbf{u}_f,\mathbf{p})}F))$ appearing in

$$P_{DN} = \underbrace{\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & D_{\mathbf{d}_s}S & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{P_{\mathcal{S}}} \underbrace{\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & C_4 & 0 & H \end{pmatrix}}_{P_{\mathcal{G}}} \underbrace{\begin{pmatrix} D_{(\mathbf{u}_f,\mathbf{p})}F & 0 & C_1^T & D_{\mathbf{d}_f}F \\ 0 & I & 0 & 0 \\ C_1 & C_2 & 0 & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{P_{\mathcal{F}}} \tag{1.22}$$

by some domain decomposition preconditioners. In the simulations in Section 1.5, we use `Trilinos IFPACK` [173], i.e., a one-level algebraic additive Schwarz preconditioner, to do so; cf. [186, 197].

In the simulations presented in Chapter 3, we employ the FaCSI preconditioner, which can be seen as an extension of the composed Dirichlet-Neumann preconditioner; see [65]. This preconditioner is built by further decomposing the fluid block, $P_{\mathcal{F}}$ in (1.22), then applying static condensation of the fluid interface variables, and approximation of the remaining fluid matrix by a SIMPLE preconditioner; see [68]. Here, we briefly describe the FaCSI preconditioner; for additional details, we refer to [65].

First, the fluid block $P_{\mathcal{F}}$ is factorized,

$$\underbrace{\begin{pmatrix} D_{\mathbf{d}_f}F & 0 & C_1^T & D_{\mathbf{d}_f}F \\ 0 & I & 0 & 0 \\ C_1 & C_2 & 0 & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{P_{\mathcal{F}}} = \underbrace{\begin{pmatrix} I & 0 & 0 & D_{\mathbf{d}_f}F \\ 0 & I & 0 & 0 \\ 0 & C_2 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{P_{\mathcal{F}}^{(1)}} \underbrace{\begin{pmatrix} D_{(\mathbf{u}_f,\mathbf{p})}F & 0 & C_1^T & 0 \\ 0 & I & 0 & 0 \\ C_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{P_{\mathcal{F}}^{(2)}}. \tag{1.23}$$

Neglecting the identities, $P_{\mathcal{F}}^{(2)}$ has the form

$$\begin{pmatrix} \mathcal{K} & \mathcal{B}^T & C_1^T \\ \mathcal{B} & 0 & 0 \\ \mathcal{C}_1 & 0 & 0 \end{pmatrix}, \tag{1.24}$$

with

$$D_{(\mathbf{u}_f,\mathbf{p})}F = \begin{pmatrix} \mathcal{K} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{pmatrix} \tag{1.25}$$

corresponding to the discretized tangent matrix of the Navier-Stokes equations. Instead of inverting $P_{\mathcal{F}}^{(2)}$, the degrees of freedom on the interface (Γ) are eliminated, resulting in a 2×2 block system of the form

$$\begin{pmatrix} \mathcal{K}_{II} & \mathcal{B}_I^T \\ \mathcal{B}_I & 0 \end{pmatrix} \begin{pmatrix} \delta \mathbf{u}_I \\ \delta \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{\mathbf{u}_I} - \mathcal{K}_{I\Gamma} \delta \mathbf{u}_\Gamma \\ \mathbf{r}_{\mathbf{p}} - \mathcal{B}_\Gamma \delta \mathbf{u}_\Gamma \end{pmatrix}, \tag{1.26}$$

which only involves the remaining (I) degrees of freedom.

The block matrix can be factorized:

$$\mathcal{F} = \begin{pmatrix} \mathcal{K}_{II} & \mathcal{B}_I^T \\ \mathcal{B}_I & 0 \end{pmatrix} = \begin{pmatrix} \mathcal{K}_{II} & 0 \\ \mathcal{B}_I & -\mathcal{S} \end{pmatrix} \begin{pmatrix} I & \mathcal{K}_{II}^{-1} \mathcal{B}_I^T \\ 0 & I \end{pmatrix} \tag{1.27}$$

with $S = \mathcal{B}_I \mathcal{K}_{II}^{-1} \mathcal{B}_I^T$. Using $D = \operatorname{diag}(\mathcal{K}_{II})$ to approximate $\mathcal{K}_{II}$ and the corresponding approximate Schur complement $\tilde{S} = \mathcal{B}_I D^{-1} \mathcal{B}_I^T$, the matrix $\mathcal{F}$ is replaced by a SIMPLE preconditioner $\tilde{\mathcal{F}}$, with

$$\mathcal{F} \approx \tilde{\mathcal{F}} := \begin{pmatrix} \mathcal{K}_{II} & 0 \\ \mathcal{B}_I & -\tilde{S} \end{pmatrix} \begin{pmatrix} I & \mathcal{D}^{-1} \mathcal{B}_I^T \\ 0 & I \end{pmatrix}; \tag{1.28}$$

cf. [68]. For more details on the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) method, we refer to the original work [160] and also to [78, 161, 79]. In [68], the SIMPLE method was interpreted as a preconditioner and used in the context of unsteady Navier-Stokes equations and its application to hemodynamics.

As in the Dirichlet-Neumann preconditioner, some domain decomposition preconditioners are used to approximate the involved inverse matrices. In particular, we use overlapping Schwarz preconditioners for this purpose as well, namely `Trilinos IFPACK` and a parallel implementation of the GDSW preconditioner [107, 106, 105] (for the structural block). The GDSW preconditioner [70, 71] is a two-level overlapping Schwarz preconditioner with an energy-minimizing coarse space functions; for the presentation of our parallel implementation of the GDSW preconditioner, see Chapter 2; for the corresponding results in FSI, we refer to Chapter 3.

1.2 Material Models for the Arterial Wall

Arterial tissue is composed of an elastin-rich ground substance with embedded collagen and smooth muscle cells. This composition yields an anisotropic and viscoelastic material response at finite strains. We thus consider FSI using sophisticated material models for the structure.

To model the hyperelastic response of the structure, various models have been proposed in the literature; however, the essential condition of polyconvexity [18], which guarantees the existence of a unique minimizer of the strain energy function, was only considered during the last decade. More precisely, in [178], anisotropic polyconvex functions were intentionally introduced for the first time. Later however, it was observed that also previously proposed anisotropic functions were indeed polyconvex; see, e.g., the function introduced in [113]. Based thereon, in [26], a variety of polyconvex functions were constructed which a priori satisfy the condition of a stress-free reference configuration. Their numerical performance and their performance using parallel iterative solvers, in particular, the FETI-DP domain decomposition method, were compared in [41]. A larger structural simulation of an arterial wall for a diseased artery using one of these anisotropic, almost incompressible hyperelastic material models was then presented in [136], applying a Newton-Krylov FETI-DP approach. To model embedded collagen fibers, anisotropy is one of the numerical challenges present in models for soft biological tissue. In [22], it was numerically observed that the anisotropy of soft tissue does clearly affect Newton's method as well as the iterative linear solver but that the effect is, in the physiological range, not severe. Damage of the fibers from overstretch [28, 19, 27] has been considered in computations performed with the FETI-DP method, for an arterial segment, in [169, 21]. It was observed that it poses no additional challenge to the solver of the linearized system since the damage rather decreases the effect of the anisotropy.

Based on these results, patient-specific simulations of arteries, neglecting the influence of the blood flow and a viscoelastic material behavior, were presented in [20]. Viscoelastic effects in FSI, using reduced models, were already considered and compared with experiments in [46]. In [179], it was found that the well-known model of [113] also fulfills the polyconvexity condition. A model that goes beyond the concept of hyperelasticity and which includes also the viscoelastic material behavior of arteries is given in [112]. This approach is mainly based on the original framework for viscoelasticity at finite strains in [182]. However, the formulation is restricted to a volumetric-isochoric split of the strain energy function, and therefore, it allows for stresses induced in the fibers by a volumetric strain; cf. [174]. Furthermore, the viscoelastic behavior is not only associated with the smooth muscle cells as it considers overstresses in the complete isochoric part including the response of the elastin matrix. A unified approach for the inelastic response of arterial tissues is given in [119].

In this section, we first clarify the notations (cf. Section 1.2.1) and then introduce the material models which are used for the arterial wall in this thesis. The linear elastic material model, which is the simplest material model and which results in a linear stiffness matrix, is introduced in Section 1.2.2. In Section 1.2.3, the Neo-Hookean material, a rather simple nonlinear hyperelastic isotropic material, is briefly introduced. The linear elastic and the Neo-Hookean material models are not considered in the simulations in this chapter; however, they are utilized to compare the performance of different preconditioners for the structure in FSI simulations in Chapter 3. Next, we introduce a highly nonlinear anisotropic hyperelastic material model, which consists of a hyperelastic part of Neo-Hookean type and anisotropic parts accounting for the fibers in the arterial wall. This model is then extended to an anisotropic viscoelastic material model, which introduces the viscoelasticity on the stress level.

Finally, we describe the $\bar{\mathrm{F}}$ mixed finite element discretization which is suitable for almost incompressible materials. We use this element type together with sophisticated material models in order to obtain enhanced deformations and stresses in the arterial wall. In the numerical results in Section 1.5, we discuss the special importance of a suitable space discretization with respect to the sophisticated models and almost incompressible materials.

Sections 1.2.1 and 1.2.2 are written following [50], [169, Chapter 3], and [40, Chapter VI] and contain well-known definitions and theory, and Sections 1.2.4, 1.2.5, and 1.2.6 are based on [23, 24].

1.2.1 Notation and Basics

Let $\bar{\Omega}' \subset \mathbb{R}^d$ be the (stress-free) *reference configuration* of a body in d dimensions; e.g., Ω^s in the FSI problem, cf. Section 1.1.1. The *current configuration* of the body $\bar{\Omega}$ is given by the mapping

$$\Phi : \bar{\Omega}' \to \mathbb{R}^d, \tag{1.29}$$

with $\bar{\Omega} = \Phi\left(\bar{\Omega}'\right)$, and Φ is called a deformation of the body if

$$\det\left(\mathbf{F}\right) > 0, \tag{1.30}$$

with $\mathbf{F} := \nabla\Phi$. The *displacement* $\mathbf{u}$ of the body is given by $\Phi - id$, where id denotes the identity map.

Under the action of a body force f and a surface force g, the equilibrium equations in the deformed configuration are

$$\begin{aligned} -\operatorname{div}\boldsymbol{\sigma} &= f \qquad \text{in } \Omega, \\ \boldsymbol{\sigma}\cdot\mathbf{n} &= g \qquad \text{on } \partial\Omega_N, \end{aligned} \tag{1.31}$$

with the symmetric Cauchy stress tensor $\boldsymbol{\sigma}$. The corresponding stress tensor in the reference configuration is the *first Piola-Kirchhoff stress tensor*, $\mathbf{P} = \boldsymbol{\sigma}\,\mathrm{Cof}\,(\mathbf{F})$, which is used in, e.g., the formulation of the structural equations of the FSI problem (1.4). The first Piola-Kirchhoff stress tensor is, in general, not symmetric. Thus, the symmetric *second Piola-Kirchhoff stress tensor*

$$\mathbf{S} = \mathbf{F}^{-1}\boldsymbol{\sigma}\,\mathrm{Cof}\,(\mathbf{F}) = \det(\mathbf{F})\,\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T}, \tag{1.32}$$

is introduced.

Elastic materials are materials for which the Cauchy stresses of any deformed configuration are given by some response function $\boldsymbol{\sigma}(\mathbf{F})$ which depends only on the gradient of the deformation $\mathbf{F}$. The specific definition of the function $\boldsymbol{\sigma}(\mathbf{F})$ is also called the *constitutive law* of the material.

A material is called *hyperelastic* if an energy functional $\tilde{\Psi}$ depending on $\mathbf{F}$ exists, such that the first Piola-Kirchhoff stresses are the derivative of this function with respect to $\mathbf{F}$,

$$\mathbf{P} = \partial_{\mathbf{F}}\tilde{\Psi}(\mathbf{F}). \tag{1.33}$$

For materials which fulfill the objectivity condition, i.e. materials which are frame indifferent, an energy function can be chosen which only depends on the *right Cauchy-Green strain tensor*, $\mathbf{C} = \mathbf{F}^T\mathbf{F}$. Then, the second Piola-Kirchhoff stresses also arise as the double of the derivative of the strain energy function Ψ with respect to the right Cauchy-Green tensor

$$\mathbf{S} = 2\partial_{\mathbf{C}}\Psi(\mathbf{C}). \tag{1.34}$$

More precisely, if a material fulfills the objectivity condition, the strain energy function can be represented in terms of the *principal invariants* of $\mathbf{C}$, i.e.,

$$I_1 = \mathrm{tr}\,\mathbf{C}\,,\ I_2 = \mathrm{tr}\,[\mathrm{Cof}\,\mathbf{C}]\,,\ I_3 = \det\mathbf{C}. \tag{1.35}$$

The right Cauchy-Green strain tensor accounts for local strains of the body. Thus any deformation with

$$\mathbf{C}(x) = I \qquad \forall x \in \Omega' \tag{1.36}$$

is called a rigid body motion, which corresponds to combinations of translations and rotations of the body. In this case, no local strains are introduced but only a movement of the complete body.

The nonlinear strain tensor is given by

$$\mathbf{E} := \frac{1}{2}\left(\mathbf{C} - I\right),$$

or entry-wise by

$$\mathbf{E}_{ij} = \frac{1}{2}\left(\partial_j \mathbf{u}_i + \partial_i \mathbf{u}_j\right) + \frac{1}{2}\sum_k \partial_i \mathbf{u}_k \partial_j \mathbf{u}_k. \tag{1.37}$$

For an isotropic material, satisfying the condition of objectivity, the second Piola-Kirchhoff stress tensor can be written as

$$\mathbf{S} = \gamma_0 I + \gamma_1 \mathbf{C} + \gamma_2 \mathbf{C}^2, \tag{1.38}$$

where γ_0, γ_1, and γ_2 are functions of the principal invariants of $\mathbf{C}$, given in (1.35). We can reformulate this expression using $\mathbf{E}$ instead of $\mathbf{C}$:

$$\mathbf{S}\left(\mathbf{E}\right) = \gamma_0\left(\mathbf{E}\right) I + \gamma_1\left(\mathbf{E}\right)\mathbf{E} + \gamma_2\left(\mathbf{E}\right)\mathbf{E}^2. \tag{1.39}$$

Neglecting the higher-order terms of $\mathbf{E}$ leads to the linear *Saint Venant-Kirchhoff* material law,

$$\mathbf{S} = 2\mu\mathbf{E} + \lambda\,\mathrm{trace}(\mathbf{E})I, \tag{1.40}$$

with the Lamé parameters λ and μ. In particular, the stresses $\mathbf{S}$ are linear with respect to the nonlinear strain tensor $\mathbf{E}$, but $\mathbf{E}$ is not linear with respect to the displacement $\mathbf{u}$ (in contrast to linear elastic materials, cf. Section 1.2.2). The second Lamé parameter is also called *shear-modulus*. The strain energy function corresponding to the Saint Venant-Kirchhoff material law is

$$\Psi = \frac{\lambda}{2}\left(\mathrm{trace}\,\mathbf{E}\right)^2 + \mu\,\mathrm{trace}\,\mathbf{C}. \tag{1.41}$$

1.2.2 Linear Elasticity

As already mentioned in Section 1.2.2, using the nonlinear strain tensor $\mathbf{E}$ yields also quadratic terms of the derivates of $\mathbf{u}$, cf. (1.37). This is the case, e.g., in the Saint Venant-Kirchhoff material law. Under the assumption of small strains, the quadratic terms in the strain tensor can be neglected, and we obtain the symmetric linearized strain tensor

$$\boldsymbol{\epsilon}_{ij} = \frac{1}{2}\left(\partial_j \mathbf{u}_i + \partial_i \mathbf{u}_j\right). \tag{1.42}$$

Using the linearized strain tensor $\boldsymbol{\epsilon}$ instead of the nonlinear strain tensor $\mathbf{E}$ in (1.40), yields the second Piola-Kirchhoff stresses of linear elasticity:

$$\mathbf{S} = 2\mu\boldsymbol{\epsilon} + \lambda\,\mathrm{trace}(\boldsymbol{\epsilon})I.$$

For linear elasticity, in contrast to the Saint Venant-Kirchhoff material law, $\mathbf{S}$ is linear with respect to the displacement $\mathbf{u}$ since $\boldsymbol{\epsilon}$ is linear with respect to $\mathbf{u}$. As a consequence, a linear elastic problem is linear with respect to the displacement.

Note that

$$\mathrm{trace}\,\boldsymbol{\epsilon} = \mathrm{div}\,\mathbf{u}$$

holds, because of (1.42). Thus, λ describes stresses due to volumetric changes.

In the case of linear material models, e.g., for linear elasticity or Saint Venant-Kirchhoff, the Lamé parameters are equivalent to two other material parameters, i.e., the *Young modulus* E and the *Poisson's ration* ν, with the following relations:

$$\begin{aligned} \nu &= \frac{\lambda}{2\left(\lambda+\mu\right)}, & E &= \frac{\mu\left(3\lambda+2\mu\right)}{\lambda+\mu}, \\ \lambda &= \frac{E\nu}{\left(1+\nu\right)\left(1-2\nu\right)}, & \mu &= \frac{E}{2\left(1+\nu\right)}. \end{aligned} \tag{1.43}$$

There is also a linear relation to the *bulk modulus*

$$\kappa = \lambda + \frac{2}{3}\mu \tag{1.44}$$

in the context of linear material models. Physical constraints yield $\lambda > 0$, $\mu > 0$, $E > 0$, $0 < \nu < 1/2$, and $\kappa > 0$. The Poisson's ratio accounts for the incompressibility of the material. In particular, for almost incompressible materials, ν is close to 0.5.

Now, we consider a specific boundary value problem for a linear elastic body, applying a body force f and a surface force g on the Neumann boundary $\partial\Omega_N$. Additionally the body $\bar{\Omega}$ is clamped at the Dirichlet boundary $\partial\Omega_D$. The boundary value problem reads

$$\begin{aligned} \operatorname{div}\boldsymbol{\sigma} &= f && \text{in } \Omega,\\ \mathbf{u} &= 0 && \text{on } \partial\Omega_D, \\ \boldsymbol{\sigma}\cdot\mathbf{n} &= g && \text{on } \partial\Omega_N. \end{aligned} \tag{1.45}$$

The derived weak formulation is

$$\operatorname*{arg\,min}_{\mathbf{u}\in V} \frac{1}{2}\int_\Omega \boldsymbol{\epsilon}(\mathbf{u}) : \boldsymbol{\sigma}(\mathbf{u}) - f\cdot\mathbf{u}\,dx - \int_{\partial\Omega_N} g\cdot\mathbf{u}\,ds \tag{1.46}$$

with respect to the Sobolev space $V = \left(H_0^1(\Omega,\partial\Omega)\right)^d$. Alternatively, using

$$\begin{aligned} \frac{1}{2}\boldsymbol{\sigma}(u) : \boldsymbol{\epsilon}(u) &= \frac{\lambda}{2}\left(\operatorname{trace}\boldsymbol{\epsilon}\left(\mathbf{u}\right)\right)\left(\operatorname{trace}\boldsymbol{\epsilon}\left(\mathbf{v}\right)\right) + \mu\boldsymbol{\epsilon}\left(\mathbf{u}\right) : \boldsymbol{\epsilon}\left(\mathbf{v}\right)\\ &= \frac{\lambda}{2}\operatorname{div}\left(\mathbf{u}\right)\operatorname{div}\left(\mathbf{v}\right) + \mu\boldsymbol{\epsilon}\left(\mathbf{u}\right) : \boldsymbol{\epsilon}\left(\mathbf{v}\right), \end{aligned}$$

yields

$$2\mu\int_\Omega \boldsymbol{\epsilon}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v})\,dx + \lambda\int_\Omega \operatorname{div}(\mathbf{u})\operatorname{div}(\mathbf{v})\,dx = \int_{\partial\Omega_D} f\mathbf{v}\,dx + \int_{\partial\Omega_N} g\cdot\mathbf{v}\,ds \tag{1.47}$$

for all $\mathbf{v}\in V$. Here, $\boldsymbol{\epsilon} : \boldsymbol{\sigma} := \sum\limits_{ij}\epsilon_{ij}\sigma_{ij} = \operatorname{trace}\left(\epsilon^T\sigma\right)$ if considering ϵ and σ as matrices. Finally, to solve this variational problem, we apply some space discretization, such as the finite element method, and solve the resulting system of linear equations.

The null space of the linearized strain tensor $\boldsymbol{\epsilon}$ is the space of the (linearized) rigid body modes. In two dimensions it is spanned by two translations,

$$r_1 := \begin{bmatrix} 1\\ 0 \end{bmatrix}, \quad r_2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{1.48}$$

and one rotation (or the linear approximation to the rotation),

$$r_3 := \begin{bmatrix} -(x_2 - \hat{x}_2) \\ x_1 - \hat{x}_1 \end{bmatrix}. \tag{1.49}$$

In three space dimensions, however, the null space is spanned by three translations,

$$r_1 := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad r_2 := \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad r_3 := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{1.50}$$

and three linearized rotations,

$$r_4 := \begin{bmatrix} x_2 - \hat{x}_2 \\ -(x_1 - \hat{x}_1) \\ 0 \end{bmatrix}, r_5 := \begin{bmatrix} -(x_3 - \hat{x}_3) \\ 0 \\ x_1 - \hat{x}_1 \end{bmatrix}, r_6 := \begin{bmatrix} 0 \\ x_3 - \hat{x}_3 \\ -(x_2 - \hat{x}_2) \end{bmatrix}. \tag{1.51}$$

In both cases, we have shifted the origin of the rotation to the point $\hat{x} \in \Omega$. In order to obtain unique solvability, essential boundary conditions have to be set to control the rigid body motions. In particular, we have to fix at least 3 or 6 linearly independent degrees of freedom in 2D or 3D, respectively; cf., e.g., Section 1.4.6 for the boundary conditions of our FSI benchmark.

The linear elastic material model is the most simple model due to the fact that the corresponding equation is linear with respect to the displacement. It can therefore very practical because the deformations are relatively small in many applications, and thus the linear behavior is sufficiently accurate.

Also in FSI simulations, the use of linear elastic material models is not uncommon; cf., e.g., [206, 29, 56, 51, 60, 66]. This can be feasible if the focus lies on the fluid flow and the influence of the structure is relatively low. For instance in biomechanics, where the structural response is typically by far more complicated, the use of a linear elastic material model may a very rough simplification. In particular, in the simulation of FSI in the human cardiovascular system, the stresses in the structure are typically of high interest, and they can be only approximated poorly by a linear elastic model.

1.2.3 Neo-Hookean Material

The Neo-Hookean material law [209] is hyperelastic (and objective), and therefore, a strain energy function Ψ exists, which depends only on the principal invariants of $\mathbf{C}$, namely I_1, I_2, and I_3.

The strain energy function is specified in form of an isochoric-volumetric split, i.e., as the sum of an isochoric and a volumetric part:

$$\Psi_{NH} = \underbrace{\frac{1}{2}\mu\left(\bar{I}_1 - 3\right)}_{\text{isochoric part}} + \underbrace{\frac{\kappa}{4}\left((J-1)^2 + (\ln J)^2\right)}_{\text{volumetric part}}, \tag{1.52}$$

Isochoric part	Volumetric part
$\mu/2\,(I_1-3)$	$1/4\,(\kappa-2\mu/3)\,\left(J^2-1-2\ln J\right)$
	$1/2\,(\kappa-2\mu/3)\,(J-1)^2$
	$1/2\,(\kappa-2\mu/3)\,(\ln J)^2$
$\mu/2\,\left(\bar{I}_1-3\right)$	$\kappa/4\,\left(J^2-1-2\ln J\right)$
	$\kappa/2\,(J-1)^2$
	$\kappa/2\,(\ln J)^2$

Table 1.2: Different types of Neo-Hookean material models implemented in `FEAP`; cf. [194].

with $J = I_3^{1/2} = (\det \mathbf{C})^{1/2}$ and $\bar{I}_1 = J^{-2/3} I_1 = I_3^{-1/3} I_1$. The parameters μ and κ are the shear modulus and the bulk modulus, respectively; cf. Section 1.2.1. The formulation of the Neo-Hookean material law (1.52) is implemented in `LifeV`.

There are many different types of Neo-Hookean material models due to different possible choices for the isochoric and the volumetric part; see, e.g., the `FEAP` user manual [194] and Table 1.2. Using the new interface of `LifeV` and `FEAP`, cf. Section 1.3, all Neo-Hookean materials which are implemented in `FEAP` are available in our FSI code in `LifeV`; also all other material available in `FEAP` can be used. For more details on the coupling of `LifeV` and `FEAP`, see Section 1.3.

The Neo-Hookean material model is very often used in order to describe rubber-like materials. Also note that the strain energy function of the Neo-Hookean material is not convex but polyconvex; therefore, the existence of a unique minimizer is guaranteed; cf. [40, 50].

1.2.4 Anisotropic Polyconvex Hyperelastic Material Model

Going from linear elasticity, Saint Venant-Kirchhoff, and Neo-Hookean material models to more sophisticated and realistic material models for the structure, i.e., the arterial vessel wall, we describe an anisotropic polyconvex hyperelastic material model in this section. The arterial wall contains reinforcing fibers (collagen and smooth muscle cells), which are aligned in mainly two distinct directions and wind cross-wise helically around the arterial wall, and an elastin-rich ground substance.

Assuming a weak interaction of the fiber families, which are the origin of the anisotropy, we consider an additively decoupled energy consisting of two transversely isotropic parts $\psi_{(a)}^{\mathrm{ti},\infty}$ for the individual fiber families (a) and a purely isotopic part ψ^{isot} for the ground substance. The model is formulated in the framework of classical continuum mechanics at finite strains, and the modeling of anisotropy employs the concept of structural tensors; see, e.g., [38]. In particular, an additional argument tensor, the structural tensor for transverse isotropy $\mathbf{M}_{(a)} = \mathbf{a}_{(a)} \otimes \mathbf{a}_{(a)}$, is considered, such that

$$\psi^{\text{ti},\infty}_{(a)} := \psi^{\text{ti},\infty}_{(a)}(\mathbf{C}, \mathbf{M}_{(a)}).$$

Here, $\mathbf{a}_{(a)}$ is the direction vector corresponding to the fiber family (a).

The strain energy function of the material model can be represented in terms of the principal and mixed invariants,

$$I_1 = \text{tr}\mathbf{C}\,,\ I_2 = \text{tr}[\text{Cof}\,\mathbf{C}]\,,\ I_3 = \text{det}\mathbf{C}\,,\ J_4^{(a)} = \text{tr}[\mathbf{C}\mathbf{M}_{(a)}]\,,\ J_5^{(a)} = \text{tr}[\mathbf{C}^2\mathbf{M}_{(a)}].$$

Since $J_5^{(a)}$ itself is not polyconvex, it is replaced by $K_3^{(a)} := I_1 J_4^{(a)} - J_5^{(a)}$; see [178]. The polyconvexity condition in the sense of [18] is the essential condition to ensure the existence of minimizers and material stability, cf. [179]. To obtain polyconvexity, the strain energy function is expressed in the polynomial basis, $\mathcal{P} := \{I_1, I_3, K_3^{(1)}, K_3^{(2)}\}$, whereas the isotropic part ψ^{isot} is chosen such that it depends only on $\mathbf{C}$ in order to fulfill the objectivity condition; cf. Section 1.2.1. More precisely, a Neo-Hookean energy function,

$$\psi^{\text{isot}} = \epsilon_1 \left(I_3^{\epsilon_2} + I_3^{-\epsilon_2} - 2\right) + c_1 \left(I_1\, I_3^{-1/3} - 3\right), \tag{1.53}$$

is considered for the isotropic part, and for the transversely isotropic part, the function for arterial tissues proposed in [26] is used; see also [25], where this function is also applied in an engineering context.

The transversely isotropic parts are given by

$$\psi^{ti,\infty}_{(a)} = \alpha_1 \left\langle K_3^{(a)} - 2 \right\rangle^{\alpha_2}. \tag{1.54}$$

The restrictions $c_1 > 0$, $\epsilon_1 > 0$, $\epsilon_2 > 1$, $\alpha_1 > 0$, and $\alpha_2 > 2$ ensure polyconvexity and smooth tangent moduli; the Macaulay brackets are defined as $\langle \bullet \rangle := 1/2(|\bullet| + \bullet)$. Note that a volumetric-isochoric split is considered for the isotropic function, but not for the transversely isotropic part in order to avoid the unphysical behavior observed in [174].

Using (1.53) and (1.54), we obtain the strain energy function

$$\psi(\mathbf{C}, \mathbf{M}_{(1)}, \mathbf{M}_{(2)}) = \psi^{\text{isot}}(I_1, I_3) + \sum_{a=1}^{2} \psi^{\text{ti},\infty}_{(a)}(I_1, K_3^{(a)}), \tag{1.55}$$

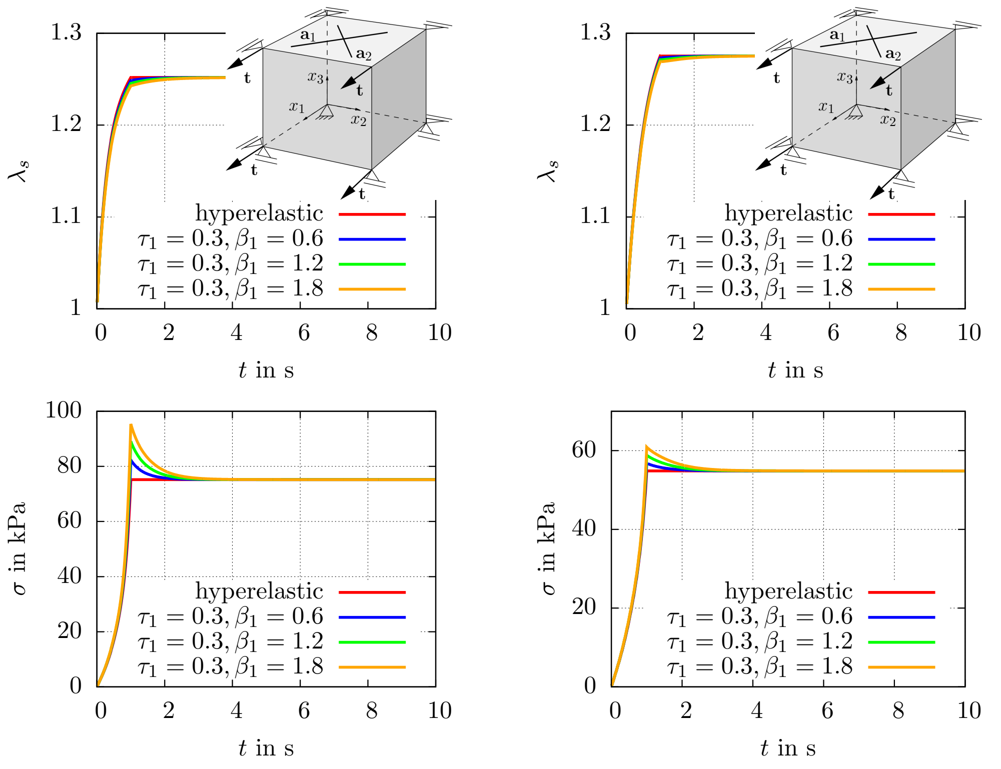


Figure 1.1: Creep (top) and relaxation (bottom) tests in circumferential (left) and axial (right) direction for different viscoelastic material parameter sets. Taken from [24]; courtesy of Balzani, Fausten, and Schröder. Copyright © 2015 John Wiley & Sons, Ltd.

cf. [113].

We use this material model in many of the simulations in Section 1.5 with the material parameters from Section 1.4.2.

1.2.5 Anisotropic Viscoelastic Material Model

Based on the anisotropic hyperelastic material model from the previous section, in this section, we describe how viscoelastic effects in the reinforcing fibers can be incorporated to model the mechanical behavior of the layers of the arterial wall even more appropriately. We present the approach from [84, 23, 24], where a viscoelastic overstress in direction of the fibers is introduced in form of an internal variable $\mathbf{Q}^{\mathrm{ti}}_{\alpha(a)}$ on the stress level. In order to avoid an unphysiological response, the overstresses are not considered to be isochoric, as discussed in [174]. In contrast to the approach discussed here, in [112], viscoelasticity is considered for the isotropic elastin-rich ground substance.

In correspondence to (1.34), we compute the second Piola-Kirchhoff stresses from the strain energy function of each additive part, i.e., $\mathbf{S}^{\text{isot}} := 2\partial_{\mathbf{C}}\psi^{\text{isot}}$ and $\mathbf{S}^{\text{ti},\infty}_{(a)} := 2\partial_{\mathbf{C}}\psi^{\text{ti},\infty}_{(a)}$. Then we add the viscoelastic overstresses to the stresses, resulting in the second Piola-Kirchhoff stresses of the viscoelastic material model,

$$\mathbf{S} = \mathbf{S}^{\text{isot}} + \sum_{a=1}^{2} \left[\mathbf{S}^{\text{ti},\infty}_{(a)} + \sum_{\alpha=1}^{m} \mathbf{Q}^{\text{ti}}_{\alpha(a)} \right], \tag{1.56}$$

where the inner summation represents the classical parallel arrangement of viscoelastic elements; cf. [182]. In particular, m corresponds to the number of viscoelastic relaxation processes. The $\mathbf{Q}^{\text{ti}}_{\alpha(a)}$ evolve in time corresponding to the linear differential equation

$$\dot{\mathbf{Q}}^{\text{ti}}_{\alpha(a)} + \frac{\mathbf{Q}^{\text{ti}}_{\alpha(a)}}{\tau_\alpha} = \beta^\infty_\alpha \dot{\mathbf{S}}^{\text{ti},\infty}_{(a)}, \tag{1.57}$$

with the relaxation parameter τ_α and the associated viscoelastic intensity β_α. We solve (1.57) using the update formula

$$\mathbf{Q}^{ti}_{\alpha(a)} = \mathbf{H}_{\alpha(a)n} + \hat{\mathbf{Q}}^{ti}_{\alpha(a)}, \tag{1.58}$$

where $\mathbf{H}_{\alpha(a)n}$ depends only on quantities evaluated at the previous time step (index n), and is given by

$$\mathbf{H}_{\alpha(a)n} = \exp\left(\frac{-\Delta t}{2\tau_\alpha}\right) \left[\exp\left(\frac{-\Delta t}{2\tau_\alpha}\right) \{\mathbf{Q}^{\text{ti}}_{\alpha(a)}\}_n - \beta^\infty_\alpha \{\mathbf{S}^{\text{ti},\infty}_{(a)}\}_n \right]. \tag{1.59}$$

The second part of (1.57), $\hat{\mathbf{Q}}^{ti}_{\alpha(a)}$, is computed on the basis of the transversely isotropic second Piola-Kirchhoff stresses of the hyperelastic material model, $\mathbf{S}^{\text{ti},\infty}_{(a)}$. It is computed by the formula

$$\hat{\mathbf{Q}}^{\text{ti}}_{\alpha(a)} = \beta^\infty_\alpha \exp\left(\frac{-\Delta t}{2\tau_\alpha}\right) \mathbf{S}^{\text{ti},\infty}_{(a)} \tag{1.60}$$

at the current time step. For further information regarding the derivation of the update formulae from the evolution equation, we refer to [112] and the references therein. Here, only one viscoelastic relaxation process is considered in order to keep the number of additional material parameters small; therefore, we choose $m = 1$.

In our simulations, the derivatives of the second Piola-Kirchhoff stresses with respect to the right Cauchy-Green tensor are numerically computed using the complex-step derivative approximation scheme which was proposed in [192] for the calculation of tangent moduli in a finite strain setting.

To illustrate the model response of the viscoelastic material model, virtual creep and relaxation tests have been performed in [24] for different viscoelastic parameters β_1 and τ_1 in circumferential as well as in axial direction; cf. Figure 1.1. Therefore, parameters from experimental data of the media of a human abdominal aorta have been used for the hyperelastic part; cf. [41]. The fiber angle has been set to $\bar{\beta}_{\mathrm{f}} = 43°$. The viscoelastic parameters have not been adjusted to experimental data but chosen to obtain a significantly high viscoelastic effect; see [24] for more details.

For the creep tests, a stress of 75 kPa is applied incrementally in steps of one second in circumferential direction and 55 kPa in axial direction. Then, the stresses are kept constant for nine additional seconds in order to analyze the resulting creep behavior. For the relaxation tests, a stretch of $\lambda_s = 1.25$ in circumferential and $\lambda_s = 1.27$ in axial direction is applied within one second and then, again, kept constant for nine additional seconds. These stretches are associated to the stresses considered in the creep tests.

The results obtained are shown in Figure 1.1. On the one hand, the anisotropy of the material can be observed by comparing the results for circumferential (left) and axial (right) direction. On the other hand, the sensitivity with respect to the viscoelastic intensity parameters β_α is visible; it is higher for the relaxation than for the creep test. Since the deformation of arteries is mostly traction driven (induced by the blood flow) and less displacement driven, which corresponds to the creep test rather than to the relaxation test, we expect a comparatively small sensitivity with respect to the viscoelastic parameters in our FSI simulations.

1.2.6 Three-Field Mixed Finite Elements

In order to avoid locking effects arising in finite element simulations with almost incompressible materials, we use the three-field (mixed) finite element formulation which is also known as the $\bar{\mathrm{F}}$-approach as a space discretization for the structure; see [183, Section 45].

With $J = J(\boldsymbol{\varphi}) = \det(\mathbf{F})$, we have

$$\mathbf{F} = J^{1/3}\tilde{\mathbf{F}}, \quad \tilde{\mathbf{F}} = J^{-1/3}\mathbf{F}. \tag{1.61}$$

We introduce a new scalar variable θ, satisfying $\theta = J$ in a weak sense, and

$$\bar{\mathbf{F}} := \theta^{1/3}\tilde{\mathbf{F}}, \quad \bar{\mathbf{C}} := \bar{\mathbf{F}}^T\bar{\mathbf{F}} \tag{1.62}$$

with $\bar{\mathbf{F}} = \bar{\mathbf{F}}(\boldsymbol{\varphi}, \theta)$, $\bar{\mathbf{C}} = \bar{\mathbf{C}}(\boldsymbol{\varphi}, \theta)$. Then, we consider the following three-field Lagrangian function

$$Ł(\boldsymbol{\varphi}, \theta, \pi) = \int_\Omega W(\bar{\mathbf{C}}(\boldsymbol{\varphi}, \theta)) + \pi(J(\boldsymbol{\varphi}) - \theta)\, dx - V_{\text{ext}}(\boldsymbol{\varphi}), \tag{1.63}$$

where $V_{\text{ext}}(\boldsymbol{\varphi})$ is the potential energy of external forces; for more details, see [183, Section 45]. It is discretized by P2-P0-P0 mixed finite elements, i.e., piecewise quadratic elements for the deformation field $\boldsymbol{\varphi}$ and piecewise constant elements for the scalar fields θ and π. Local static condensation of θ and π on each finite element leads to a reduced problem, which is then solved. Since the reduced problem is formulated in the degrees of freedom of the deformation field, the implementation of the $\bar{\text{F}}$ approach differs from the implementation of standard piecewise quadratic elements only in the assembly of Jacobian matrix and the residual vector, on the element-level.

As a result of using the $\bar{\text{F}}$-discretization, volumetric changes are not penalized point-wise but rather in an element-wise average sense by the term $\epsilon_1 \left(I_3^{\epsilon_2} + I_3^{-\epsilon_2} - 2\right)$; see the hyperelastic energy in Equation (1.53). Note that, since the almost incompressibility constraint is nonlinear, there is no direct relation to a Poisson's ratio in the linear case. The severity of potential locking, if standard finite elements are used, is therefore difficult to assess a priori. Typical parameter sets for biological soft tissue can still result in a volumetric change at the order of one percent [42], which is considered acceptable. This is one of the reasons for our numerical study with respect to the space discretization of the arterial wall; see, e.g., Sections 1.5.1.2 and 1.5.2.1. Our numerical results indeed show that P1 finite elements are not sufficient to obtain good estimates of all quantities of interest, even for simulations within the physiological range; see Section 1.5.2.3 on the stresses. Surprisingly, for viscoelasticity, we even observe a qualitatively wrong behavior using P1 elements; see Section 1.5.1.4.

The implementation of the almost incompressibility constraint by a penalty term can pose challenges to direct and iterative solvers for the linearized systems. In [42, 39], it was observed that an augmented Lagrange approach can be computationally profitable in the context of soft biological tissue: in the quasi-static setting larger pseudo-time steps could be chosen and the number of Krylov iteration for the linearized systems was reduced. It is also noteworthy that, using an augmented Lagrangian method, the element-wise volumetric change can exactly be controlled, whereas, using a penalty term, the violation is known only a posteriori. Here, however, to avoid additional complications, we do not apply an augmented Lagrange approach for the incompressibility. This increases the challenges for the iterative solution method.

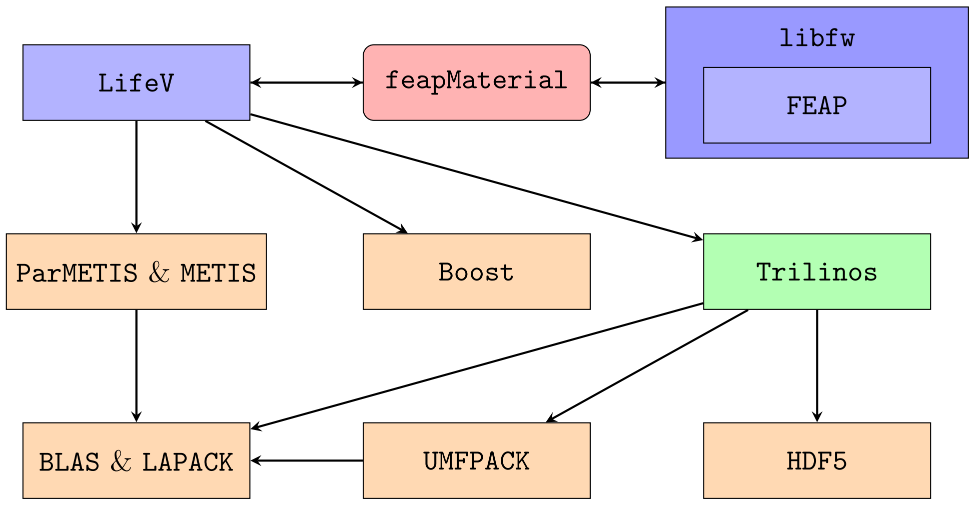


Figure 1.2: Dependencies of software packages which are needed for the LifeV-FEAP coupled FSI software. The most important packages are LifeV [90, 92], which strongly relies on Trilinos [109] in many regards, and FEAP [193]. We make use of a wrapper library for FEAP, i.e., libfw [89]. The packages BLAS, LAPACK [6], UMFPACK [59], HDF5 [196] are needed to compile Trilinos, and in addition we require METIS, ParMETIS [123], and Boost [1, 177] to build our FSI application in LifeV. The coupling (i.e., feapMaterial) is established between LifeV and FEAP (using the features of libfw).

1.3 Coupling of LifeV and FEAP

In this section, the coupling of the software packages LifeV and FEAP (feapMaterial) is described. The coupling is necessary in order to run FSI simulations using a LifeV-based FSI implementation employing material models from the material library of FEAP. For instance, several Neo-Hookean type materials are available in FEAP, cf. Section 1.2.3. We use a customized version of FEAP (cf. Section 1.3.2), in which also the anisotropic polyconvex hyperelastic and the anisotropic viscoelastic material models described in Sections 1.2.4 and 1.2.5, respectively, are implemented. Since the framework of FEAP is well-engineered and, due to many users, well tested, the implementation of the material models in FEAP is very reliable.

Our implementation of the coupling is designed to meet the following targets:

- the use of the material models from the material library in FEAP,
- access to the material models from existing structural mechanics or fluid-structure interaction codes requiring only minimal changes,

- transfer of data between `LifeV` and `FEAP`,
- possible use of time dependent material models (e.g., the viscoelastic material model which is described Section 1.2.5), and
- common export of data.

We first introduce briefly the two main software packages, `LifeV` and `FEAP`, which form the basis of our FSI software. In particular, we concentrate on the parts of `LifeV` and `FEAP` which are important for our implementation. Figure 1.2 gives an overview of the most important software packages which are used within our FSI software.

Afterwards, we detail the main challenges of the implementation, namely:

- the initialization of `FEAP` within the FSI code,
- the transfer of material data and meshes,
- the structural assembly in `FEAP`,
- how the time stepping of FEAP is handled,
- the export of structural data, and
- the restart of simulations with the viscoelastic material model.

1.3.1 LifeV

According to the `LifeV` website [90, 92], `LifeV` is an open source library for the numerical solution of partial differential equations with the finite element method, which is distributed under the LGPL license. It is implemented in C++ and is entirely coded with an object-oriented approach and advanced programming features. The library includes solvers for incompressible fluid dynamics, (linear) structural problems, transport in porous media, fluid-structure interaction, and electrical conduction in the heart.

Note that this section is written on the basis of `LifeV` version 3.6.2, which has also been used to implement the coupling with `FEAP`.

As displayed in Figure 1.2, `LifeV` is based on many packages of the software library `Trilinos`; see Section 2.3.1 for a more detailed description of `Trilinos`. In particular, `LifeV` provides wrapper classes and interfaces for many `Trilinos` packages, e.g., the parallel linear algebra package `Epetra`, the linear solver packages `AztecOO` and `Belos`, the algebraic one-level overlapping Schwarz preconditioner `IFPACK` [173], the AMG preconditioner `ML` [98], and the parameter list tool `Teuchos::ParameterList`. In `LifeV`, mesh partitioning is

```
virtual void
setup(const FESpacePtr_Type& dFESpace,
      const ETFESpacePtr_Type& ETFESpace,
      const boost::shared_ptr<const MapEpetra>& monolithicMap,
      const UInt offset, const dataPtr_Type& dataMaterial,
      const displayerPtr_Type& displayer  ) = 0;

virtual void
computeLinearStiff(dataPtr_Type& dataMaterial,
                   const mapMarkerVolumesPtr_Type /*mapsMarkerVolumes*/,
                   const mapMarkerIndexesPtr_Type /*mapsMarkerIndexes*/ ) = 0;

virtual void
updateJacobianMatrix(const vector_Type& disp,
                     const dataPtr_Type& dataMaterial,
                     const mapMarkerVolumesPtr_Type mapsMarkerVolumes,
                     const mapMarkerIndexesPtr_Type mapsMarkerIndexes,
                     const displayerPtr_Type& displayer ) = 0;

virtual void
apply(const vector_Type& sol,
      vector_Type& res,
      const mapMarkerVolumesPtr_Type mapsMarkerVolumes,
      const mapMarkerIndexesPtr_Type mapsMarkerIndexes) = 0;
```

Figure 1.3: The most important methods of the abstract class `structuralConstitutiveLaw`: the method `setup` initializes the fields of the class, the stiffness matrix for linear elastic material models is built in the method `computeLinearStiff` (if the material model is nonlinear, the method is empty), the Jacobian for nonlinear material models is assembled in the method `updateJacobianMatrix`, and the residual vector is built in the method `apply`.

performed using `ParMETIS` [123], by default, and the smart pointers from the `Boost` library are used to prevent leaking of memory.

However, `LifeV` also contains many original features for finite element simulations, such as

- boundary conditions,
- time discretizations,
- space discretizations,
- algebraic solvers and preconditioners,

- physical solvers,
- a geometrical multiscale framework, and
- useful tools, e.g., for import and export of data.

Most important for our purpose, `LifeV` contains a basic implementation of a monolithic FSI solver in 3D. This FSI code uses the structural and the fluid solver packages of `LifeV` in order to handle the corresponding subproblems. Additional features, such as classes for the handling of the nonlinear FSI block system (cf. Equation (1.15)), the framework for the handling of boundary conditions, the importer and exporter tools for postprocessing, and the time and space discretizations, which are provided by `LifeV`, are employed in the implementation as well. All settings of the simulation, e.g., the specifications of the time stepping or the space discretizations, are specified in a datafile.

We use `Trilinos IFPACK` [173] to approximate the inverse matrices of the blocks in the Dirichlet-Neumann preconditioner, as described in Section 1.1.3. The use of `IFPACK` is provided by the corresponding interface, which is implemented in `LifeV`.

The FSI implementation is fully MPI-parallel: the partition of the mesh, which is computed by `ParMETIS`, defines the parallel distribution of the `Trilinos` (i.e., `Epetra`) matrices and vectors. In particular, the fluid and the structural meshes are distributed separately: one fluid, one structural, and one subdomain of the geometry problem are assigned to each process.

The abstract class `structuralConstitutiveLaw` is of major relevance for our implementation of the coupling of the software packages `LifeV` and `FEAP`. The class defines a consistent interface for the implementation of structural material models in `LifeV`. In particular, the structural stresses, which are included in the right-hand side of the linearized system (1.17), and the structural Jacobian matrix, which is part of the tangent matrix (1.18), are implemented in this class. Among others, `LifeV` provides the classes `VenantKirchhoffMaterialLinear` (for linear elasticity) and `NeoHookeanMaterialNonLinear` (for a Neo-Hookaen material), which are specializations of the class `structuralConstitutiveLaw`. The material model and the corresponding parameters can also be specified within the datafile.

The most important methods of the class `structuralConstitutiveLaw` with respect to the implementation of a new material law are listed in Figure 1.3. The method `setup` initializes the fields of the material object such that the object can be used to assemble the residual vector (`apply`) and Jacobian matrix (`updateJacobianMatrix`). However, if the stresses are linear with respect to

```
COORdinates
1 0 0.0 0.0
2 0 1.0 0.0
3 0 0.0 1.0
4 0 1.0 1.0
```

Figure 1.4: Specification of the list of coordinates of a two-dimensional mesh. The command `COORdinates` is cut off after four characters, i.e., it is equivalent to the command `COOR`. The nodes of the mesh are $(0.0, 0.0)$, $(1.0, 0.0)$, $(0.0, 1.0)$, and $(1.0, 1.0)$.

Command	Description
`TANG`	Assembly of the tangent matrix.
`FORM`	Assembly of the residual vector.
`SOLV`	Solution of the system/update of the solution vector.
`TIME`	Progress in the time/load stepping.
`PROP,,1`	Initialization of the time/load stepping.
`PLOT,STRE,1`	Computation of the stresses.
`SAVE,a`	Saving actual state to a file with suffix ‘a’.
`REST,a`	Restarting from a file with suffix ‘a’.

Table 1.3: List of `FEAP` commands which are used for the implementation of the `LifeV-FEAP` coupling class `feapMaterial`. Using `libfw`, the command `SOLV` was modified such that no solution is computed by `FEAP` and the solution vector is only updated instead. The commands `SAVE,a` and `REST,a` are needed to perform the restart feature of `FEAP`. The additional parameter `a` defines the suffix of the corresponding state file.

the displacement (linear elasticity), the Jacobian matrix is assembled only once within the method `computeLinearStiff`.

In order to use `FEAP` within `LifeV` simulations (fluid-structure interaction or structure-only) with minimal changes to existing codes, we implemented a new material class, `feapMaterial`, which is derived from `structuralConstitutiveLaw`, in `LifeV`. To explain how this class is implemented, we first describe `FEAP` and the `FEAP` wrapper library `libfw` [89].

1.3.2 FEAP and libfw

`FEAP` (Finite Element Analysis Program) [193] is a software for finite element computations including an element library for solids, structures and thermal

```
 1  FEAP
 2  NUMNP NUMEL NUMMAT NDM NDF NEN
 3
 4  ELEMents
 5  INCLude, Elements.e
 6
 7  COORdinates
 8  INCLude, Coordinates.c
 9
10  MATE 1
11   SOLId
12   ELAStic
13   NEOHook 1000 0.49
14
15  END
```

Figure 1.5: Example of an input file for FEAP: The command FEAP starts the input file (line 1), followed by the specification of the number of nodal points (NUMNP), the number of elements (NUMEL), the number of material property sets (NUMMAT), the space dimensions of the mesh (NDM), the maximum number of unknowns per node (NDF), and the maximum number of nodes per element (NEN) in line 2. Next, with the command ELEM, the list of elements (in the file Elements.e) of the mesh (lines 4–5), and, with the command COOR, the list of coordinates (in the file Coordinates.c) of the mesh (lines 7–8) are specified. In the material block (lines 11–13), a Neo-Hookean material with the material parameters $E = 1\,000$ (Young's modulus) and $\nu = 0.49$ (Poisson's ratio) is defined. The manipulation of data ends with the command END (line 15).

analysis, solution algorithms for a wide range of applications, and graphical and numerical output capabilities; see [194].

We use a customized version of FEAP which is based on FEAP version 8.2. The customized version includes additional features and a larger library of material models, e.g., the material models described in Sections 1.2.4 and 1.2.5. In order to use FEAP as library instead of as an application, we make use of the FEAP wrapper library libfw which was implemented by Andreas Fischle; cf. his PhD thesis [89] for more details.

A parallel version of FEAP, which uses METIS and ParMETIS [123] for the mesh partitioning and PETSc [15, 16, 17] for the parallelization of the solution steps, is also available. However, we use the (customized) serial version of FEAP. Nonetheless, the implementation of the coupling should sustain the parallelism

```
fw_run_cmd_seq("BATCH\nTANG\nFORM\nEND\n\n");
```

Figure 1.6: Calling the commands `BATCH`, `TANG`, `FORM`, and `END` in `FEAP` using `fw_run_cmd_seq` from `libfw`. Consequently, `FEAP` switches to batch (non-interactive) mode, the tangent matrix (`TANG`) and the right-hand side (`FORM`) are assembled in a sequence, and the batch mode is stopped (`END`).

of the FSI code in `LifeV`. We discuss this issue and the implementation of the parallelism in Section 1.3.3.

`FEAP` is typically used from the command line in an interactive mode, using commands containing a maximum of four characters with possible additional parameters. If the input is a command with more than four characters, the remaining characters are neglected; cf. Figure 1.4. In Figure 1.4 the list of coordinates of a mesh is specified. This can be performed from the command line in interactive mode or, alternatively, the list of coordinates can be read from a file on the hard drive. This file containing the list of coordinates can be specified analogously to the lines of code shown in Figure 1.5. In this figure, an example for an input file, which only specifies the setting of the finite element computation, is shown. However, the whole program flow of a simulation (including assembly, solution, and output) could be prescribed in the file as well. The input file can be used in addition or instead of the interactive mode.

To implement the coupling, permanent access to the functions and the data of `FEAP` from the FSI code at execution time is necessary. This is not naturally supported by `FEAP`. The `FEAP` wrapper library `libfw` provides classes and functions which make it possible to control `FEAP` and to use the data of `FEAP` from the code of the user. As a result, `libfw` facilitates the use of `FEAP` similarly to a library. For instance, in Figure 1.6, the execution of `FEAP` commands within some user code is depicted. In particular, `libfw` simulates the input of the commands to the command line in interactive mode.

The most important `FEAP` commands which are necessary to use `FEAP` for the structural assembly within the FSI simulation are shown in Table 1.3. Note that the solver functionality in `FEAP` was removed. Instead `SOLV` updates the solution vector. Since we want to use `FEAP` only for the structural assembly, the very limited set of commands in Table 1.3 is sufficient for our purpose. In addition, we use input files to specify the mesh and the material model at the initialization of the FSI simulation; cf. Figure 1.5.

	Proc 0	Proc 1	Proc 2
Input file	`I0`	`I1`	`I2`
Point-list file	`I0.c`	`I1.c`	`I2.c`
Element-list file	`I0_0.e`	`I1_0.e`	`I2_0.e`

Table 1.4: Files necessary for using separate instances of `FEAP` on 3 processes (one subdomain/instance assigned to each process): one input file, one file containing the list of elements of the local mesh, and one file containing the list of coordinates of the local mesh are needed for each instance of `FEAP`. The name of the file containing the list of elements is chosen such that multiple files could be specified on one process, e.g., to support multiple subdomains per process.

The coupling library `libfw` also provides the functionality to access data from the internal data fields of `FEAP`, e.g., the solution, the stresses, or even the dimension of the solution vector. This is essential in the implementation of the coupling to transfer data between `LifeV` and `FEAP`.

1.3.3 Implementation of the Coupling

As already stated in Section 1.3.1, we implemented the coupling in form of the class `feapMaterial`, which is a specialization of the abstract class `structuralConstitutiveLaw`. We will now explain the implementation of the coupling in detail.

The fact that `FEAP` is not MPI-parallel, in contrast to the FSI code in `LifeV`, is very important for the design of the coupling. We solve this issue in the following way: the structural mesh is decomposed using `LifeV`, and only a local matrix and a local vector have to be assembled for each subdomain. The local tangent matrices and residual vectors can be assembled locally on each process/subdomain, using a serial instance of `FEAP`. As a consequence, we execute one serial instance of `FEAP` for each structural subdomain, and all communication is performed by `LifeV`.

Since the handling of `FEAP` is restricted to the four main methods of the abstract class `structuralConstitutiveLaw`, cf. Figure 1.3, we do not have access to the whole state of the FSI simulation from within the structural material class. Thus, a small overhead to handle all possible states cannot be avoided.

1.3.3.1 Initialization of FEAP

First, the initialization of the `FEAP` instances has to be implemented. Typically, one would use either the constructor or the method `setup` of the class

```
fw_start_cmdl(argc_libfw, argv_libfw);
```

Figure 1.7: Execution of `FEAP`: as usual, an argument count and an argument vector are specified at the start of the execution of `FEAP`. In particular, the name of an input file (cf. Figure 1.5) should be specified here; e.g., for the input file `File`, the string `-iFile` has to be given to `FEAP`.

`feapMaterial` to initialize `FEAP`, cf. Figure 1.3 (methods of the abstract class `structuralConstitutiveLaw`). This is not practical since, for the existing `LifeV` implementations (e.g., the FSI code), data necessary at the time of creation of the material object, e.g., the local subdomain meshes, is not available yet.

In order to start an instance of `FEAP`,

- the input file, cf. Figure 1.5,
- the point-list file, and
- the element-list file

are needed.

The first execution of a method of the material class, once all data for the initialization of `FEAP` is available, is the first execution of `computeLinearStiff`. Thus, we implement the initialization of `FEAP` at the beginning of this method. The method `computeLinearStiff` computes the Jacobian matrix for a linear elastic material model. It is important to note that the method `computeLinearStiff` is called for all types of material laws. However, it is empty for nonlinear materials.

Since the material type and the material parameters are already available when the `setup` method is called, we write the input files to the hard disk at this point. Due to the fact that we start one `FEAP` instance on each process (corresponding to one subdomain), we write one input file, one coordinate-list file, and one element-list file per process to the hard disk; see Table 1.4. The file names are chosen such that one file name uniquely corresponds to one `FEAP` instance (i.e., one process). More than one subdomain per process is not supported by `LifeV` at the moment; however, our naming convention, which is explained in Table 1.4, supports many subdomains per process.

One serial instance of `FEAP` is launched using `libfw`, as shown in Figure 1.7, where `argc_libfw` and `argv_libfw` are the input arguments given to `FEAP`.

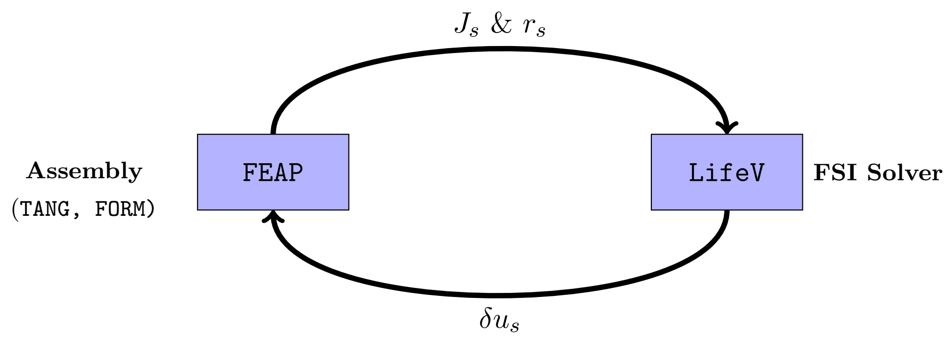


Figure 1.8: Software flow and transfer of the structural data between `LifeV` and `FEAP`: the update of the structural displacement δu_d is transferred to `FEAP` where the tangent matrix J_s and the residual vector r_s are assembled. Then, J_s and r_s are transferred back and used in the next iteration in the FSI code in `LifeV` to compute the next update δu_s.

In particular, the input file corresponding to the serial instance of `FEAP` is specified here. In `computeLinearStiff`, we also set up the time stepping in `FEAP`. Note that a time stepping in `FEAP` is only needed for, e.g., viscoelastic material models, cf. Section 1.2.5, where the stresses in the actual time step depend also on quantities at previous time steps. The time stepping for the FSI simulation is, however, handled by `LifeV`.

Directly after initialization of `FEAP` we assemble the Jacobian, in the case of linear elasticity; otherwise we leave the method `computeLinearStiff` doing nothing.

1.3.3.2 Structural Assembly in FEAP

The most crucial part of the coupling is the handling of the structural assembly in `FEAP`. More precisely, the assembly of the tangent matrix and the residual vector for the structure is implemented inside the methods `updateJacobianMatrix` and `apply`, respectively, in `LifeV`. In both cases, we first update the actual displacement in `FEAP`. As can be observed in the list in Table 1.3, this is performed using the command `SOLV`.

`FEAP` expects at least one assembly of the Jacobian matrix (`TANG`) and of the residual vector (`FORM`) before `SOLV` is called for the first time. This makes sense if `SOLV` is actually used to solve the corresponding linear equation system. However, as already mentioned, we only update the solution when calling `SOLV`, cf. Figure 1.3. Thus, if the simulation is started with an initial displacement, which is not equal to zero, `TANG` and `FORM` have to be called once, before the

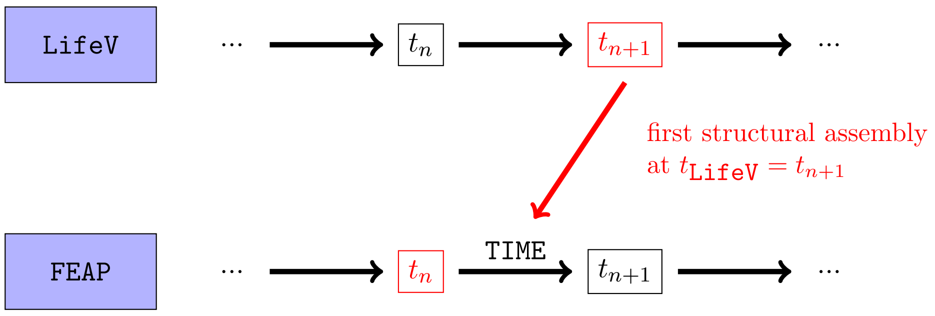


Figure 1.9: Handling of the time stepping of `FEAP` and `LifeV`: in the first structural assembly performed by `FEAP` within a new time step the internal times of `LifeV` and `FEAP` differ, i.e., $t_{n+1} = t_{\text{LifeV}} = t_{\text{FEAP}} + \delta t$. The command `TIME` is called in `FEAP` to proceed to the next time step.

solution can be updated using `SOLV`. This is typically the case if the simulation is not started at rest, e.g. when a previous simulation is restarted.

In Figure 1.8, the coupling is presented schematically. The structural assembly is performed by `FEAP`, whereas the FSI solver is part of `LifeV` (i.e., the time stepping, the Newton iteration in each time step, and the GMRES iteration).

Since there is one independent instance of `FEAP` on each process, the assembly of the global Jacobian and of the residual vector as well as the restriction of the global displacement to the local meshes have to be performed by the coupling class `feapMaterial` in `LifeV`. To transfer the data between `FEAP` and `LifeV`, we access the memory of `FEAP` using `libfw`.

In order to be consistent, the (Jacobian) matrices and the (displacement and residual) vectors, have to be permuted at this point. This is necessary since the degrees of freedom are sorted differently in the two software packages. While in `LifeV`, the degrees of freedom are ordered dimension-wise (i.e., all x coordinates first), the degrees of freedom are ordered node-wise in `FEAP`.

1.3.3.3 Time Stepping

For some material models, the stresses at the current time depend also on data from the previous time steps, in addition to the current displacement. Therefore, to handle such material models, a time stepping has to be used in `FEAP` as well. In particular, this is necessary for the use of the viscoelastic material model in Section 1.2.5.

Since, from the software point of view, the time stepping in `LifeV` and `FEAP` are independent from each other, they have to to be synchronized by

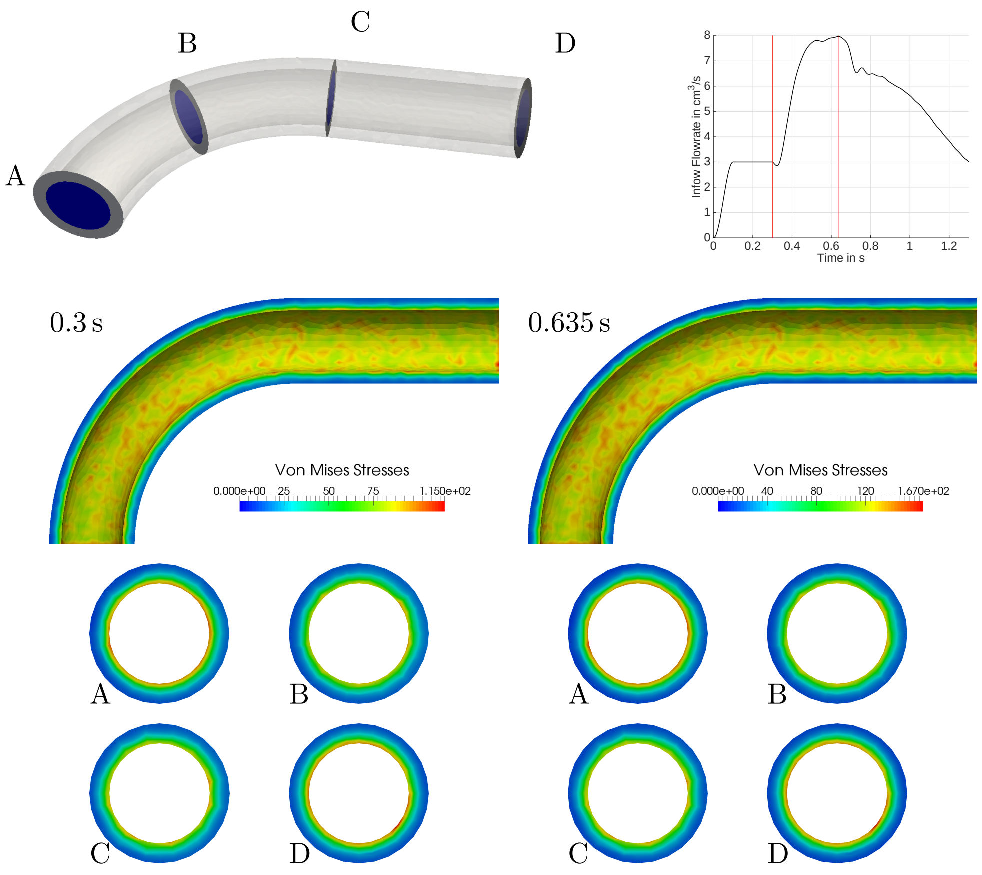


Figure 1.10: Exported von Mises stresses using $\bar{\text{F}}$ elements in a curved tube at two different points of time during the simulation of a heartbeat, cf. Section 1.5.2.1.

`feapMaterial`. Note also that the time discretization schemes used in `LifeV` and `FEAP` may be completely different.

The time stepping is handled in the following way: after the actual displacement is updated in `FEAP` (i.e., after calling `SOLV` within `updateJacobianMatrix` or `apply`), we compare the actual time of `LifeV` t_{LifeV} and of `FEAP` t_{FEAP}. Since we compare the times in each assembly, in fact, only two cases are possible. Either $t_{\text{LifeV}} = t_{\text{FEAP}}$ or $t_{\text{LifeV}} = t_{\text{FEAP}} + \delta t$. The latter is the case if `LifeV` has advanced to the next time step since the last structural assembly in `FEAP`. Then, we call the command (`TIME`), such that also `FEAP` advances in the time stepping; see Figure 1.9.

1.3.3.4 Common Export of Data

`LifeV` provides importer and exporter tools which are used for post processing purposes as well as for restarting simulations. In each time step, the solution vector is exported and can be opened with visualization tools (e.g., `Paraview` [4, 108]) or used by the FSI application to perform a restart of the simulation at a specific time step.

In addition to the solution at each time step, also other data, based on the nodes of the finite element mesh, can be exported, e.g., the structural stresses. For our purpose, they are of particular importance; however, they are computed at the Gauss points by `FEAP`. Thus, in order to export the stresses in `LifeV`, they are first interpolated to the nodes of the finite element mesh and then transferred to `LifeV`. Finally, the stresses interpolated at the nodes of the mesh can be exported using the exporter tool of `LifeV`.

Since the export feature is typically not accessible from the material class (`feapMaterial`), minor changes had to be applied to the existing FSI code in order to add the vectors containing the stresses to the exporter object, and to compute and to interpolate the stresses in `FEAP` before performing the export.

In Figure 1.10, an example for the distribution of the von Mises stresses in a curved tube is depicted; see also Section 1.5.2.3. The stresses have been transferred from `FEAP` to `LifeV`, exported, and visualized using `Paraview` [4, 108]. Note that exporting the stresses at each time step can be very memory demanding.

1.3.3.5 Restart of FSI Simulations for the Viscoelastic Material Model

To restart FSI simulations in which the viscoelastic material model (cf. Section 1.2.5) is used to model the arterial wall, the stresses (at the Gauss points) at the previous time step have to be exported. This is because they are needed to compute the stresses at the current time step. However, the exporter of `LifeV` only allows the export of node-based data.

Thus, we employ the restart functionalities of `FEAP` and of `LifeV` separately: the structural part of the restart is handled by `FEAP`, the fluid and the geometry parts of the restart are handled by `LifeV`. In particular, we use the commands `SAVE` and `REST` to perform the restart in `FEAP`; cf. Table 1.3. Therefore, in the coupling class, we call the `SAVE` command directly after proceeding to a new time step to store the data to the hard disk; see Section 1.3.3.3.

Since, analogously to the export feature, the restart functionality of `FEAP` and `LifeV` cannot be accessed from within the coupling class `feapMaterial`, the existing `LifeV` code had to be modified slightly to enable the restart of `FEAP` by the class.

1.4 Benchmark Settings

In this section, we describe the benchmark problem which has been introduced in [24] and has been used in our simulations. Its geometry corresponds to a curved pipe mimicking a tract of an artery, and it includes an initialization ("ramp") phase and a heartbeat phase. The ramp phase is needed in order to inflate the curved geometry, and the second phase includes the periodic application of the inflow profile of a human heartbeat. First, we provide detailed information about the geometry and the corresponding meshes, the material parameters, and the boundary conditions of our boundary value problem.

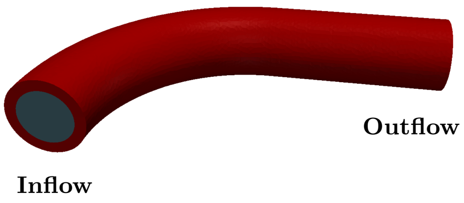
In order to be able to interpret the results, we define an idealized geometry and simplified boundary conditions, which still show characteristic results. As we focus on the numerical analysis of sophisticated arterial wall models in this thesis, the boundary value problem is constructed such that effects due to nonlinearities are revealed and the importance of the use of such material models in general applications is highlighted. We also describe the temporal and spatial discretizations used in our simulations.

1.4.1 Geometry

In Figure 1.11, the benchmark geometry and its dimensions are shown. It consists of a curved and a straight section and can be regarded as an idealized coronary artery. We have already used a similar geometry in [23]. As in [23], we restrict ourselves to just one material layer, the media, whereas the wall thickness is chosen according to realistic arteries here. Note that this represents a simplification since healthy arteries consist of mainly three layers, the intima, the media, and the adventitia, cf. [113]. Here, however, we are interested in an easily reproducible benchmark problem and, thus, focus on the mechanically most relevant layer, the media. In order to obtain realistic transmural wall stresses, also the other layers, i.e., the intima and the adventitia, for atherosclerotic arteries also the plaque components, would have to be taken into account.

1.4.2 Material Parameters

For the modeling of the media, we use the hyperelastic and the viscoelastic material models which are described in Sections 1.2.4 and 1.2.5. We choose the material parameters from [41, 20, (Ψ_A Set 2)], cf. Table 1.5, for the hyperelastic material model. These parameters are fitted to the material response of the media of a human abdominal aorta. For the viscoelastic material, we use two



Inner radius of the structure	0.15 cm
Outer radius of the structure	0.21 cm
Radius of curved part	1.0 cm
Length of straight part	1.0 cm

Figure 1.11: Geometry of the FSI problem. Copyright © 2015 John Wiley & Sons, Ltd.

different parameter sets, cf. Table 1.6. Note that Set 2 in Table 1.6 has a significantly reduced relaxation time in order to show viscoelastic effects more clearly; cf. Section 1.2.5.

c_1 [kPa]	ε_1 [kPa]	ε_2	α_1 [kPa]	α_2
17.5	499.8	2.4	30 001.9	5.1

Table 1.5: Parameters for the hyperelastic material model; see Section 1.2.4.

Set	c_1 [kPa]	ε_1 [kPa]	ε_2	α_1 [kPa]	α_2	τ_1	β_1
1	17.5	499.8	2.4	30 001.9	5.1	2.0	1.0
2	17.5	499.8	2.4	30 001.9	5.1	0.3	1.8

Table 1.6: Parameters for the viscoelastic material model: long relaxation time (Set 1), and short relaxation time (Set 2); see Section 1.2.5.

1.4.3 Time Discretization

For the simulations of our benchmark test, we use the Convective Explicit (CE) time discretization scheme [56], as already mentioned in Section 1.1.2, while the temporal derivatives are approximated using a second-order backward differentiation formula (BDF-scheme). The time discretization scheme and the composed Dirichlet-Neumann preconditioner are described in Sections 1.1.2 and 1.1.3. Due to the semi-explicit treatment of the convective term and the fast dynamics of our solution, in general, we have to use very small time steps. We thus first use a time step $\Delta t = 10^{-4}$ s. As we are going to discuss later, it is possible to use larger time steps in the heartbeat phase of the simulation; see Section 1.5.2.1.

1.4.4 Space Discretizations

In Table 1.7, all combinations of space discretizations which have been used within our FSI simulations are listed. The short names are introduced to distinguish between the different space discretizations more easily.

Short Name	Fluid (velocity–pressure)	Structure	Geometry
"P1"	P1-P1 stabilized	P1	P1
"P2"	P2-P1	P2	P2
"$\bar{\mathrm{F}}$"	P2-P1	$\bar{\mathrm{F}}$	P2

Table 1.7: Description of the space discretizations considered.

We remark that, when using the "P1"-approach, the fluid velocity and pressure are discretized by P1-P1 finite elements stabilized by interior penalty [45]. Finally, although the choice of the discretizations "P2" and "$\bar{\mathrm{F}}$" would lead to isoparametric meshes, we keep straight tetrahedral elements in our computational grids.

1.4.5 Meshes

Mesh	#Fluid elements	#Structural elements	Total Dofs "P1"	Total Dofs "P2"	Total Dofs "$\bar{\mathrm{F}}$"
#0	2 404	12 348	-	96 285	96 285
#1	6 549	21 636	30 880	186 658	186 658
#2	8 187	45 360	47 995	307 579	307 579
#3	12 670	98 742	88 670	590 555	590 555
#4	19 978	183 420	146 817	1 016 913	1 016 913
#5	40 011	274 500	230 713	-	-
#6	78 318	517 464	423 534	-	-
#7	179 513	1 036 800	871 323	-	-

Table 1.8: Degrees of freedom of the meshes used in the simulations. The internal variables in $\bar{\mathrm{F}}$ are condensed locally and thus not counted here.

In order to investigate whether or not the computed quantities converge for the different spatial discretizations listed in Section 1.1.2, cf. Table 1.7, eight different meshes are used; cf. Table 1.8. For P1 elements, these are the Meshes #1 to #7; for P2 and $\bar{\mathrm{F}}$ elements, the Meshes #0 to #4 are sufficient since Mesh #4 is already fine enough. For a detailed summary of the degrees of freedom (dofs), cf. Tables 1.9 and 1.10.

Note that, if we are only interested in fluid quantities, P1 elements for the structure can suffice if a comparatively high number of degrees of freedom is considered. For an accurate analysis of the structural stress distributions at

Mesh	Dofs $\mathbf{u}$	Dofs p	Dofs $\mathbf{d}_s$	Dofs $\boldsymbol{\lambda}$	Dofs $\mathbf{d}_f$
#1	5 430	1 810	14 664	3 546	5 430
#2	6 807	2 269	27 648	4 464	6 807
#3	10 545	3 515	57 096	6 969	10 545
#4	15 345	5 115	101 937	9 075	15 345
#5	27 777	9 259	152 295	13 605	27 777
#6	51 408	17 136	282 165	21 417	51 408
#7	113 175	37 725	564 252	42 996	113 175

Table 1.9: Degrees of freedom of the P1 meshes: fluid velocity ($\mathbf{u}$), fluid pressure (p), structural displacement ($\mathbf{d}_s$), coupling ($\boldsymbol{\lambda}$), and geometry/fluid mesh motion ($\mathbf{d}_f$).

Mesh	Dofs $\mathbf{u}$	Dofs p	Dofs $\mathbf{d}_s$	Dofs $\boldsymbol{\lambda}$	Dofs $\mathbf{d}_f$
#0	14 505	843	58 296	8 136	14 505
#1	34 368	1 810	101 808	14 304	34 368
#2	43 071	2 269	201 168	18 000	43 071
#3	66 648	3 515	425 700	28 044	66 648
#4	100 455	5 115	774 396	36 492	100 455

Table 1.10: Degrees of freedom of the P2 and $\bar{\mathrm{F}}$ meshes: fluid velocity ($\mathbf{u}$), fluid pressure (p), structural displacement ($\mathbf{d}_s$), coupling ($\boldsymbol{\lambda}$), and geometry/fluid mesh motion ($\mathbf{d}_f$). The internal degrees of freedom in the $\bar{\mathrm{F}}$ approach are statically condensated and are thus not considered.

a lower number of degrees of freedom, at least P2 or, even better, $\bar{\mathrm{F}}$ elements should be used to discretize the structure.

1.4.6 Boundary Conditions

In a realistic regime, coronary arteries are embedded in surrounding tissue, hindering the artery from moving freely in space when being under the influence of a pulsatile blood flow and pressure. Since we simulate only a section of a

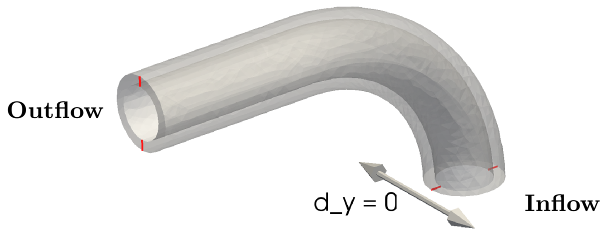


Figure 1.12: Dirichlet boundary condition at the inlet and outlet: fixed y-displacement for the red-colored nodes. Copyright © 2015 John Wiley & Sons, Ltd.

coronary artery neglecting the surrounding tissue, appropriate but artificial boundary conditions have to be applied to statically determine the idealized artery. In particular, we fix the structure in the directions perpendicular to the respective faces at the inlet and outlet, still allowing the artery to move in y-direction. In addition to that, we impose zero displacement in y-direction for all nodes at the inlet and outlet of the structure with $y = 0$; cf. the red lines in Figure 1.12.

The inflow boundary condition for the fluid is of particular importance because it is the driving force for our FSI simulation. In the ramp phase, we apply an increasing inflow flow rate. Applying a suitable inflow condition, together with an absorbing type boundary condition [158] at the outlet, allows to obtain a steady condition at the end of the ramp phase, for which the internal blood pressure is $p_{\text{steady}} = 80\,\text{mmHg}$ and the flow rate is $Q_{\text{steady}} = 3\,\text{cm}^3/\text{s}$; see Section 1.5.1. In that way, a prestretch of the arterial wall is generated, which is necessary as a starting configuration for a subsequent simulation of realistic heartbeats.

We call T_{R} the time when the inflow flow rate reaches its peak value of Q_{steady} and T_{steady} the time when the steady state is reached. There are many possible choices for the ramp function Q_{ramp} to increase the flow rate. They should, of course, satisfy the following conditions:

- $Q_{\text{ramp}}(0s) = 0.0\,\text{cm}^3/\text{s}$,
- $Q_{\text{ramp}}(t) = 3.0\,\text{cm}^3/\text{s} \qquad \forall T_{\text{R}} \leq t \leq T_{\text{steady}}$.

The time T_{steady} strongly depends on the special choice of T_{R} and Q_{ramp}. In Sections 1.5.1.1 and 1.5.1.2, we consider a linear- and a cosine-type ramp to model Q_{ramp}.

The inflow boundary conditions of both parts of the simulation, i.e., of the ramp and of the heartbeat phase, are imposed as Dirichlet boundary conditions for the fluid velocity. Precisely, we impose an inflow flow rate over time, i.e., Q_{ramp} for the ramp phase and $Q_{\text{heartbeat}}$ for the heartbeat phase, respectively.

The flow rate $Q(t^{n+1}) = Q^{n+1}$ at time t^{n+1}, being either Q_{ramp} (ramp phase) or $Q_{\text{heartbeat}}$ (heartbeat phase), is imposed at the inflow section of the fluid domain $\Gamma^f_{t,in}$, namely $\mathbf{u}^{n+1}(\mathbf{x})|_{\Gamma^f_{t,in}} = (0,\, 0,\, u_z(\mathbf{x}))$, with

$$u_z^{n+1}(\mathbf{x}) = \alpha^{n+1}\; \hat{u}_z \circ \mathcal{A}_t^{-1}(\mathbf{x}), \tag{1.64}$$

where

$$\hat{u}_z(\hat{\mathbf{x}}) = \frac{R^2 - \|\hat{\mathbf{x}} - \widehat{\mathbf{x}}_c\|^2}{R^2}. \tag{1.65}$$

This means that $\hat{u}_z$ is a parabolic profile defined on $\widehat{\Gamma}^f_{in}$, which is the inflow section of the fluid domain at time $t = 0\,\mathrm{s}$, i.e., in reference configuration. The variables R and $\widehat{\mathbf{x}}_c$ are the radius and the barycenter of $\widehat{\Gamma}^f_{in}$, while α^{n+1} reads

$$\alpha^{n+1} = \frac{Q^{n+1}}{\widehat{Q}^{n+1}}, \quad \text{with} \quad \widehat{Q}^{n+1} = \int_{\Gamma^f_{t,in}} \hat{u}_z \circ \mathcal{A}_t^{-1}(\mathbf{x}) \cdot \mathbf{n}_f \, d\gamma. \tag{1.66}$$

Indeed, these choices ensure that

$$\int_{\Gamma^f_{t,in}} \mathbf{u}^{n+1} \cdot \mathbf{n} = Q^{n+1}. \tag{1.67}$$

This essential property of the inflow boundary condition was not fulfilled for the implementation of the inflow boundary condition in [23], contributing to the very high oscillations observed in all measured quantities. Another cause of the high oscillations was the use of a very short linear-type ramp. As a result, the system was unfortunately prevented from reaching a steady state.

Our geometry only represents a section of an idealized coronary artery. Thus, to properly model the behavior of the solution at the artery outflow, an absorbing boundary condition is imposed at the outlet of the fluid. This is necessary to circumvent wave reflections at the outflow of our tube. The absorbing boundary condition internally builds on a lower-dimensional linear elastic material model. In our nonlinear setting, in general, we can therefore not expect this absorbing boundary condition to completely remove reflections since a linear elastic material model can barely approximate the material response of the highly nonlinear material models appropriately for the whole range of the FSI simulation (ramp and heartbeat phase). In the ramp phase, in order to reach at steady state, a desired pressure (here, a physiological value of 80 mmHg) in the fluid, we consider a modified version of the absorbing boundary condition proposed in [158]. At time t^{n+1}, the following absorbing boundary condition is enforced at the outflow section of the fluid domain:

$$\boldsymbol{\sigma}_f^{n+1} \cdot \mathbf{n}_f|_{\Gamma_{\text{out}}} = p_{\text{out}}^{n+1} \mathbf{n}_f|_{\Gamma_{\text{out}}}, \tag{1.68}$$

with

$$p_{\text{out}}^{n+1} = \left(\frac{\sqrt{\rho_F}}{2\sqrt{2}} \frac{Q_{\text{out}}^n}{\overline{A}} + \sqrt{\frac{dE}{1-\nu^2} \frac{\pi}{A^0} \sqrt{A^0}} \right)^2 - \frac{dE}{1-\nu^2} \frac{\pi}{A^0} \sqrt{A^0} + p_{\text{ref}}, \tag{1.69}$$

where E and ν are Young's modulus and Poisson's ratio of the underlying linear elastic material law (cf. Section 1.2.1). Furthermore, d is the thickness of the structure, p_{ref} is a reference pressure in the fluid, Q^n_{out} is the outflow flow rate at the discrete time t^n, and A^0 is the area of the outflow section of the fluid domain in its reference configuration.

In our setting, $\overline{A}$ is computed from (1.69) by imposing the steady state conditions, i.e., $p^{n+1}_{\text{out}} = p_{\text{steady}} = 80\,\text{mmHg}$ when $Q_{\text{out}} = Q_{\text{steady}} = 3\,\text{cm}^3/\text{s}$. Note that during the simulations of the heartbeats, the absorbing boundary condition (1.69) is modified according to [158], i.e., we use A^n_{out} instead of $\overline{A}$.

We remark that the values of Young's modulus E and Poisson's ratio ν are adjusted to the response of the nonlinear anisotropic hyperelastic material model described in Section 1.2.4. Here, only the material behavior in circumferential direction is considered for the adjustment of the linear elastic absorbing boundary condition, resulting in a Young's modulus of 120 kPa and a Poisson's ratio of 0.49. However, we do not observe any sensitivity of the absorbing boundary condition with respect to the Young's modulus; cf. also Section 1.5.1.5.

Although an absorbing boundary condition is used at the fluid outflow, minor oscillations remain at the end of the ramp; cf. Section 1.5.1.3 and Figure 1.19. However, from the results shown in Section 1.5.1.6, we are inclined to believe that the minor oscillations are physical, in the sense that they are simply an artifact of the somewhat frugal boundary conditions for the structure, rather than being an artifact caused by the imperfectly absorbed waves at the outflow. We discuss the details in Section 1.5.1.6.

These oscillations vanish over time, cf., e.g., Section 1.5.1.3. We consider ramps of different shapes, see Sections 1.5.1.1 and 1.5.1.2, and slopes, see Section 1.5.1.3, to further reduce oscillations. After reaching the steady state, in the second phase of the simulation, the inflow profile of a heartbeat, cf. Figure 1.26, is applied periodically; see Section 1.5.2.1.

1.5 Numerical Experiments

In this section, we present a detailed discussion of our results for the FSI benchmark problem, including mesh convergence studies, investigation of the boundary conditions, and comparisons of space discretizations and material models; the results are taken from [24]. If not stated otherwise, we choose an absolute tolerance of 10^{-7} as a stopping criterion for the Newton method, i.e., the Newton iteration is stopped when $\|r_n\|_\infty < 10^{-7}$, and a relative tolerance of 10^{-8} for the GMRES, i.e., the GMRES iteration is stopped when $\|r_n\|_2/\|r_0\|_2 < 10^{-8}$. With $\|\cdot\|_\infty$ and $\|\cdot\|_2$ we refer to the corresponding vector norms, and r_n denotes the residual in the correspond n-th iteration step. The complete set of our computational results can be found in Section 1.6 (see Figures 1.36 to 1.56), whereas in this section we just present those results relevant to our discussion.

1.5.1 Initiating Physiological Blood Pressure (Ramp Phase)

Before we start the simulation of heartbeats, we apply an interior blood pressure of 80 mmHg to the interior of the artery; see Figure 1.13. This is crucial since the resulting prestretch has a strong influence on the fluid-structure interaction during the heartbeat. We initiate a slowly increasing blood flow (ramp), driven by imposing a flow rate at the inlet, until an internal pressure corresponding to roughly 80 mmHg is reached. We refer to this part of the simulation as the ramp phase.

Since our goal is to reach a steady flow rate, any oscillations are unwanted, may they be physical or only numerical artifacts. The shape and slope of the ramp affects the magnitude of oscillations. Therefore, we first investigate the ramp phase of the FSI simulation with respect to different shapes and slopes of the ramp using different meshes and different finite element formulations. We expect to reach a steady state at a physiological blood pressure. This is the prestressed physiological configuration of the coronary artery that will be used for the simulation of heartbeats; see Section 1.5.2.1. We impose a flow rate at the inlet in the ramp phase featuring a parabolic inflow profile; see Figure 1.29. In the ramp phase, we choose a time step of 10^{-4} s.

Already in this phase, we observe differences between the different discretizations; see Section 1.5.1.2.

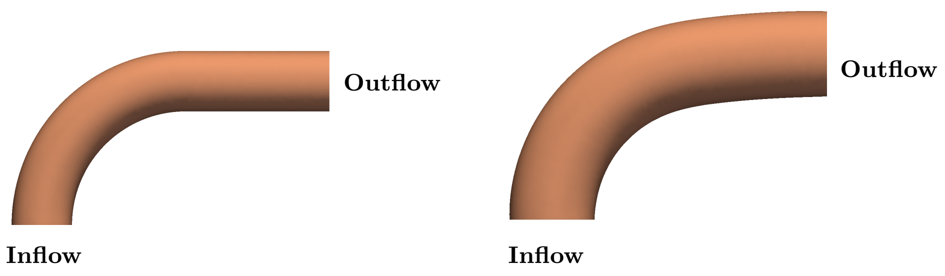


Figure 1.13: Geometry at a pressure of 0 mmHg (left) and at 80 mmHg (right); displacement scaled by a factor of three. Copyright © 2015 John Wiley & Sons, Ltd.

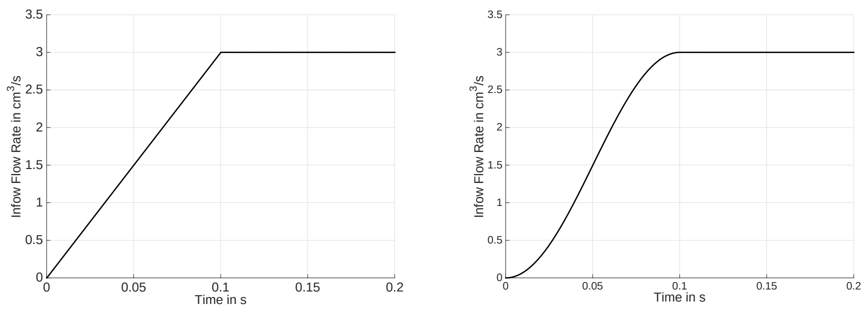


Figure 1.14: Linear (left) and cosine (right) type ramp with $T_\text{R} = 0.1\,\text{s}$. Copyright © 2015 John Wiley & Sons, Ltd.

1.5.1.1 Linear Ramp

We first choose a simple linear ramp according to

$$Q_\text{ramp}(t) = \begin{cases} \dfrac{t}{T_\text{R}} Q_\text{steady} & \text{for } 0 \leq t \leq T_\text{R}, \\ Q_\text{steady} & \text{for } T_\text{R} \leq t \leq T_\text{steady}, \end{cases} \tag{1.70}$$

cf. [23] and Figure 1.14 (left). In [23], a linear ramp with $T_\text{R} = 0.0177\,\text{s}$ was already considered. Relatively high oscillations in all measured quantities, namely the flow rate, average pressure, and the cross-sectional lumen area have been observed. Additionally, the amplitude of the oscillations have increased until the end of the simulation time for all depicted meshes. Thus, even though mesh convergence was observed, no steady state was reached in [23] within the simulation time.

Here, we consider seven different meshes, i.e., Mesh #1 to Mesh #7, cf. Table 1.8 and Table 1.9, with an increasing number of degrees of freedom. As we use unstructured meshes, we should note that the ratio of finite elements (and degrees of freedom) from one mesh to the next mesh is not constant. In

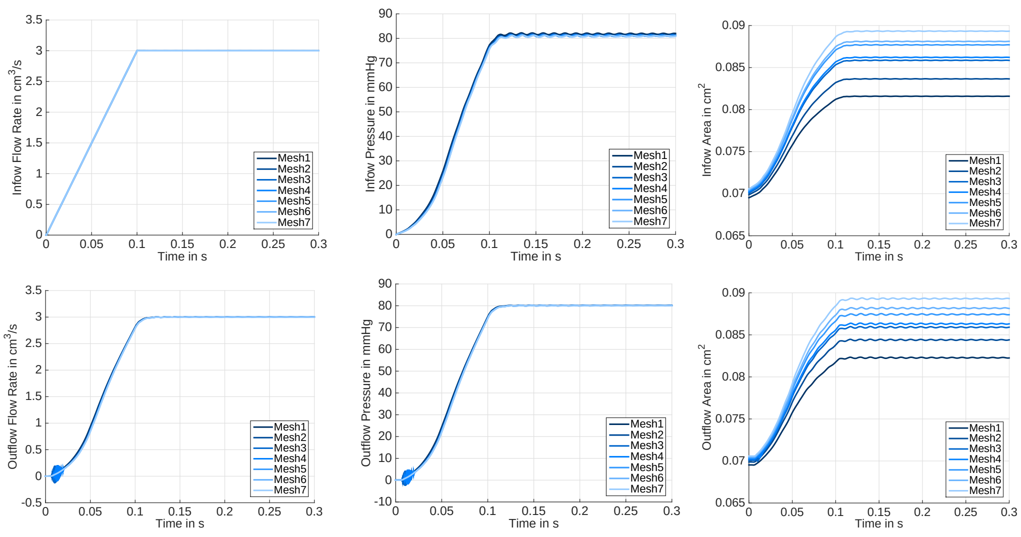


Figure 1.15: P1 mesh convergence study for the hyperelastic material using the linear ramp with $T_{\mathrm{R}} = 0.1\,\mathrm{s}$. Flow rate (left), average pressure (middle), cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom). Copyright © 2015 John Wiley & Sons, Ltd.

this section, we restrict ourselves to P1 elements for a comparison with the results in [23].

With respect to [23], T_{R} is increased and the inflow boundary condition was corrected. In [23], the change in the inflow cross-sectional lumen area during the simulation was not accounted for the inflow profile, and this led to an incorrect flow rate profile over the cross-sectional lumen area. Additionally, the absorbing boundary condition has been adjusted, such that an outflow pressure of 80 mmHg is reached at the outflow.

We measure the flow rate, the average pressure, and the cross-sectional lumen area at the inflow and the outflow; cf. Figure 1.11. The results of our simulations are presented in Figure 1.15. Compared to the results in [23], oscillations have been reduced considerably, and now they decrease during the simulation time. Besides, a pressure of approximately 80 mmHg, which is imposed by the absorbing boundary condition, is reached at the outflow, see Figure 1.15 (bottom, middle), while small but clearly visible oscillations are apparent.

As expected, the pressure at the inflow, see Figure 1.15 (top, middle), is slightly higher than the outflow pressure, cf. Figure 1.16. In this figure, we also see that the frequency and amplitude of the oscillations in the pressure seem to be independent of the mesh size.

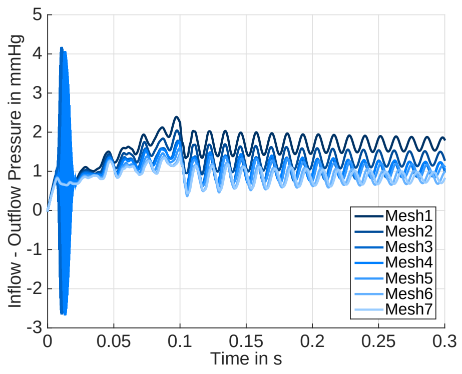


Figure 1.16: Inflow minus outflow pressure for the hyperelastic material using the linear ramp with $T_\mathrm{R} = 0.1\,\mathrm{s}$. Copyright © 2015 John Wiley & Sons, Ltd.

The inflow flow rate, which is imposed by a Dirichlet boundary condition, is now represented accurately in the simulation. For the outflow flow rate and pressure, see Figure 1.15 (bottom), small perturbations in the beginning of the ramp phase can be observed for the Meshes #2, #3, and #4. At the first glance, they could be caused by the fact that the linear ramp is starting with a steep slope, and thus the first time steps are very difficult to solve. As we discuss in Section 1.5.1.2, they indicate instabilities due to the P1 discretization, and are not caused by the shape of the ramp.

Similarly to the results in [23] the area is increasing when refining the mesh, but with much smaller oscillations. The cross-sectional lumen area, at the inflow as well as at the outflow, does not seem to converge when refining the meshes.

Within the simulation time of 0.3 s, which corresponds to 3 000 time steps, the oscillations decrease but do not vanish. Thus, the time until a steady state is reached would be significantly longer than 0.3 s when using a linear ramp inflow condition.

1.5.1.2 Cosine-Type Ramp

An alternative, better suited ramp is

$$Q_\mathrm{ramp}(t) = \begin{cases} \frac{1}{2} Q_\mathrm{steady}\left(1 - \cos\left(\frac{\pi}{T_\mathrm{R}} t\right)\right) & \text{for } 0 \le t \le T_\mathrm{R}, \\ Q_\mathrm{steady} & \text{for } T_\mathrm{R} \le t \le T_\mathrm{steady}; \end{cases} \tag{1.71}$$

cf. Figure 1.14. It is a $\mathcal{C}^1$-function which satisfies $\dot{Q}_\mathrm{ramp}(0) = 0\,\mathrm{cm}^3/\mathrm{s}^2$, meaning that the transition between the increasing and the constant part of the ramps is smooth, and that the difficulty of solving the first time steps is decreased.

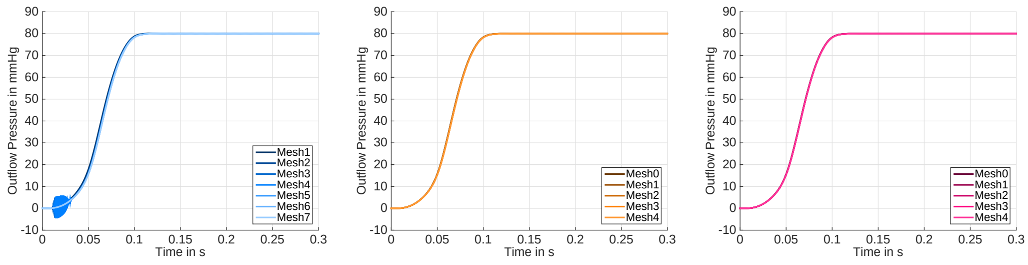


Figure 1.17: Outflow pressure for the hyperelastic material using the cosine-type ramp with $T_{\mathrm{R}} = 0.1\,\mathrm{s}$ for P1 (left), P2 (middle) and $\bar{\mathrm{F}}$ (right) elements. Copyright © 2015 John Wiley & Sons, Ltd.

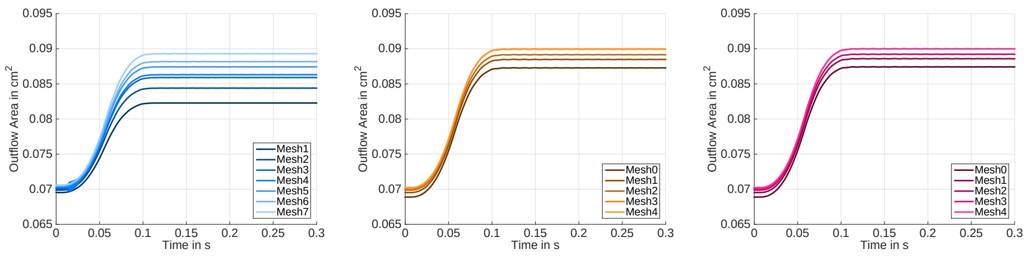


Figure 1.18: Mesh convergence study of the outflow cross-sectional lumen area for the hyperelastic material using the cosine-type ramp with $T_{\mathrm{R}} = 0.1\,\mathrm{s}$ for P1 (left), P2 (middle) and $\bar{\mathrm{F}}$ (right) elements. In this diagram, the graphs for Mesh #3 and Mesh #4 completely overlap for P2 and $\bar{\mathrm{F}}$ elements. Copyright © 2015 John Wiley & Sons, Ltd.

Let us first discuss the perturbations in the outflow quantities. Therefore, we again consider for P1 elements the Meshes #1 to #7 and for P2 and $\bar{\mathrm{F}}$ elements the five meshes, Mesh #0 to Mesh #4; cf. Tables 1.8 and 1.10. In Figure 1.17, the corresponding outflow pressure is displayed, and we can observe that the perturbations remain also for this improved type of ramp. However, for P2 (middle) and $\bar{\mathrm{F}}$ (right) elements, the perturbations in the beginning of the ramp vanish. This shows clearly an improvement due to the use of these discretizations.

Another purpose for the use of the cosine-type ramp is to reduce oscillations in the constant part of the ramp phase. In Figure 1.18, the outflow cross-sectional lumen area for P1, P2, and $\bar{\mathrm{F}}$ elements is displayed for different meshes. Comparing these results to those of Figure 1.15, on the visible scale we appreciate a substantial reduction of the amplitude of the oscillations. More precisely, the amplitude is reduced by one order of magnitude, cf. Table 1.11, using the cosine-type ramp instead of the linear one.

Mesh	Linear ramp P1	Cosine ramp P1	Cosine ramp P2	Cosine ramp $\bar{\mathrm{F}}$
# 0	-	-	$2.7 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$
# 1	$1.1 \cdot 10^{-4}$	$3.6 \cdot 10^{-6}$	$2.8 \cdot 10^{-5}$	$2.8 \cdot 10^{-5}$
# 2	$1.2 \cdot 10^{-4}$	$8.6 \cdot 10^{-6}$	$2.7 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$
# 3	$1.3 \cdot 10^{-4}$	$1.1 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$
# 4	$1.3 \cdot 10^{-4}$	$1.3 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$
# 5	$1.4 \cdot 10^{-4}$	$1.6 \cdot 10^{-5}$	-	-
# 6	$1.2 \cdot 10^{-4}$	$1.9 \cdot 10^{-5}$	-	-
# 7	$1.2 \cdot 10^{-4}$	$2.0 \cdot 10^{-5}$	-	-

Table 1.11: Amplitude of the oscillations of the outflow cross-sectional lumen area in cm^2 at $t = 0.2\,\mathrm{s}$.

In order to investigate mesh convergence, we again consider Figure 1.18. As in Figure 1.15, mesh convergence cannot yet be observed for P1 elements (left). On the other hand, the results for P2 (middle) and $\bar{\mathrm{F}}$ (right) finite elements do suggest mesh convergence. Moreover, P2 and $\bar{\mathrm{F}}$ finite elements show a very similar behavior in Figure 1.17 and Figure 1.18, and there is a difference to P1 elements as can be seen in the latter figure. We further discuss this in Section 1.5.1.4.

Based on our findings in this section, from now on, we proceed using a cosine-type ramp to reduce oscillations. Next, we investigate whether we can further decrease T_{R}.

1.5.1.3 Steepness of the Ramp

In this section, we consider three different slopes of cosine-type ramps, i.e., $T_{\mathrm{R}} \in \{0.05\,\mathrm{s}, 0.1\,\mathrm{s}, 0.2\,\mathrm{s}\}$, in order to study the sensitivity. We again consider the oscillations of the outflow cross-sectional lumen area, and restrict ourselves to Mesh #1 for P2 elements. In addition to the hyperelastic material model, we also consider the viscoelastic material model, cf. Section 1.2.5. The complete set of results regarding the steepness of the ramp can be found in Section 1.6, including results for P1 elements.

We remark that the relaxation time of the viscoelastic material model is longer than 2 s, and thus it cannot be at steady state after 0.5 s. For the parameter Set 1 of Table 1.6, which has been used for the simulations, the overstresses are small and the relaxation time is long. Thus, viscoelastic effects are difficult to observe.

In Figure 1.19, oscillations are only visible for the shortest ramp. Until the end of the simulation time, i.e., 0.5 s, these oscillations are still visible. For both longer ramps, the oscillations are not visible at the scale presented, and thus

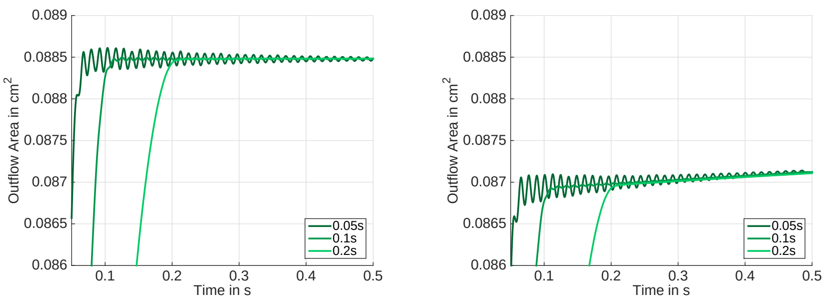


Figure 1.19: Outflow cross-sectional lumen area using P2 elements on Mesh #1. Hyperelastic (left), and viscoelastic (right) material model using parameter Set 2 from Table 1.6. Copyright © 2015 John Wiley & Sons, Ltd.

considered as acceptable. The qualitative behavior of the hyperelastic and the viscoelastic material is similar, however, we can observe that the cross-sectional lumen area is lower for the viscoelastic material and slowly increases due to creep behavior. We not focus on the differences of the material models now, but refer to the next section, in which we discuss the influence of viscoelasticity in the FSI simulations with regard to the chosen space discretization.

As a result of the discussions so far, we decide to use a cosine-type ramp with a length of $T_{\mathrm{R}} = 0.1\,\mathrm{s}$ from now on. After another $0.2\,\mathrm{s}$ of constant inflow flow rate we consider the system to be at steady state, i.e., $T_{\mathrm{steady}} = 0.3\,\mathrm{s}$, and thus ready for the subsequent simulation of a heartbeat. All further presented simulations were performed using this ramp.

1.5.1.4 Space Discretization and Viscoelastic Effects

As already mentioned in Section 1.5.1.2, our results suggest that P2 and $\bar{\mathrm{F}}$ discretizations should be favored over simple P1 finite elements.

In this section, we compare the different space discretizations in detail. Using the viscoelastic material model we observe that P1 elements do not only show disadvantageous approximation properties but also show a qualitatively incorrect behavior.

Thus, we first present in Figure 1.20 the results of simulations using the viscoelastic material model compared to the hyperelastic material model. In order to detect more clearly the viscoelasticity, we use parameter Set 2 from Table 1.6, which has a much shorter relaxation time compared to Set 1 from Table 1.6. In Figure 1.20, the expected behavior can be observed: if we impose the same pressure (left) as for the hyperelastic material model, the displacement is smaller in the beginning and converges to the displacement of the hyperelastic

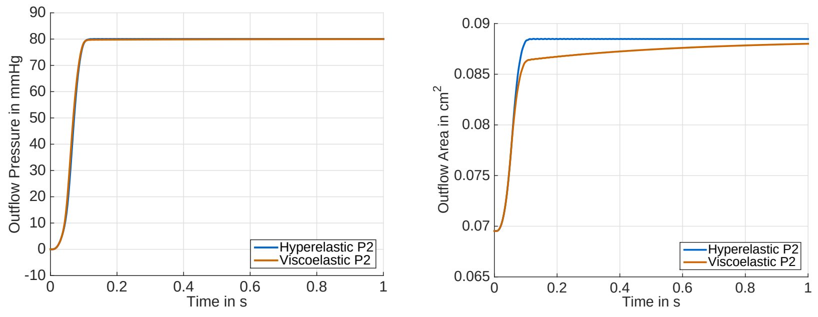


Figure 1.20: Comparison of the hyperelastic and the viscoelastic material for P2 elements using the cosine-type ramp for parameter Set 2 from Table 1.6 on Mesh #1. Outflow average pressure (left), outflow cross-sectional lumen area (right). Copyright © 2015 John Wiley & Sons, Ltd.

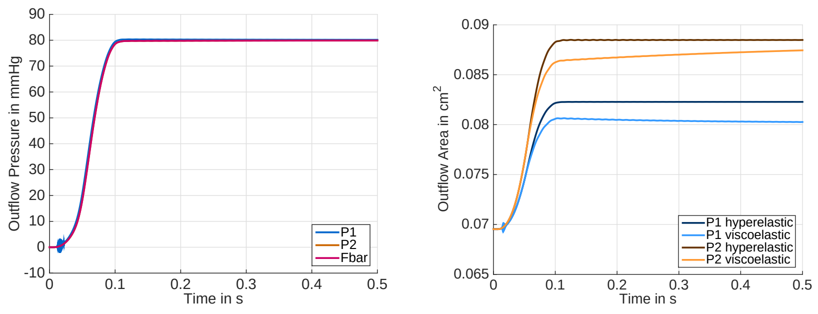


Figure 1.21: Viscoelastic material on Mesh #1, outflow pressure (left). Note that the curves for P2 and $\bar{\mathrm{F}}$ are almost identical. Comparison of the hyperelastic and the viscoelastic material model using the parameter Set 2 from Table 1.6 and P1 and P2 elements (right). Copyright © 2015 John Wiley & Sons, Ltd.

material model over time, cf. Section 1.2.5 and Figure 1.1. The displacement is here represented by the cross-sectional lumen area (right). This is caused by the creep behavior introduced in fiber direction, as described in Section 1.2.5.

In contrast to the appropriate results obtained with a P2 discretization, the use of a P1 discretization leads to a qualitatively wrong behavior, as can be seen in Figure 1.21: for a constant pressure (left), the displacement (right) decreases using P1 elements, while $\bar{\mathrm{F}}$ elements again yield the same correct results as P2 elements. Moreover, we observe that P2 elements yield the expected asymptotic behavior of the viscoelastic material model, while P1 elements do not.

We also observe that P1 elements lead to much smaller displacements than P2 or $\bar{\mathrm{F}}$ elements for the same mesh. This can be seen for the hyperelastic and also for the viscoelastic material model; cf. Figure 1.21 (right). We understand this behavior as the property of P1 elements to tend to locking effects and to be mechanically stiffer compared to P2 and $\bar{\mathrm{F}}$ elements.

These observations, in combination with the discussion in Section 1.5.1.2 about the perturbations which arise in the beginning of the ramp phase for P1 elements, are convincing arguments that P1 elements are not sufficient to describe the structural behavior accurately.

Still, it may be sufficient to consider P1 elements for the structure if a hyperelastic material is considered, provided that a very fine mesh is used, depending on the quantities under consideration. If, for instance, fluid quantities or the structural displacement are analyzed, a very fine P1 discretization may be sufficient, whereas the accuracy of the computed structural stresses strongly depends on the choice of an appropriate discretization; cf. Section 1.5.2.3.

1.5.1.5 Sensitivity to Parameters of the Absorbing Boundary Condition

A reason for the oscillations observed in Sections 1.5.1.1, 1.5.1.2, and 1.5.1.3 could be the absorbing boundary condition, see Section 1.4.6, which we apply at the outflow to remove wave reflections. This absorbing boundary condition is based on a one-dimensional linear elastic model. Since we use highly nonlinear material models here, it is not obvious if the absorbing boundary condition is able to completely remove wave reflections at the outlet, especially if the corresponding linear elastic material parameters are not chosen appropriately.

We investigate the influence of Young's modulus E in the absorbing boundary condition. In order to minimize the computational effort, we used P1 elements and Mesh #1 for these simulations.

As can be seen in Figure 1.22, neither the inflow pressure (left) nor the outflow cross-sectional lumen area (right) are influenced strongly by the varying Young's modulus. These measured quantities showed the strongest oscillations in the previous sections, which are however relatively small due to the particular choice of the ramp. This suggests that the remaining oscillations in the constant part of the ramp are not caused by the absorbing boundary condition.

1.5.1.6 Further Investigations on the Oscillations

Our structure is statically determined only from applying Dirichlet boundary conditions at both ends. The surrounding tissue is thus neglected. The inflow of fluid into the curved geometry may therefore excite a bending mode of the structure.

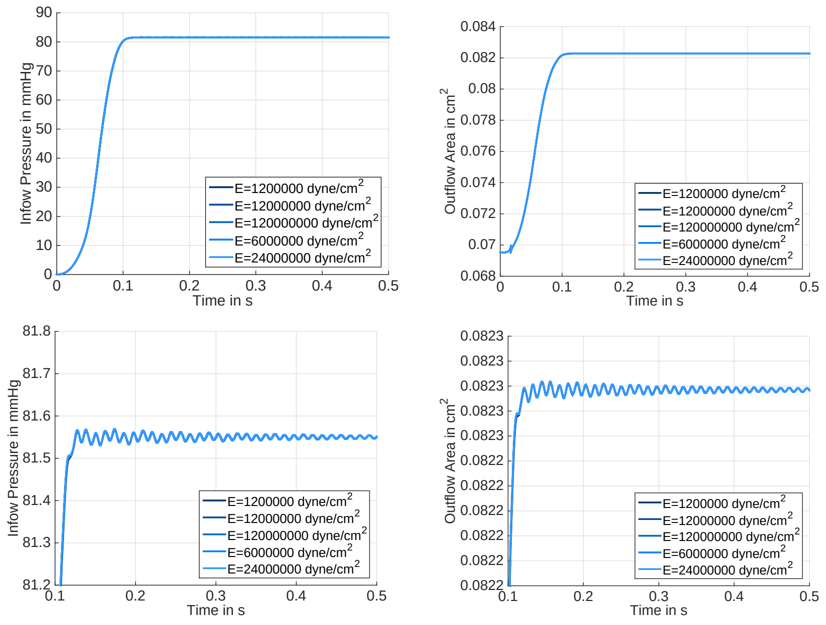


Figure 1.22: Sensitivity analysis for the absorbing boundary conditions, performed with P1 elements using Mesh #1: inflow average pressure (left), and outflow cross-sectional lumen area (right). All curves overlap completely. Global view (top), and zoom (bottom). Copyright © 2015 John Wiley & Sons, Ltd.

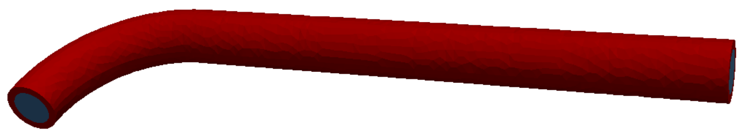


Figure 1.23: Curved tube with a longer straight part (4 cm). Copyright © 2015 John Wiley & Sons, Ltd.

We consider a new geometry, with a longer straight part of length 4 cm; cf. Figure 1.23. Details about the mesh can be found in Tables 1.12 and 1.13. We use a cosine ramp of length $T_{\mathrm{R}} = 0.05\,\mathrm{s}$.

As can be seen in Figure 1.24, there is indeed a strong bending of the tube over time.

In Figure 1.25, the outflow flow rate (left), inflow average pressure (middle), and outflow cross-sectional lumen area (right) are shown for P1 and P2 elements, comparing the geometry described in Figure 1.11 and the similar geometry with a longer straight section. As expected, the outflow flow rate of the long tube has a delay compared to the standard geometry, because it takes longer for the fluid wave to reach the outflow of the tube.

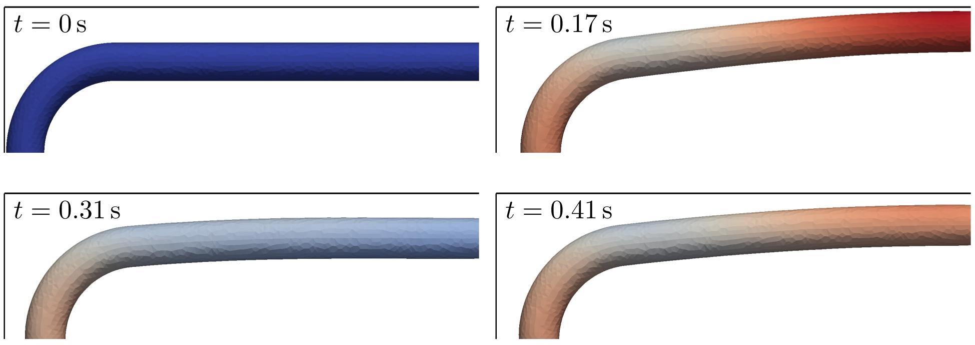


Figure 1.24: Bending of the long tube. Copyright © 2015 John Wiley & Sons, Ltd.

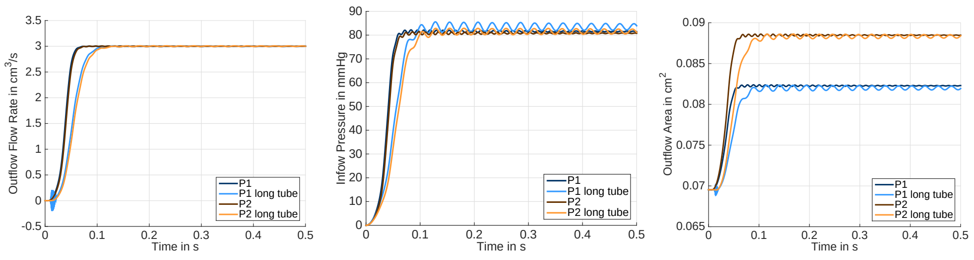


Figure 1.25: Comparison of the outflow flow rate (left), the inflow average pressure (middle), the outflow cross-sectional lumen area (right) of Mesh #1 and the corresponding mesh of the long tube, cf. Tables 1.12, 1.13, and Figure 1.23. Copyright © 2015 John Wiley & Sons, Ltd.

The oscillations arising in the inflow pressure and the outflow cross-sectional lumen area show a significantly larger amplitude and also a frequency which is roughly smaller by a factor of approximately 1/4. Thus, the oscillations may depend on the length of the geometry. The fact that the amplitude of the oscillations increases significantly could also be an indication that the source of oscillations is a bending mode.

Now that we have discussed the ramp, we believe that a reasonable way to carry out the simulation of the ramp phase, including the type and length of

#Fluid elements	#Structural elements	Total Dofs “P1”	Total Dofs “P2”	Total Dofs “$\bar{\text{F}}$”
2 404	12 348	64 999	391 693	391 693

Table 1.12: Degrees of freedom of the meshes corresponding to the long tube; see Figure 1.23.

Discretization	Dofs $\mathbf{u}$	Dofs p	Dofs $\mathbf{d}_s$	Dofs $\boldsymbol{\lambda}$	Dofs $\mathbf{d}_f$
P1	10 641	3 547	32 232	7 938	10 641
P2/$\bar{\mathrm{F}}$	65 745	3 547	224 784	31 872	65 745

Table 1.13: Degrees of freedom of the meshes corresponding to the long tube: fluid velocity ($\mathbf{u}$), fluid pressure (p), structural displacement ($\mathbf{d}_s$), coupling ($\boldsymbol{\lambda}$), and geometry/fluid mesh motion ($\mathbf{d}_f$); see Figure 1.23.

the ramp, and a reasonable space discretization have been found. In the next sections, we are going to concentrate on the simulation of heartbeats after the ramp phase.

1.5.2 Heartbeat Phase

1.5.2.1 Simulation of Several Heartbeats

We now present and discuss our results of the simulation of several full heartbeats. In advance of the heartbeat phase, the artery has been prestretched up to a physiological pressure of 80 mmHg. Referring to Sections 1.5.1.2 and 1.5.1.3, we choose a cosine-type ramp with length $T_{\mathrm{R}} = 0.1\,\mathrm{s}$ in order to inflate the tube. As already mentioned in the previous section, we continue the simulation at a constant inflow flow rate Q_{steady} until 0.3 s, and consider the system to be at steady state at this point of simulation time. We neglect some very small oscillations still remaining after 0.3 s; cf. also Figure 1.19 for $T_{\mathrm{R}} = 0.1\,\mathrm{s}$.

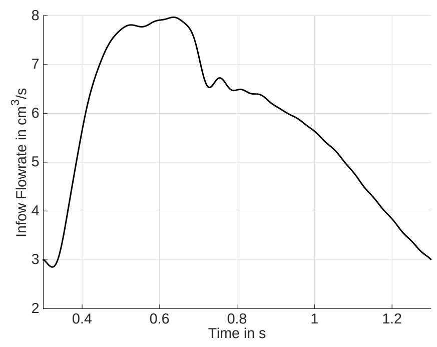


Figure 1.26: Inflow flow rate for the heartbeat phase. Copyright © 2015 John Wiley & Sons, Ltd.

Afterwards, we impose a flow rate profile $Q_{\mathrm{heartbeat}}$ over time according to Figure 1.26, which corresponds to a heartbeat. A typical pressure profile in coronary arteries is provided by [2, 204]. However, imposing a pressure or, more precisely, a normal stress at the inflow can lead to instabilities, which we indeed observed using fine meshes. These instabilities did not occur with Mesh #1. Therefore, from a full heartbeat simulation with Mesh #1, the resulting inflow flow rate over time was approximated by means of a Fourier series of order 20, and thus a periodic function $Q_{\mathrm{heartbeat}}$ was obtained. The resulting flow rate profile ranges from approximately $3\,\mathrm{cm}^3/\mathrm{s}$ in the diastolic phase, where the heart is at rest and the blood pressure is minimal, to approximately $8\,\mathrm{cm}^3/\mathrm{s}$ in the systolic phase, where the heart contracts and the blood pressure rises to the maximum value. In this way, the inflow flow rate results in a pressure (which mainly influences the stress distribution through the arterial wall) which follows a typical profile for a coronary artery.

The reason why the pressure and the flow rate profiles used in our simulations are not significantly different is that, in our model, we do not take into account

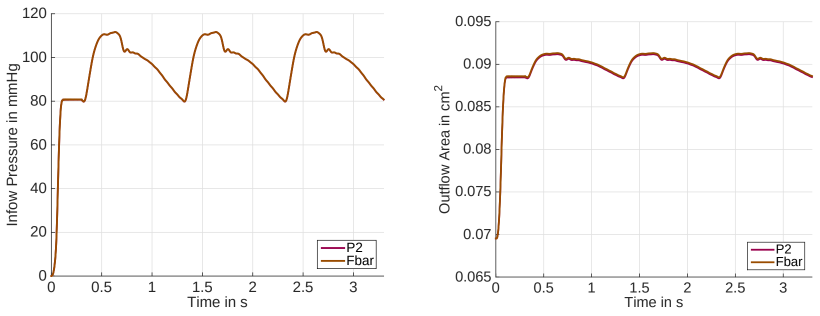


Figure 1.27: Simulation of 3 heartbeats using Mesh #1 and the hyperelastic material model: Inflow pressure (left), and outflow cross-sectional lumen area (right). Copyright © 2015 John Wiley & Sons, Ltd.

the forces exerted by the heart muscle on the coronary vessels. Nevertheless, the flow rate reported in Figure 1.26 is in rough accordance to the one of a right coronary artery, for which the systolic heart compression through the right ventricular myocardium has much smaller effect on the flow, compared to the influence which the left ventricular myocardium has on the left coronary [155, 48].

As already mentioned, the function $Q_{\text{heartbeat}}$ describes the inflow flow rate over time during each heartbeat, cf. Figure 1.26. Analogously to the inflow flow rate profile Q_{ramp}, used for the ramp phase, we use $Q_{\text{heartbeat}}$ as a Dirichlet boundary condition in the second part of the simulation.

During the simulation of a heartbeat, we choose a time step ten times larger than in the ramp phase, i.e., $\Delta t = 10^{-3}\,\text{s}$. In Figure 1.27, the average inflow pressure (left) and outflow cross-sectional lumen area (right) are presented for P2 and $\bar{\text{F}}$ elements. Similarly to the ramp phase, cf. Section 1.5.1.4, also during all three heartbeats, we observe a very similar behavior for both discretizations.

In Figures 1.28 and 1.29, the deformation and flow rate profile over time is depicted. The largest deformation is observed at the inner part of the curvature, near the inlet. Here, the tube deforms mainly due to the increasing flow rate at the inlet. Considering the flow rate, we observe that the flow is faster at the outer part of the curved pipe; see also Figure 1.30.

1.5.2.2 Mesh Convergence for the Heartbeat

In this section, we address the influence of the mesh refinement during the heartbeat phase, analogously to the discussions in Section 1.5.1.2 and Section 1.5.1.1 about the ramp phase.

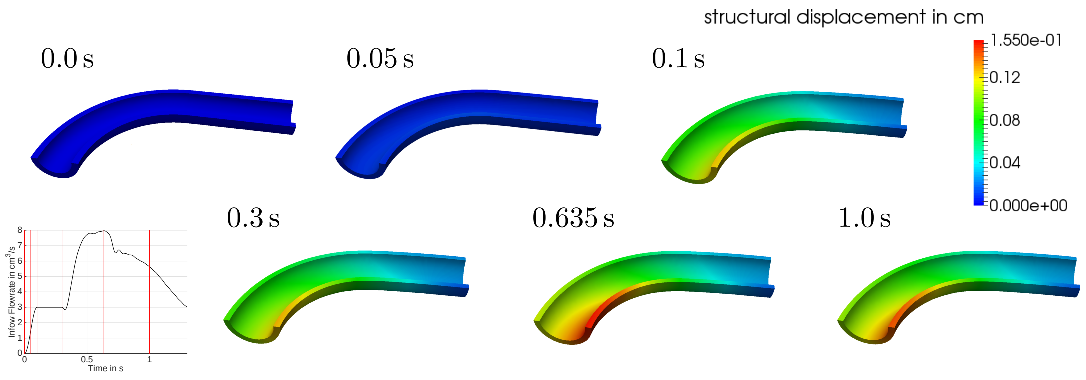


Figure 1.28: Evolution of the magnitude of the displacement of the structure for Mesh #3 and $\bar{\mathrm{F}}$ elements in the deformed configuration, at times 0.0 s (top left), 0.05 s (top middle), 0.1 s (top right), 0.3 s (bottom left), 0.635 s (bottom middle), and 1.0 s (bottom right). Displacement is scaled by a factor of 2.0. Copyright © 2015 John Wiley & Sons, Ltd.

In Figure 1.31, mesh convergence plots for P1 (left), P2 (middle), and $\bar{\mathrm{F}}$ (right) elements are shown. We show the results using Meshes #1 to #7 for P1 elements and using Meshes #0 to #4 for P2 and $\bar{\mathrm{F}}$ elements. For P2 and $\bar{\mathrm{F}}$ elements, Mesh #4 is only plotted until 0.6 s.

Considering the outflow area, for P1 elements, we do not see mesh convergence. For P2 and $\bar{\mathrm{F}}$ finite elements, the graphs for Mesh #3 and Mesh #4 overlap. With respect to the plots of the inflow pressure (top), there are small deviations for the different meshes because we do not impose a pressure anymore, in contrast to the ramp phase. This can be observed for all element types. The differences are clearly visible in the systolic phase of the heartbeat, whereas the inflow pressures are very close to each other for all meshes during the diastolic part. Thus, especially the maximum pressure decreases when refining the mesh. For the coarsest P1 element mesh, the maximum pressure is significantly higher than for the coarsest P2 or $\bar{\mathrm{F}}$ element mesh. We conclude that for P1 elements we are still far away from asymptotics, while for P2 and $\bar{\mathrm{F}}$ elements we have an indication of mesh convergence: the graphs for Mesh #3 and Mesh #4 completely overlap for all quantities.

In Figure 1.32, we present the number of Newton iterations for each time step as well as the sum of GMRES iterations in each time step, using Mesh #3 and $\bar{\mathrm{F}}$ finite elements. Improvements of the monolithic preconditioner and of the overlapping Schwarz method, to reduce the number of GMRES iterations, are reported in Chapter 3. Therein, we present results of FSI simulation using

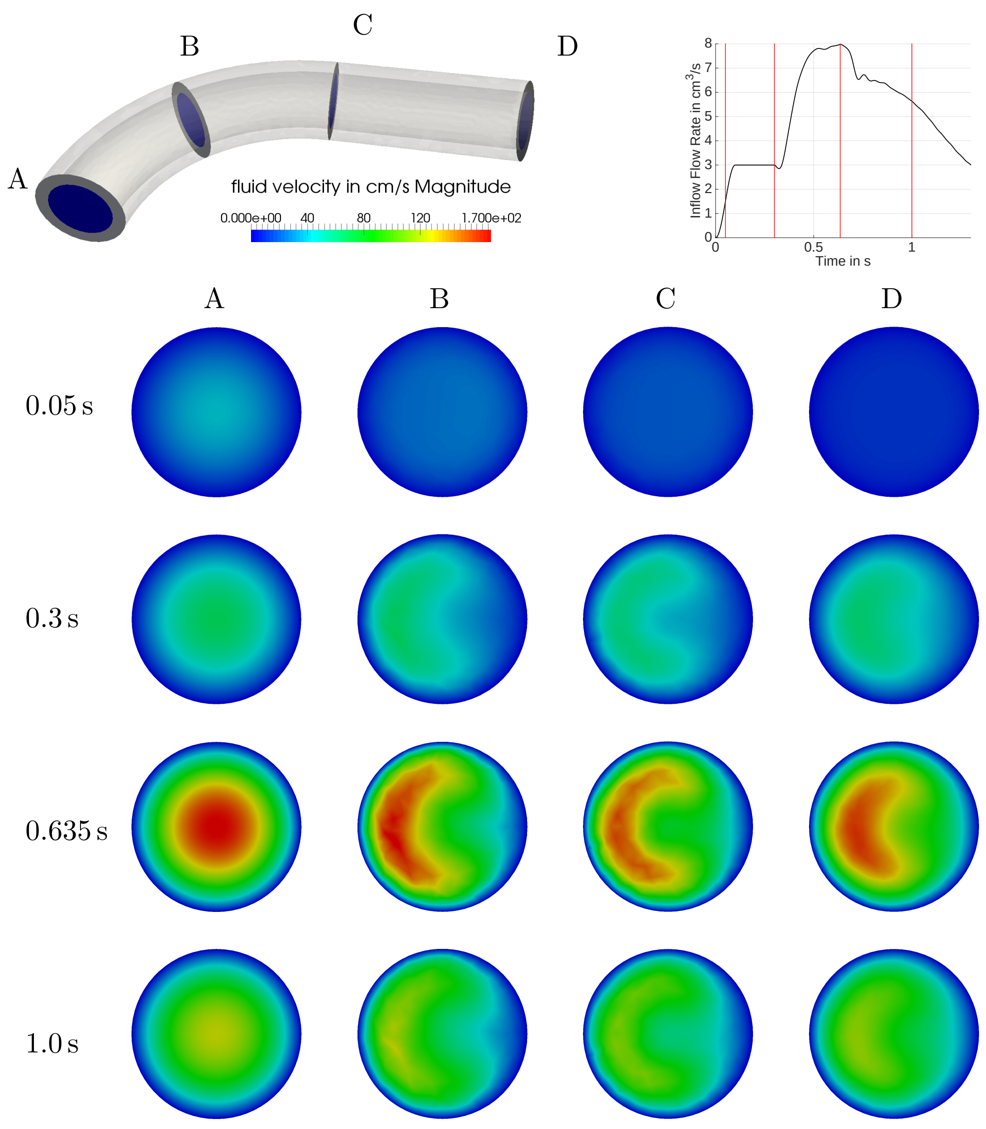


Figure 1.29: Evolution of the flow for Mesh #7 and P1 elements at different slices; cf. [24]. Copyright © 2015 John Wiley & Sons, Ltd.

the FaCSI preconditioner as a preconditioner for the monolithic matrix and the GDSW preconditioner as a preconditioner for the structural block.

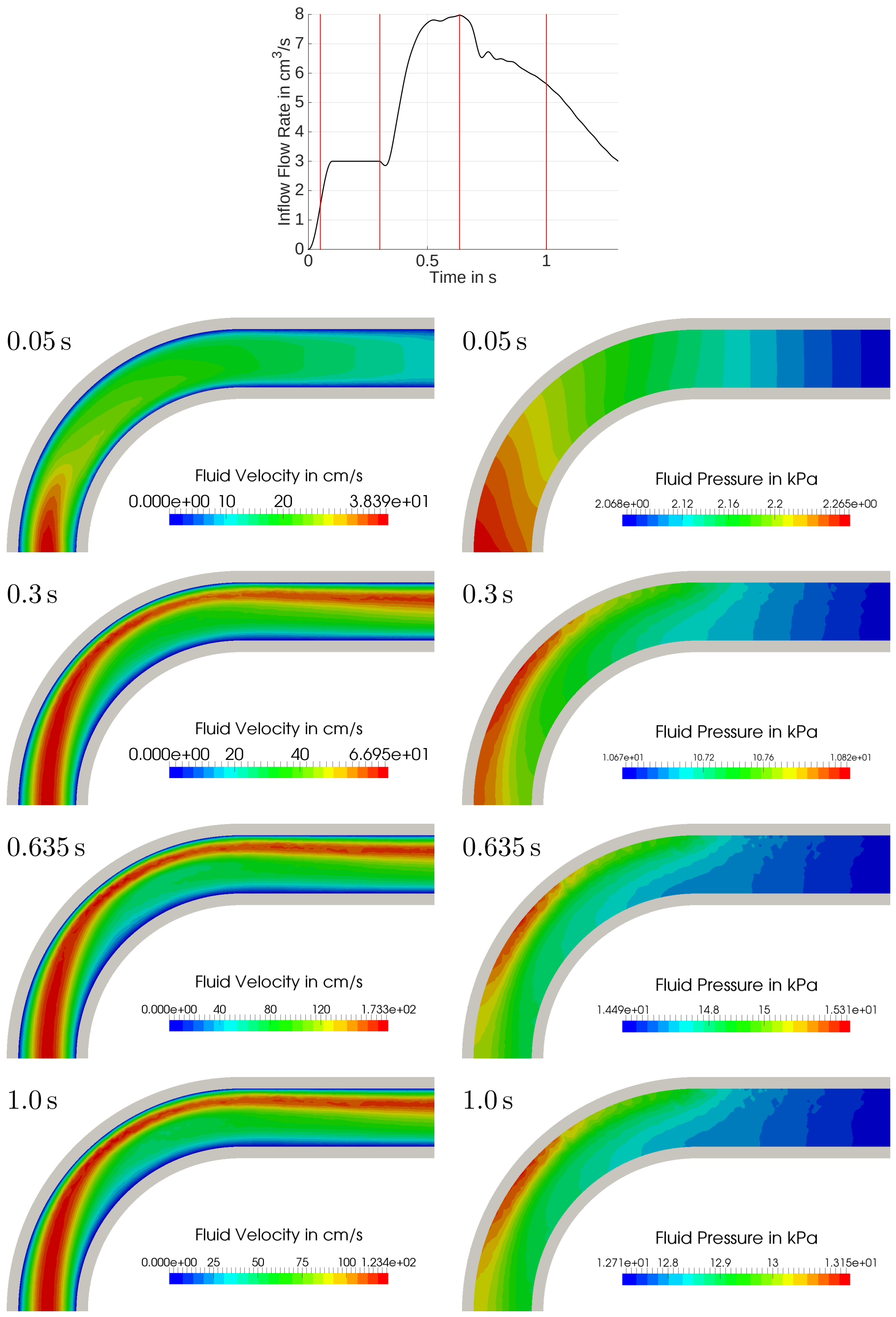


Figure 1.30: Evolution of velocity and pressure for Mesh #7 and P1 elements.

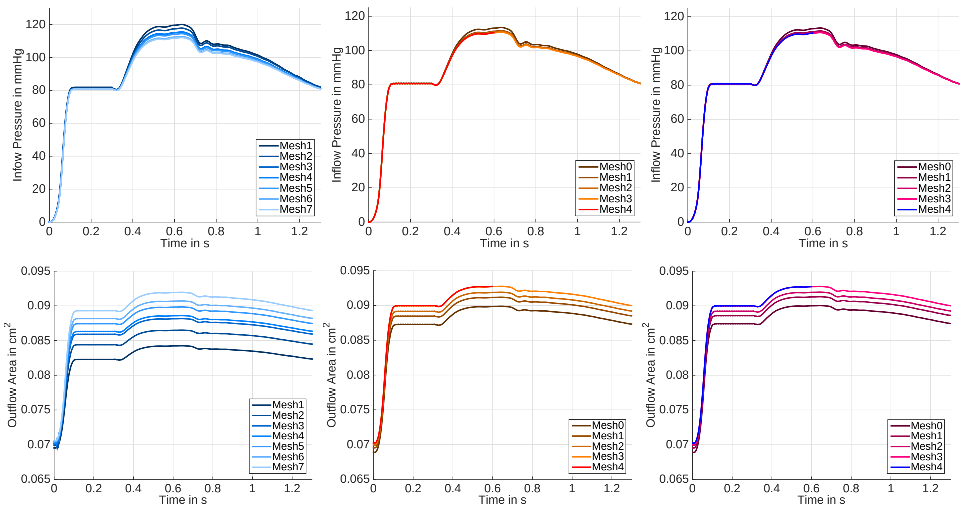


Figure 1.31: Mesh convergence of the inflow pressure (top) and outflow cross-sectional lumen area (bottom) during the heartbeat phase: P1 (left), P2 (middle) and $\bar{\mathrm{F}}$ (right) elements. Copyright © 2015 John Wiley & Sons, Ltd.

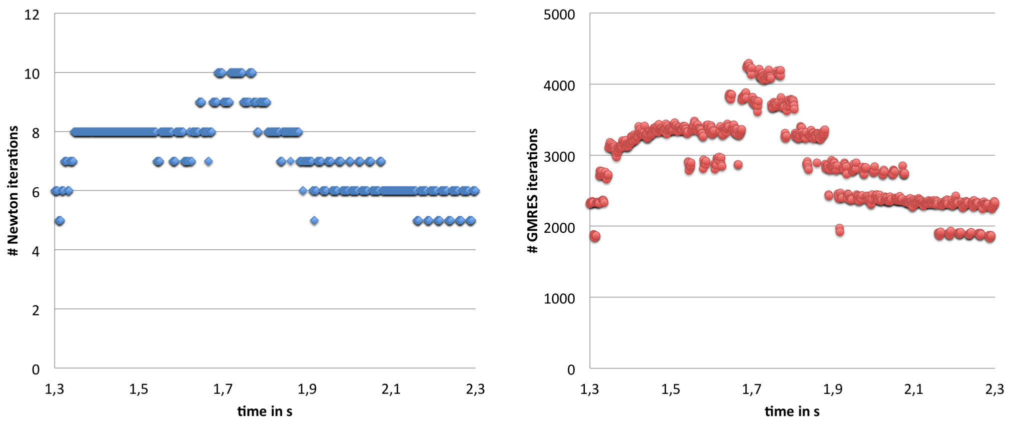


Figure 1.32: Number of Newton iterations for each time step (left) and sum of GMRES iterations in each time step (right) during the heartbeat using Mesh #3 and $\bar{\mathrm{F}}$ finite elements. We use 96 processors and thus 96 subdomains for the overlapping Schwarz method within our monolithic Dirichlet-Neumann preconditioner. Copyright © 2015 John Wiley & Sons, Ltd.

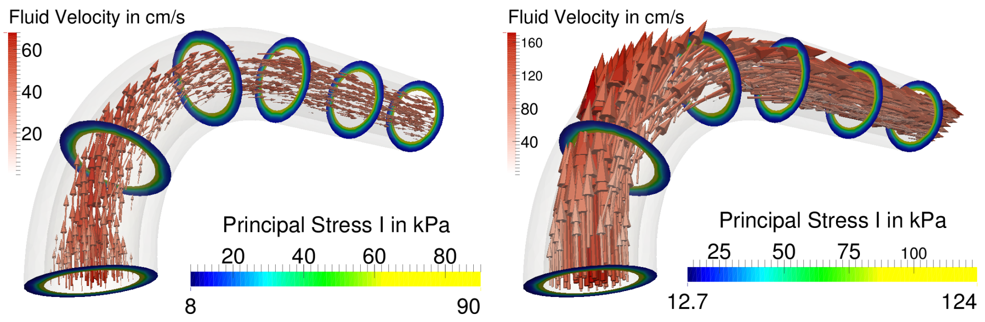


Figure 1.33: Fluid velocity and first principal Cauchy stress at $t = 0.3\,\mathrm{s}$ (left) and at $t = 0.635\,\mathrm{s}$ (right); for fluid velocities cf. Figure 1.29 and for the stress distribution cf. Figure 1.35. Taken from [24]; courtesy of Balzani, Fausten, and Schröder. Copyright © 2015 John Wiley & Sons, Ltd.

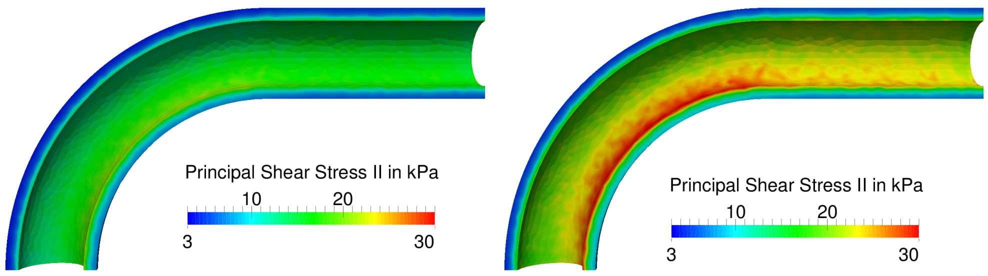

Principal Shear Stress II in kPa
10 20
3
30
Principal Shear Stress II in kPa
10 20
3
30

Figure 1.34: The principal Cauchy shear stress using the $\bar{\mathrm{F}}$ element at $t = 0.3\,\mathrm{s}$ (left) and $t = 0.635\,\mathrm{s}$ (right). Taken from [24]; courtesy of Balzani, Fausten, and Schröder. Copyright © 2015 John Wiley & Sons, Ltd.

1.5.2.3 Stresses

In this section, we present in Figures 1.33, 1.35 and 1.34 the transmural stress distributions which have been observed in [24] at simulation times $t = 0.3\,\mathrm{s}$ and $t = 0.635\,\mathrm{s}$ using different spatial discretizations. The stresses have been exported using the coupling class `feapMaterial`; see Section 1.3.3.4. In particular, Figure 1.33 shows the fluid velocity and transmural distribution of the first principal Cauchy stresses at different slices of the geometry, using P2 elements on Mesh #3.

In Figure 1.35, the first principal Cauchy stresses are depicted. In correspondence to [24], we observe that the stresses on the inner surface of the tube are strongly oscillatory with very high peak stresses for P1 elements; we refer to the discussion about the stiffness and locking of P1 elements in Section 1.2.6. On the contrary, $\bar{\mathrm{F}}$ and P2 discretizations show a smoother stress distribution and are thus preferable to P1 elements when accurate stress distributions are of interest.

Figure 1.34 presents the maximum shear stresses in the fluid-solid interface plane. As can be seen, these shear stresses are significantly higher at the inner curve. This observation corresponds well with the common hypotheses that the plaque evolves where low flow rates and high shear stresses are found in domains close to the endothelial cells, which is mostly at the inner curves of vessel walls.

We refer to [24] for a more detailed discussion on the transmural stress distribution in the arterial wall.

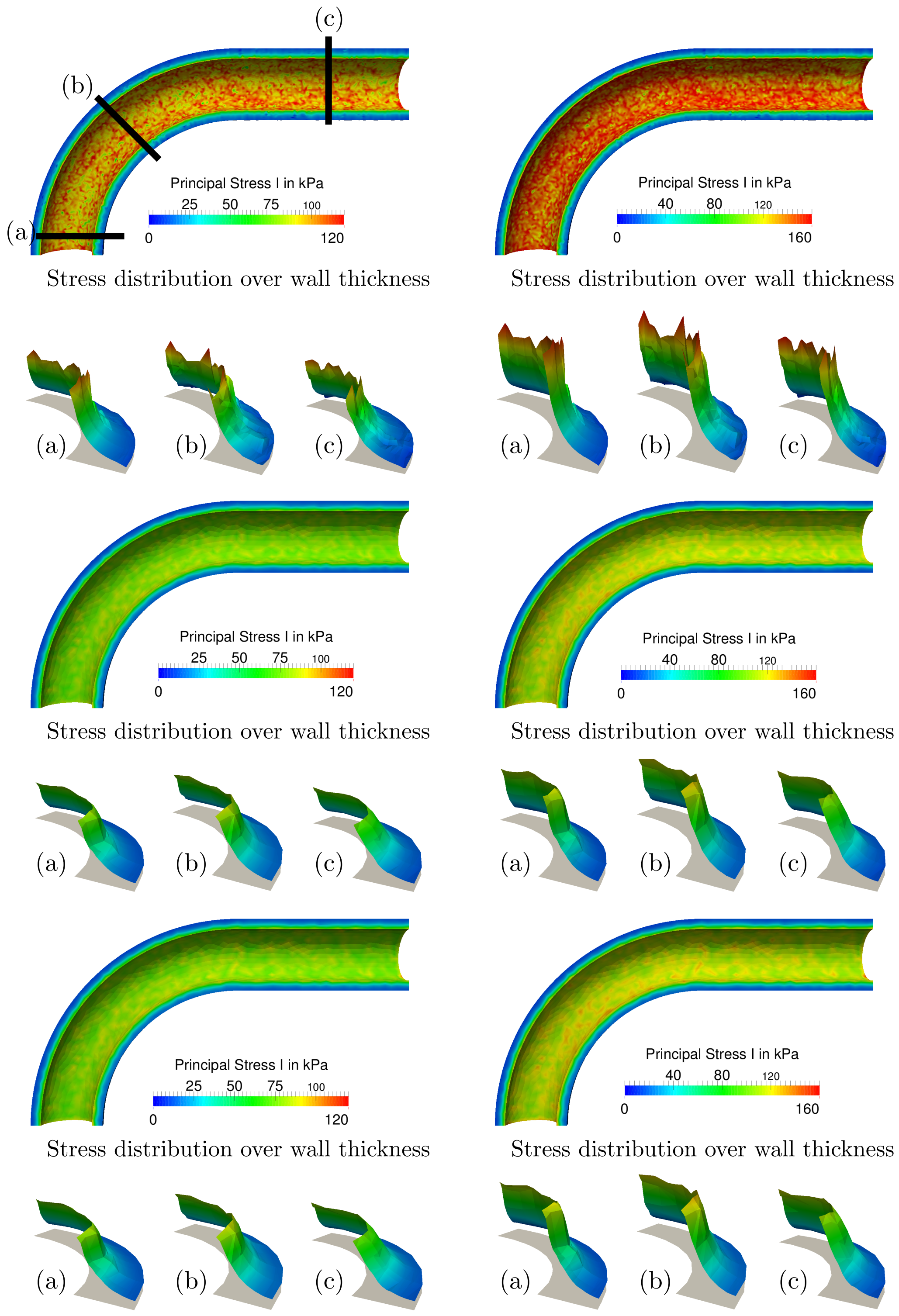

(c)
(b)
(a)
Principal Stress I in kPa
25 50 75 100
0 120
40 80 120
0 160
Stress distribution over wall thickness
(a) (b) (c)

Figure 1.35: Comparison of the first principal Cauchy stress for P1 (top), P2 (middle) and $\bar{\mathrm{F}}$ (bottom) elements at the inner surface and over the wall thickness at $t = 0.3\,\mathrm{s}$ (left) and $t = 0.635\,\mathrm{s}$ (right). Taken from [24]; courtesy of Balzani, Fausten, and Schröder. Copyright © 2015 John Wiley & Sons, Ltd.

1.6 Collection of Results

For completeness, this section collects systematically the numerical results presented in the text as well as additional graphs.

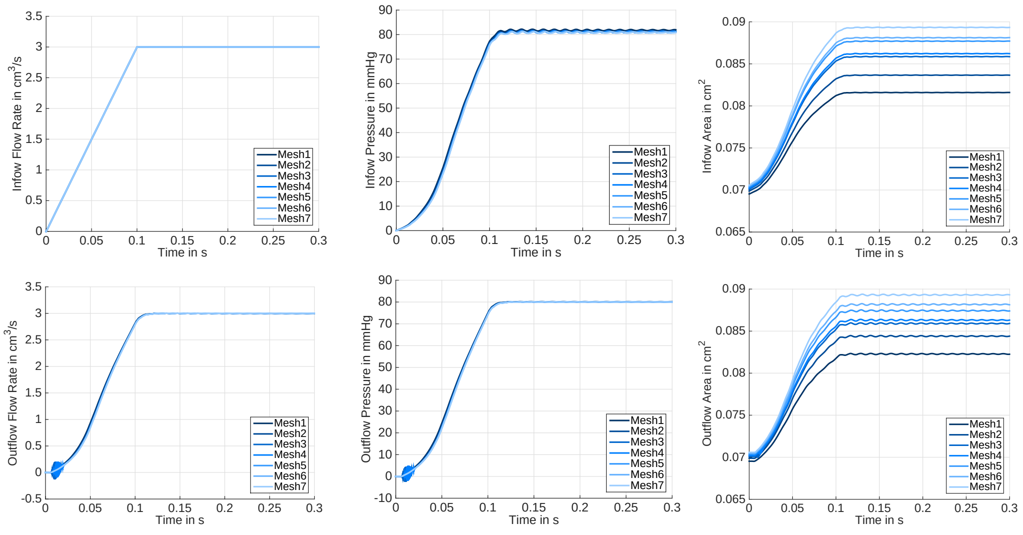


Figure 1.36: P1 mesh convergence study for the hyperelastic material using the linear ramp with $T_{\mathrm{R}} = 0.1\,\mathrm{s}$. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.1. Copyright © 2015 John Wiley & Sons, Ltd.

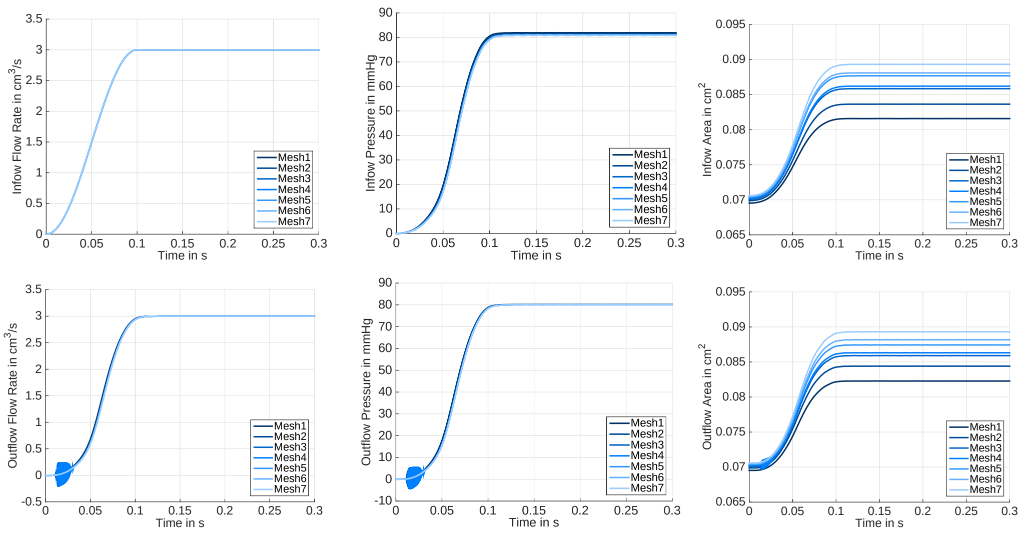


Figure 1.37: P1 mesh convergence study for the hyperelastic material using the cosine-type ramp with $T_R = 0.1\,s$. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.2. Copyright © 2015 John Wiley & Sons, Ltd.

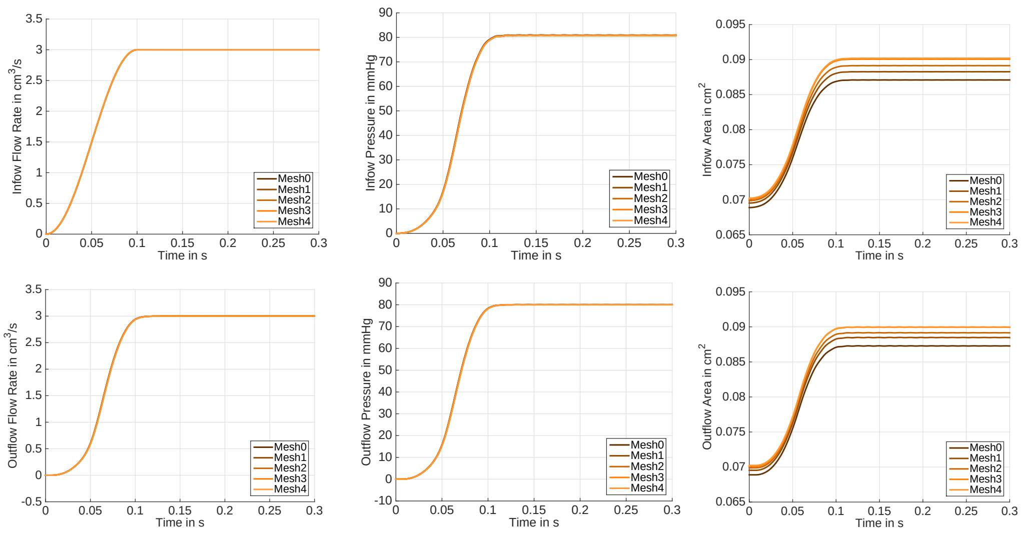


Figure 1.38: P2 mesh convergence study for the hyperelastic material using the cosine-type ramp with $T_R = 0.1\,s$. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.2. Copyright © 2015 John Wiley & Sons, Ltd.

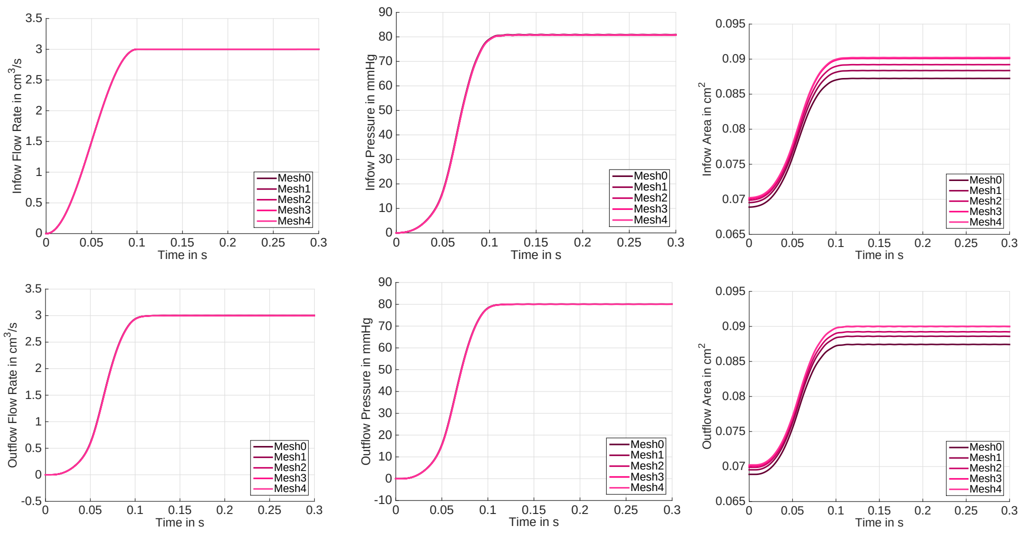


Figure 1.39: $\bar{\mathrm{F}}$ mesh convergence study for the hyperelastic material using the cosine-type ramp with $T_\mathrm{R} = 0.1\,\mathrm{s}$. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.2. Copyright © 2015 John Wiley & Sons, Ltd.

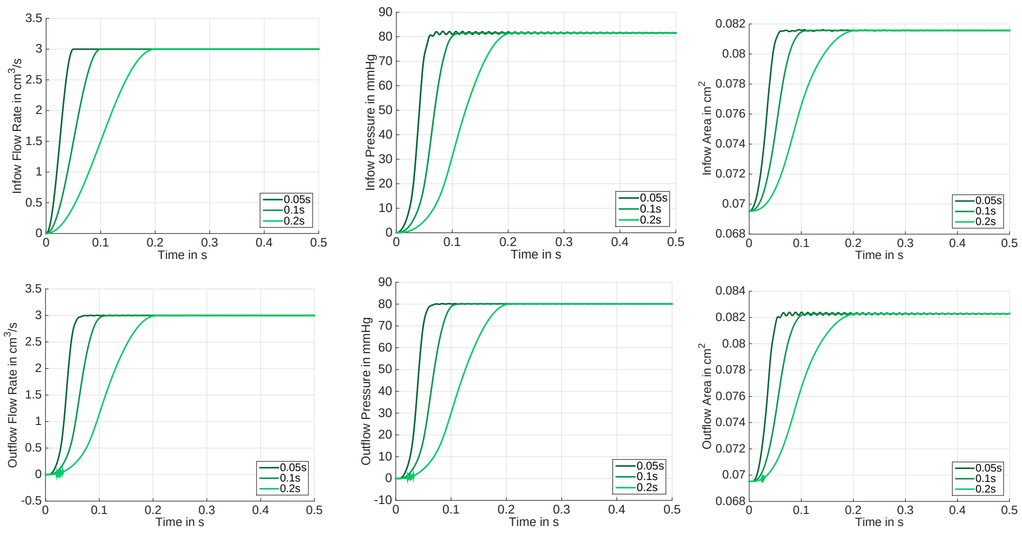


Figure 1.40: Hyperelastic material using P1 elements on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.2. Copyright © 2015 John Wiley & Sons, Ltd.

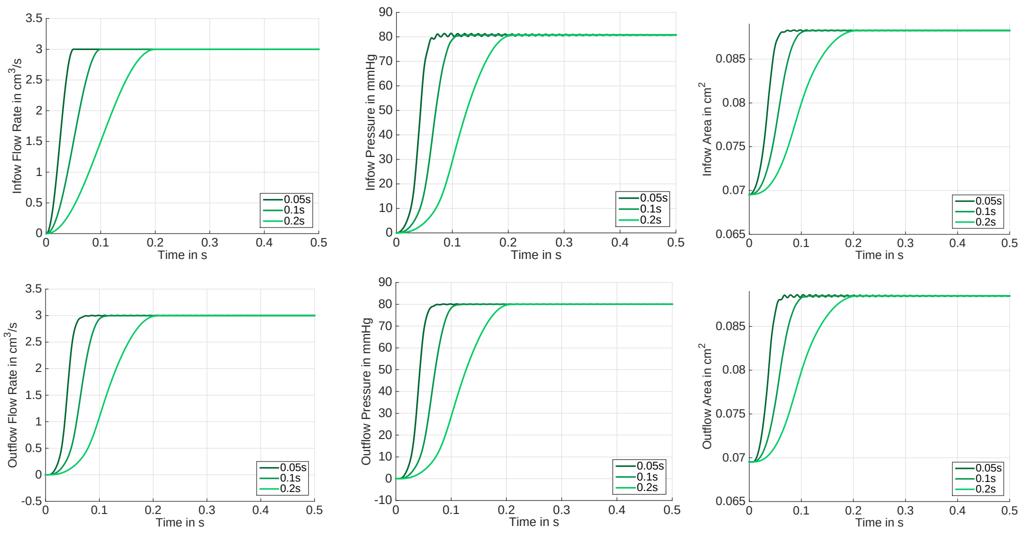


Figure 1.41: Hyperelastic material using P2 elements on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.3. Copyright © 2015 John Wiley & Sons, Ltd.

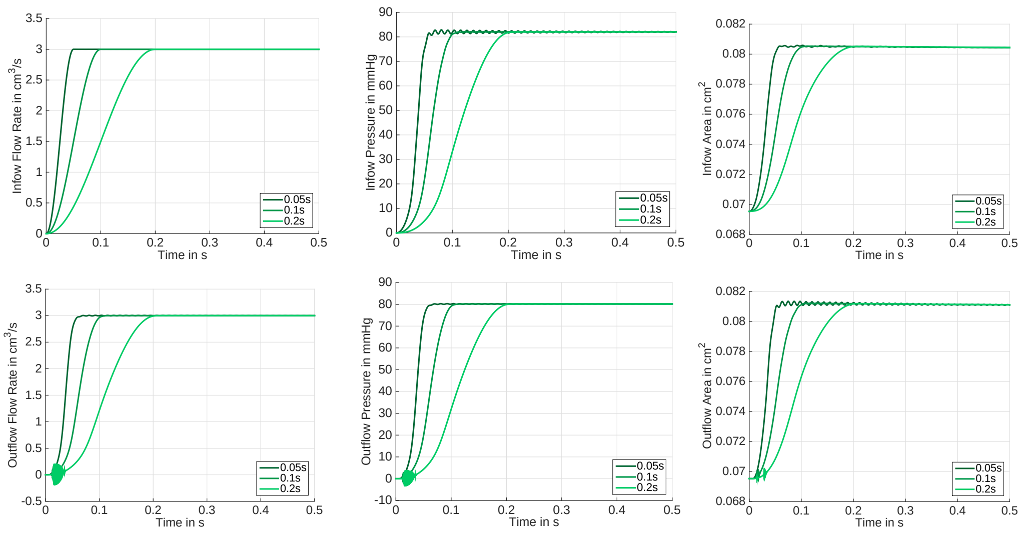


Figure 1.42: Viscoelastic material with P1 using the parameter Set 1 from Table 1.6 elements on Mesh #1. Flow rate (left), average pressure (middle), lumen cross section area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.3. Copyright © 2015 John Wiley & Sons, Ltd.

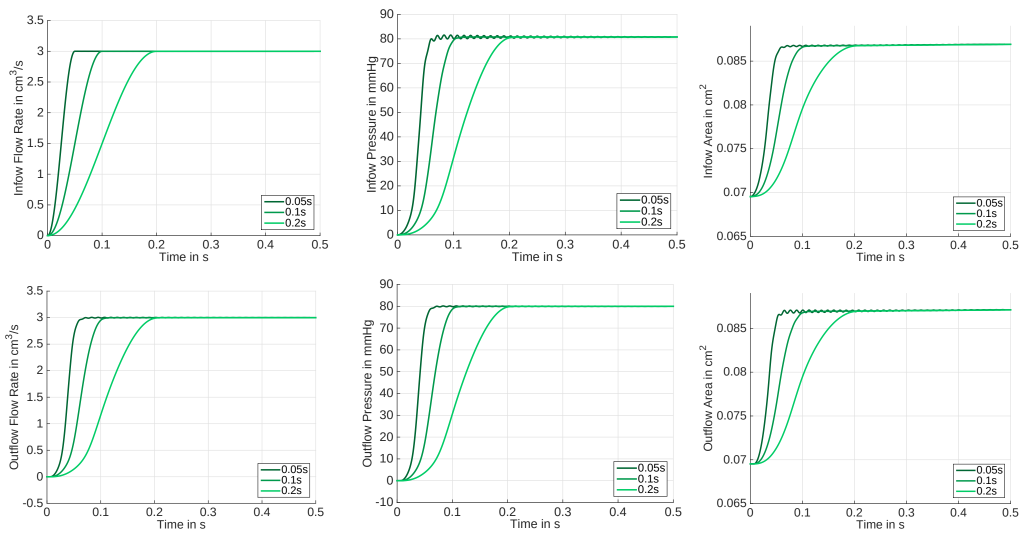


Figure 1.43: Viscoelastic material with P2 elements using the parameter Set 1 from Table 1.6 on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.3. Copyright © 2015 John Wiley & Sons, Ltd.

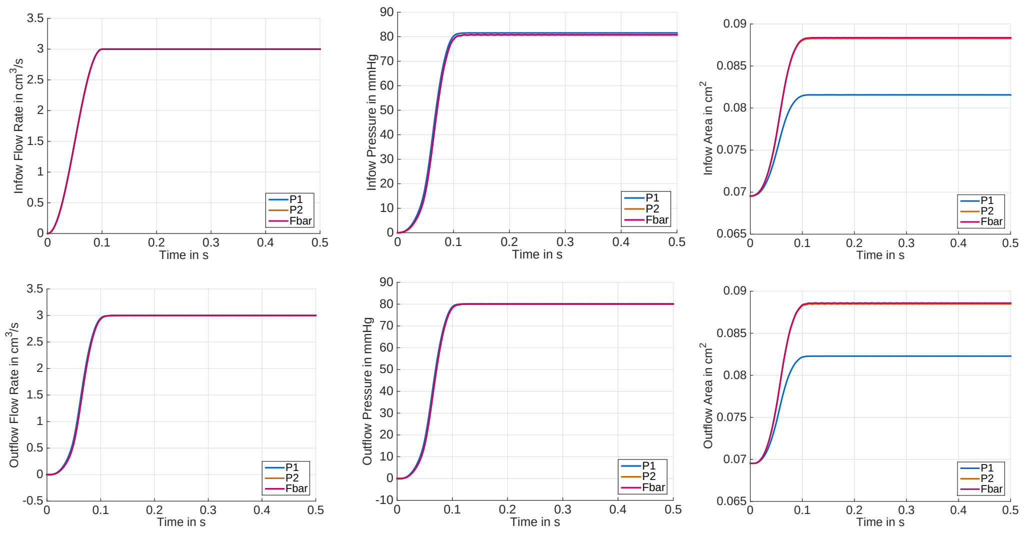


Figure 1.44: Hyperelastic material on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.4. Copyright © 2015 John Wiley & Sons, Ltd.

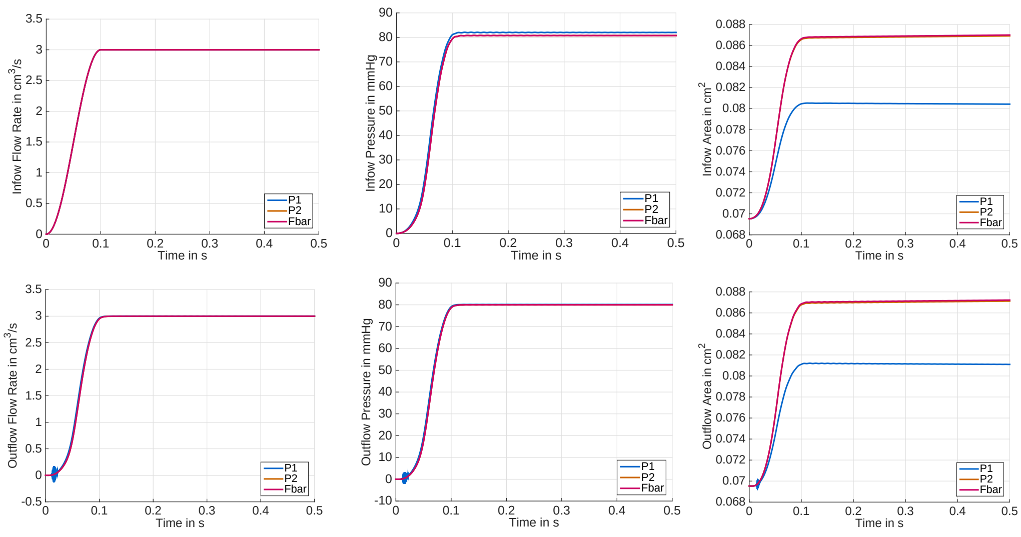


Figure 1.45: Viscoelastic material using the parameter Set 1 from Table 1.6 on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.4. Copyright © 2015 John Wiley & Sons, Ltd.

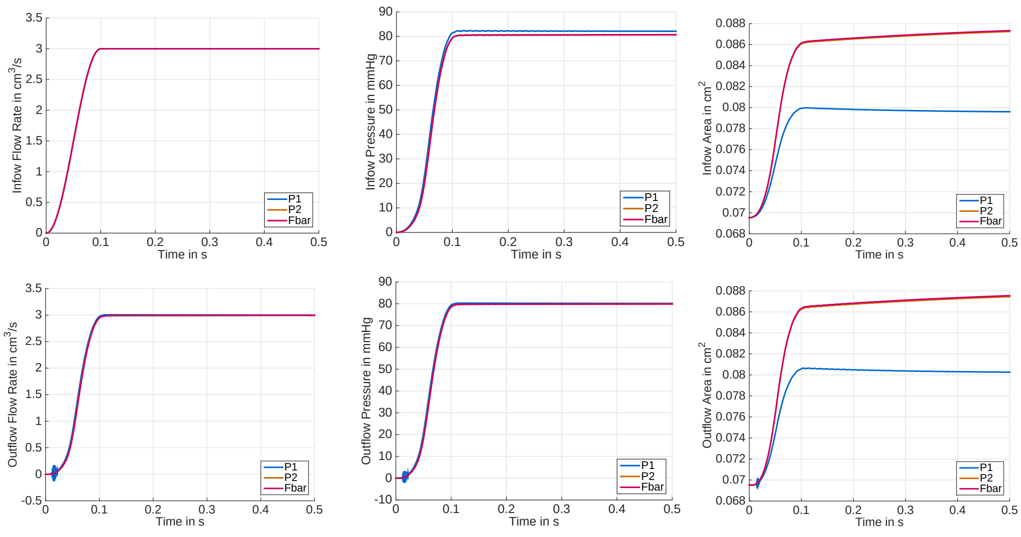


Figure 1.46: Viscoelastic material using the parameter Set 2 from Table 1.6 on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.4. Copyright © 2015 John Wiley & Sons, Ltd.

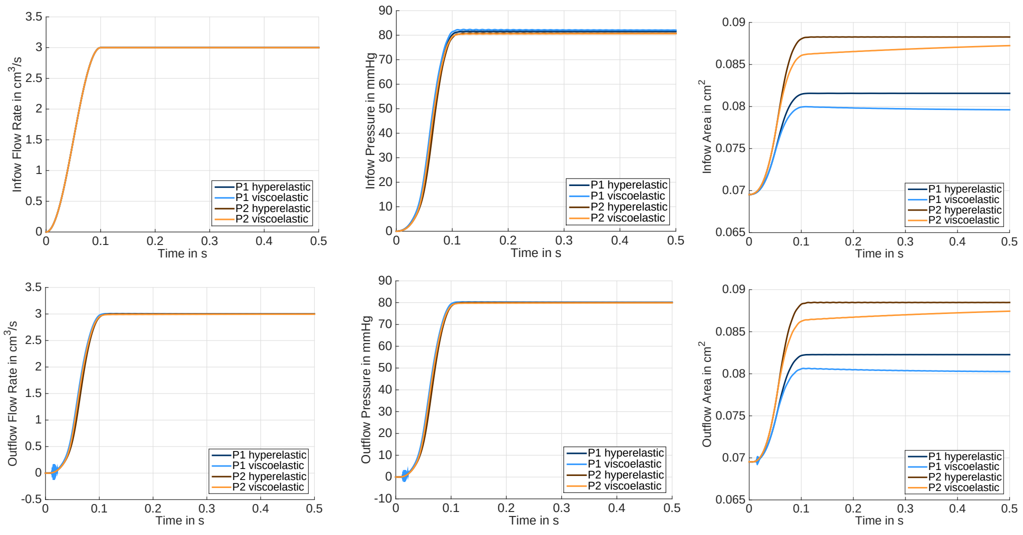


Figure 1.47: Comparison of the hyperelastic and the viscoelastic material using the parameter Set 2 from Table 1.6 on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.4. Copyright © 2015 John Wiley & Sons, Ltd.

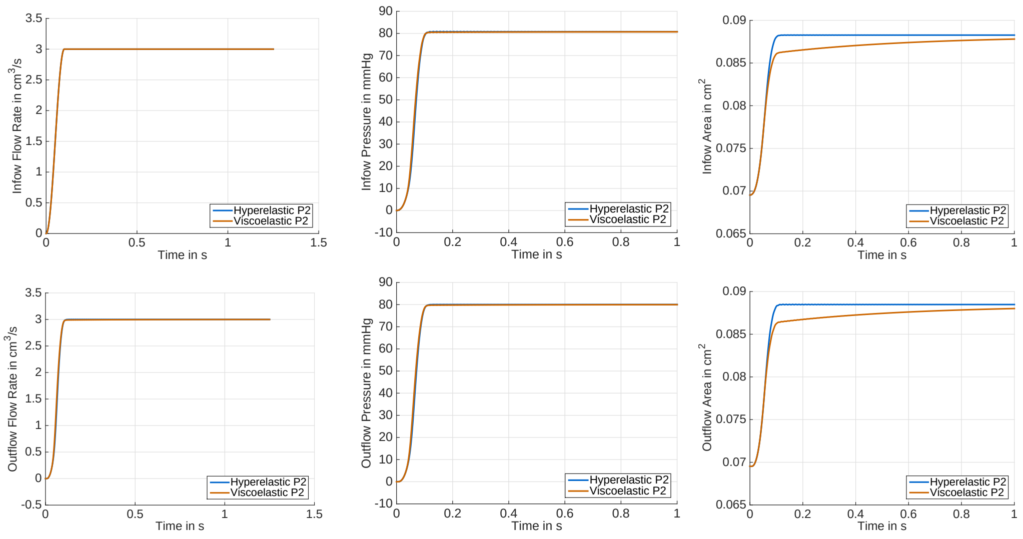


Figure 1.48: Comparison of the hyperelastic and the viscoelastic for P2 elements material using the cosine-type ramp and parameter Set 2 from Table 1.6 on Mesh #1. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.4. Copyright © 2015 John Wiley & Sons, Ltd.

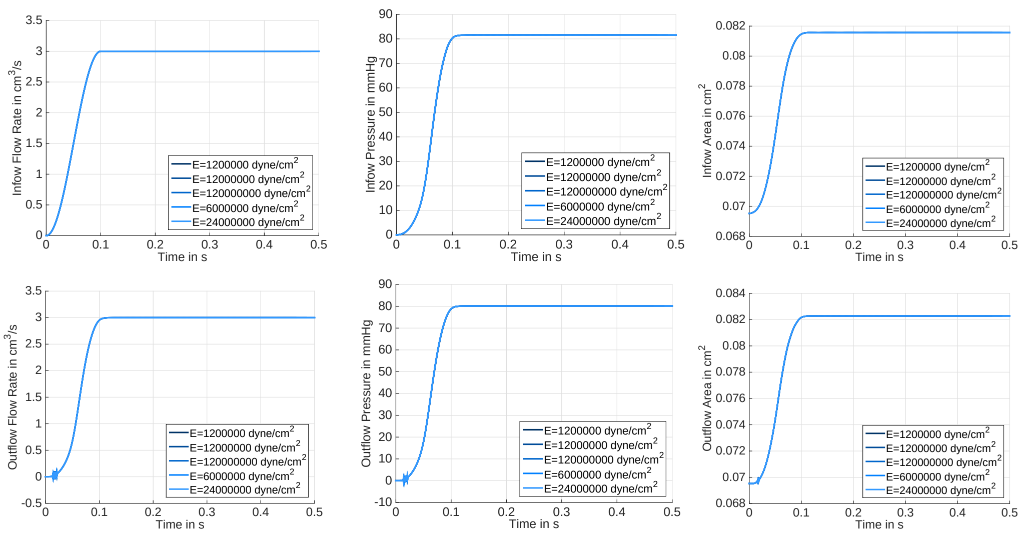


Figure 1.49: Sensitivity analysis of the absorbing boundary conditions. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.1.5. Copyright © 2015 John Wiley & Sons, Ltd.

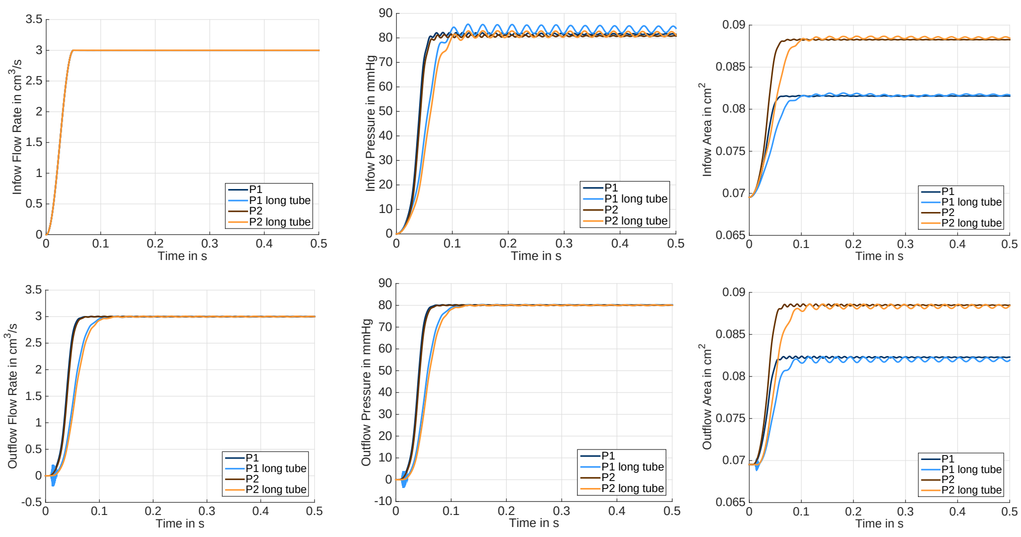


Figure 1.50: Comparison of Mesh #1 and the corresponding mesh of the long tube, cf. Tables 1.12 and 1.13. Outflow flow rate (left), inflow average pressure (middle), outflow cross-sectional lumen area (right); cf. Section 1.5.1.6. Copyright © 2015 John Wiley & Sons, Ltd.

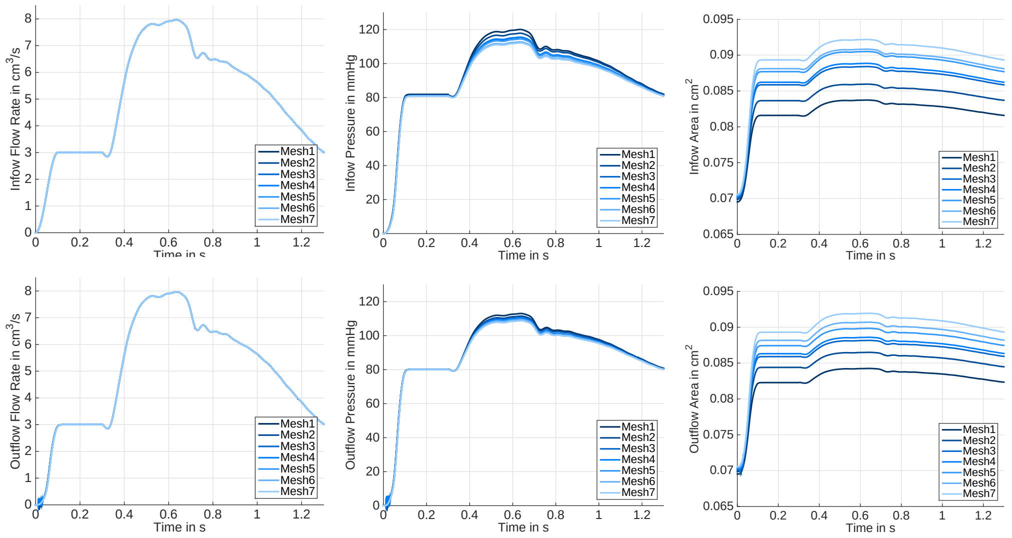


Figure 1.51: P1 mesh convergence study for the hyperelastic material for the heartbeat. Flow rate (left), average pressure (middle), lumen cross section area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.2.1. Copyright © 2015 John Wiley & Sons, Ltd.

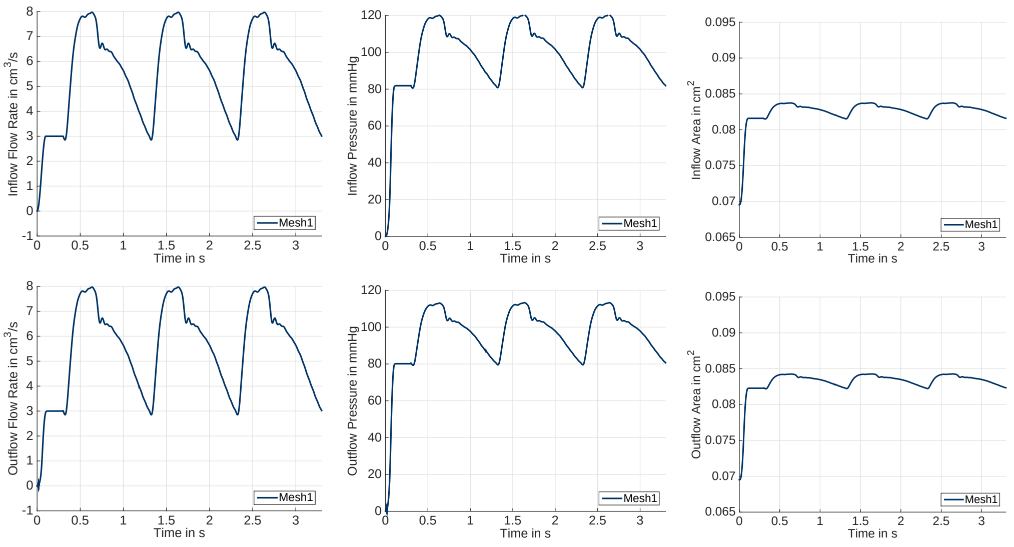


Figure 1.52: P1 simulation for the hyperelastic material for three heartbeats. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.2.1. Copyright © 2015 John Wiley & Sons, Ltd.

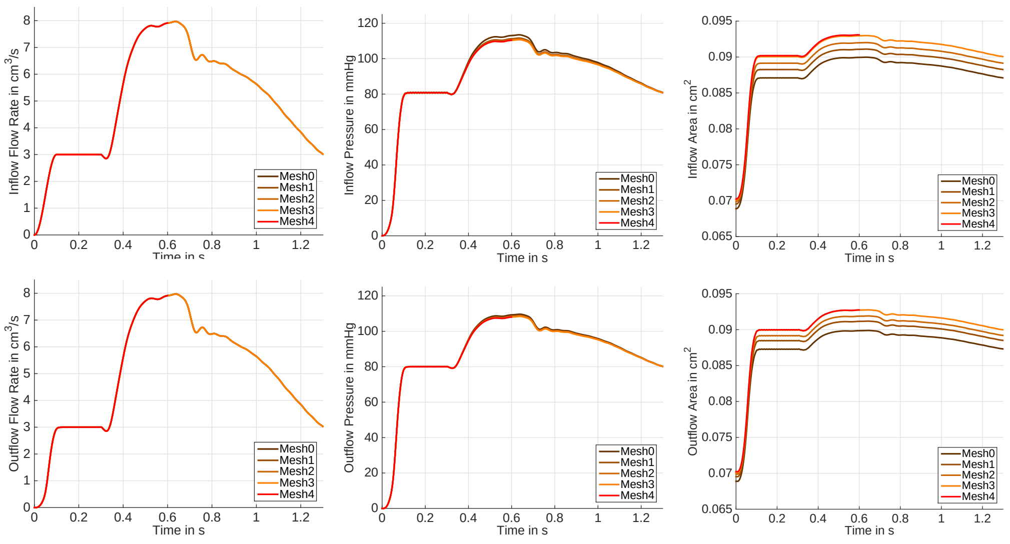


Figure 1.53: P2 mesh convergence study for the hyperelastic material for the heartbeat. Flow rate (left), average pressure (middle), lumen cross section area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.2.1. For clarity, Mesh #4 is plotted only until 0.6 s. Copyright © 2015 John Wiley & Sons, Ltd.

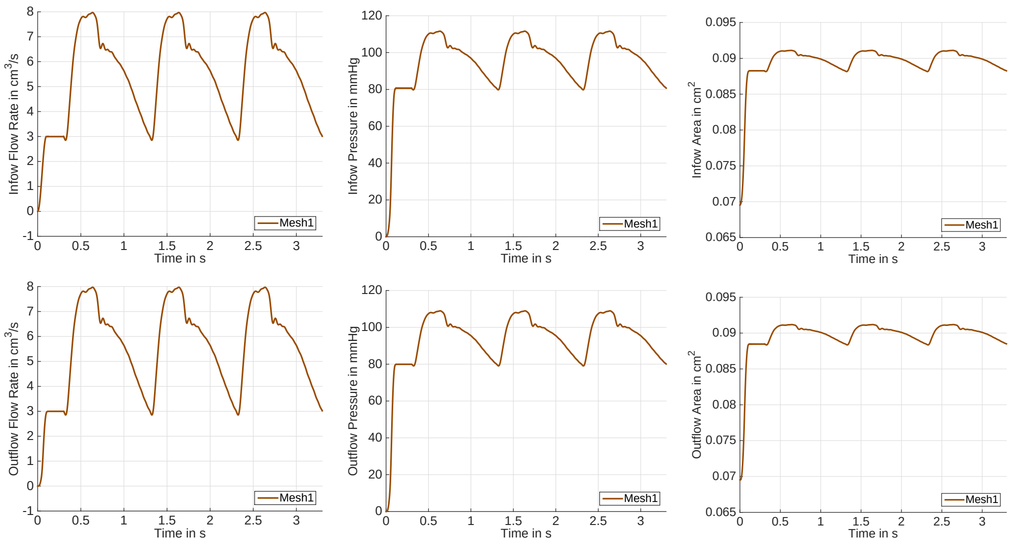


Figure 1.54: P2 simulation for the hyperelastic material for three heartbeats. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.2.1. Copyright © 2015 John Wiley & Sons, Ltd.

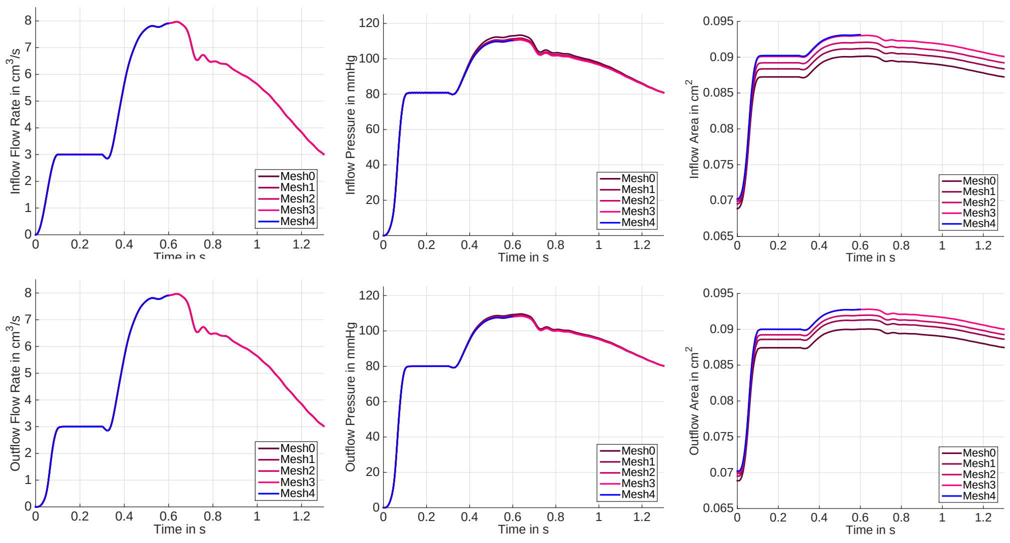


Figure 1.55: $\bar{\mathrm{F}}$ mesh convergence study for the hyperelastic material for the heartbeat. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.2.1. For clarity, Mesh #4 is plotted only until 0.6 s. Copyright © 2015 John Wiley & Sons, Ltd.

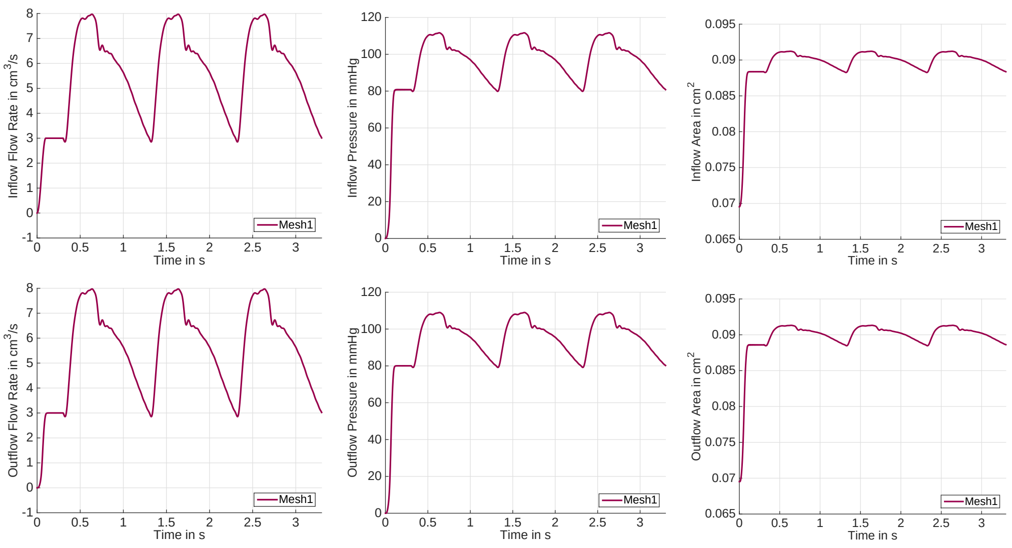


Figure 1.56: $\bar{\mathrm{F}}$ simulation for the hyperelastic material for three heartbeats. Flow rate (left), average pressure (middle), and cross-sectional lumen area (right), over time at the inlet (top) and outlet (bottom); cf. Section 1.5.2.1. Copyright © 2015 John Wiley & Sons, Ltd.

1.7 Conclusion

We have proposed precise simulation settings for a synthetic simulation of blood flow through a geometry representing an idealized coronary artery, in order to set the ground for the computation of realistic transmural stresses in the future. We use a simple geometry to provide a benchmark which is easily reproducible but which still captures all numerical difficulties arising in realistic simulations. We discuss our algorithmic approaches and numerical results of our feasibility study and provide measures which may be useful for comparisons with future simulations, experiments or for code validation.

An important contribution of this chapter, which is based on [24], is the use of a highly nonlinear, polyconvex anisotropic structural model in FSI together with a suitable discretization. The flow conditions were set such that the flow rates and the pressure inside the artery were physiological. To account for the wall pre-stresses, we have initialized the simulation starting from the rest condition, which was then raised to a steady flow. Then, typical flow rates for coronaries were imposed at the inflow, while using an absorbing boundary condition at the outflow. Different material properties were studied, including anisotropic and viscoelastic ones at finite strains. We have found that at least P2 or $\bar{\mathrm{F}}$ finite elements are necessary for reasonable stress approximations in the vessel wall, since the accuracy of P1 discretizations is comparatively poor. This is of practical relevance since FSI simulations in biomechanics using simple, linear finite elements for the structure are not uncommon. Further steps are necessary to reach our goal of computing realistic transmural stresses. This includes the use of patient specific geometries with several layers and may include other improvements such as fiber remodeling and the modeling of plaque including the fibrous cap.

Numerically, the use of adaptive time stepping schemes may help to further reduce the computational effort. The use of a better suited two-level Schwarz preconditioner (cf. Chapter 2 for the presentation of a parallel implementation of the GDSW preconditioner), which is significantly more robust for the sophisticated material models under consideration for the structural block, is shown in Chapter 3. It is also shown that the number of GMRES iterations and therefore the total simulation time can be reduced significantly by only replacing the preconditioner for the structure. This indicates that the structure is the most severe part with respect to preconditioning in our setting. Therefore, the investigation of better suited preconditioners for the structure is a reasonable next step to improve our numerical FSI framework.

2 A Parallel Implementation of the Two-Level Overlapping Schwarz GDSW Preconditioner

In this chapter, which is based on [105, 107], we present a software framework for two-level overlapping Schwarz preconditioners in `Trilinos` [109] focusing on the GDSW (Generalized Dryja-Smith-Widlund) preconditioner as an example. The GDSW preconditioner is a two-level overlapping Schwarz preconditioner introduced in [70] with a proven condition number bound for the general case of John domains for scalar elliptic and linear elastic model problems. The coarse space is related to that of FETI-DP (Finite Element Tearing and Interconnecting - Dual-Primal) and BDDC (Balancing Domain Decomposition by Constraints) methods [197, 69], and, as for FETI-DP and BDDC methods, variants are robust for almost incompressible elasticity [72]. The method is algebraic in the sense that it can be constructed from the assembled system matrix, and additionally, no coarse triangulation is needed. This is of special interest for the use as a preconditioner for a block, e.g., in monolithic multiphysics coupling. The GDSW preconditioner is thus well-suited to be used in the context of Fluid-Structure Interaction (FSI) simulations, i.e., to replace `Trilinos IFPACK` as the default preconditioner used for the blocks in the monolithic matrix in [23, 24] and in Chapter 1. Therefore, we refer to [105, 106, 107] (see also Chapter 3), where the GDSW preconditioner is used as preconditioner for the structural block in FSI simulations. However, compared to FETI-DP or BDDC methods, the standard GDSW coarse space is larger, especially in three dimensions. It should be expected that, to sustain parallel scalability, the transition to an inexact method has to be performed earlier. Nonetheless, in this thesis, we are able to show weak parallel scalability for elasticity for up to 8 000 cores in three dimensions for the two-level method using exact solvers. Moreover, reduced coarse spaces are available (cf. [72]) which are smaller but introduce stronger coupling. First results for GDSW were presented at the 17th International Conference on Domain Decomposition Methods in Science and Engineering in Strobl, Austria, in the summer of 2006, which were then published in early 2008 in [71]. That work could be regarded as a generalization of earlier work by Dryja, Smith, and Widlund [74], cf. [71], and thus named GDSW (Generalized Dryja-Smith-Widlund) preconditioner.

Iterative solvers for problems on unstructured grids, which are scalable for elasticity problems to the full range of today's supercomputers, are parallel multilevel methods from the family of Domain Decomposition Methods (DDM) or Multigrid (MG) algorithms. Algebraic Multigrid (AMG) methods have recently scaled to 262 144 cores and 524 288 MPI processes for elasticity using special interpolations [14]. FETI-DP domain decomposition methods have scaled to 524 288 cores [127] and to 786 432 MPI ranks and cores for 63 billion displacement unknowns [126] in nonlinear hyperelasticity, making this the currently largest range of parallel scalability reported for any linear or nonlinear domain decomposition method. BDDC methods have scaled to 458 432 cores [11] for linear elasticity. These latter domain decomposition methods are, however, not completely algebraic, i.e., they need access to the matrices of the local Neumann problems.

This is not the case for domain decomposition methods of the overlapping Schwarz type, and it has been argued that they are therefore easier to construct. Overlapping Schwarz methods have scaled to 8 192 subdomains and MPI processes (on 8 192 sockets) in [122, 121]. They have also scaled for large multi-physics problems such as FSI; see [207], where an overlapping Schwarz method applied to a monolithic system has scaled to 3 072 cores. In [156], a two-level Newton-Krylov-Schwarz method has been applied to the bidomain equation using up to 2 048 cores. A hybrid multilevel version of this Schwarz preconditioner has scaled to 2 048 cores in [175, 176] by Scacchi et. al using up to 5 levels. Recently, the multilevel Schwarz preconditioner has also been applied to cardiac electro-mechanical coupling in [52]. Therein, strong scalability using 4 levels for up to 512 cores has been reported. In [142], a three-level overlapping Schwarz method has been shown to be strongly scalable for up to 10 240 cores of an IBM cluster for a three-dimensional linear elasticity problem. Two special techniques are used to obtain scalability for unstructured meshes: the partitioning is performed in two stages, and special care is taken to preserve the features of the boundary also on the coarse levels [142]. The first technique may also be of help for us. All geometric features of the boundary, however, are resolved by the GDSW coarse basis functions automatically.

In this chapter, we present a software framework for parallel scalable two-level overlapping Schwarz methods in `Trilinos` and discuss strengths and weaknesses of our approach. Further, we discuss possible future improvements to obtain a framework for overlapping Schwarz methods with improved robustness and scalability. In GDSW, no coarse triangulation is needed but instead equivalence classes denoted as vertices, edges, and faces have to be defined and identified algebraically to construct an energy minimizing coarse space; see Section 2.3.4. Our techniques to identify vertices, edges, and faces in parallel (see Section 2.3.3) are also of interest for other related preconditioners such as FETI-DP and BDDC methods [197, 149]. They may also be helpful in the parallel implementation of related multiscale discretization methods; see, e.g., [111, 103] and Chapter 4 for the ACMS method. On the other hand, multiscale discretization methods may enrich the GDSW coarse space to improve the robustness of the preconditioner with respect to highly heterogeneous (multiscale) problems. We address this issue in Chapter 5, where the basis functions of a special finite element method are employed to build a coarse space for a two-level overlapping Schwarz preconditioner. Therein, different coarse spaces, including the GDSW coarse space, are tested for different highly heterogeneous problems.

The remainder of this chapter is organized as follows: first, we give a description of the GDSW two-level overlapping Schwarz preconditioner with a focus on the application to elasticity problems in Section 2.1. Next, we describe a hybrid version of the GDSW preconditioner, which is motivated by the balancing preconditioner for the BDDC method [137]. The implementation of our two-level overlapping Schwarz framework for `Trilinos` is described in Section 2.3. In the two following Sections 2.4 and 2.5, the benchmark problems for our numerical tests and the corresponding results, respectively, are described. Finally, a conclusion is given in Section 2.6.

2.1 The GDSW Preconditioner

Consider the system of linear equations

$$Ax = b \tag{2.1}$$

arising from a finite element discretization of a partial differential equation, such as, a Laplacian or an elasticity equation on a domain Ω, with sufficient Dirichlet boundary conditions. Let $\Omega \subset \mathbb{R}^2$ or $\Omega \subset \mathbb{R}^3$ be decomposed into nonoverlapping and corresponding overlapping subdomains, cf. Figure 2.1 for a decomposition of a cube in 3D. The overlapping decomposition defines the first level, whereas the nonoverlapping decomposition is used to define the coarse level of the preconditioner. The GDSW preconditioner [70, 71] is a two-level

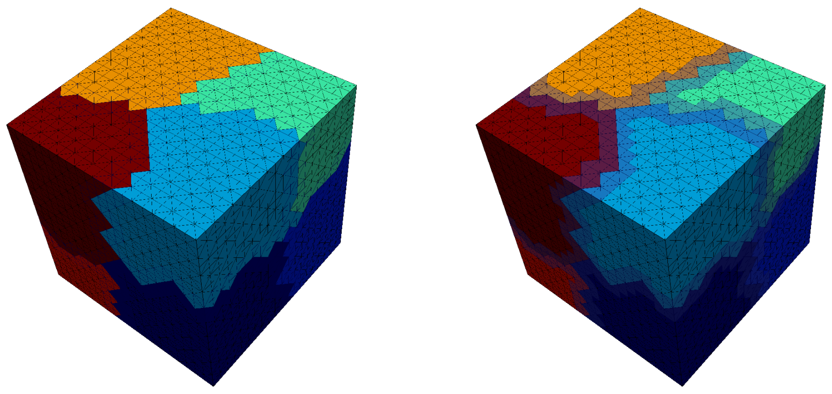


Figure 2.1: Nonoverlapping (left) and corresponding overlapping decomposition (right) of a cube. The overlap has a width of $\delta = 1h$; cf. [107].

additive overlapping Schwarz preconditioner with exact solvers; cf. [197]. Thus, the preconditioner can be written in the form

$$M_{\text{GDSW}}^{-1} = \Phi A_0^{-1} \Phi^T + \underbrace{\sum_{i=1}^{N} R_i^T \tilde{A}_i^{-1} R_i}_{=M_{OS1}^{-1}}, \tag{2.2}$$

where

$$A_0 = \Phi^T A \Phi \tag{2.3}$$

corresponds to the coarse problem and the $\tilde{A}_i = R_i^T A R_i$, $i = 1, ...N$, correspond to the concurrent local overlapping problems on the fine level (M_{OS1}^{-1}). Here, the matrices R_i represent the restriction operators to the overlapping subdomains. This definition is equivalent to the definition of the standard two-level Schwarz preconditioner,

$$M_{OS2}^{-1} = R_0^T \left(R_0 A R_0^T\right)^{-1} R_0 + \sum_{i=1}^{N} R_i^T \tilde{A}_i^{-1} R_i,$$

if $\Phi = R_0^T$. Indeed, for the GDSW preconditioner, the choice of Φ is the main ingredient. Instead of a coarse Lagrangian finite element basis, which requires a coarse triangulation, a partition of unity is defined on the interface of the decomposition, and an energy-minimizing extension to the interior is then used to define the coarse basis functions.

For the construction of these coarse basis functions, we consider the nonoverlapping domain decomposition. In particular, for linear elasticity, the interface values of the coarse basis functions are the restrictions of the rigid body modes of each subdomain to the interface of the nonoverlapping decomposition.

In two dimensions, the space of rigid body motions is spanned by two translations and one rotation (precisely, the linear approximation of the rotation), and, in three dimensions, by three translations and three linearized rotations. From the Korn inequalities, we see that we can control the null space of the operator if we set essential boundary conditions or if we require the solution to be orthogonal to all rigid body modes. For the formulae of the linearized rigid body motions, we refer to Section 1.2.2.

Let Γ be the set of degrees of freedom on the interface of the decomposition, i.e., the degrees of freedom which belong to more than one subdomain, and I be the set of the remaining degrees of freedom. All degrees of freedom corresponding to nodes on the Dirichlet boundary are considered as interior degrees of freedom. The basis functions of the GDSW coarse space are given by

$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -A_{II}^{-1} A_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}, \tag{2.4}$$

where Φ_Γ is defined from restrictions of the rigid body motions to (in 3D) faces, edges, and vertices of the interface of the nonoverlapping decomposition. Note that $A_{II} = \text{diag}_{i=1}^N (A_{II}^{(i)})$ is a block diagonal matrix containing the local matrices $A_{II}^{(i)}$ from the nonoverlapping subdomains. Its inverse can thus be computed block-wise and in parallel.

To define Φ_Γ, the set Γ is divided into M connected components Γ_j, i.e., edges and vertices in 2D and faces, edges, and vertices in 3D, which are common to the same set of subdomains. For each Γ_j, we construct a matrix Φ_{Γ_j} such that the columns are restrictions of the rigid body modes of the neighboring subdomains to the interface component. Since only two of the three linearized rotations are linearly independent for straight edges in 3D, one linearly dependent rotation is removed. For vertices, however, all rotations are omitted from Φ_{Γ_j} because the translations are sufficient. Let $R_{\Gamma j}$ be the restriction from Γ onto Γ_j. Then the values of the basis functions on Γ can be written as

$$\Phi_\Gamma = \begin{bmatrix} R_{\Gamma_1}^T \Phi_{\Gamma_1} & \dots & R_{\Gamma_M}^T \Phi_{\Gamma_M} \end{bmatrix}$$

and the complete matrix Φ is given by (2.4).

We can now compute the coarse matrix A_0 of the GDSW preconditioner in (2.2) either by means of (2.3) or by

$$\begin{aligned} A_0 &= \Phi^T A \Phi = \begin{bmatrix} -A_{II}^{-1} A_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}^T \begin{bmatrix} A_{II} & A_{\Gamma I}^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} -A_{II}^{-1} A_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix} \\ &= \Phi_\Gamma^T \begin{bmatrix} -A_{II}^{-1} A_{\Gamma I}^T \\ I_\Gamma \end{bmatrix}^T \begin{bmatrix} 0 \\ S_{\Gamma\Gamma} \Phi_\Gamma \end{bmatrix} = \Phi_\Gamma^T S_{\Gamma\Gamma} \Phi_\Gamma, \end{aligned} \tag{2.5}$$

with $S_{\Gamma\Gamma} = A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{\Gamma I}^T$ being the Schur complement arising by eliminating the interior degrees of freedom. Note that, as in FETI-DP and BDDC methods, such Schur complements are typically not built explicitly but the application to vectors are computed from right to left.

The condition number estimate for the GDSW preconditioner,

$$\kappa \left(M_{\mathrm{GDSW}}^{-1} A \right) \leq C \left(1 + \frac{H}{\delta} \right) \left(1 + \log \left(\frac{H}{h} \right) \right)^2, \tag{2.6}$$

holds also for the general case of Ω decomposed into John domains (in two dimensions), and thus, in particular, for unstructured domain decompositions; cf. [70, 71].

2.2 A Hybrid GDSW Preconditioner

The GDSW preconditioner is related to other preconditioners using energy-minimizing coarse spaces, e.g., to the balancing preconditioner applied to BDDC [137], which is defined by

$$(I-P)\,M_{BDDC}^{-1}\,(I-P)^T + U_\Gamma \left(U_\Gamma^T S_{\Gamma\Gamma} U_\Gamma\right)^{-1} U_\Gamma^T, \tag{2.7}$$

with $P = U_\Gamma \left(U_\Gamma^T S_{\Gamma\Gamma} U_\Gamma\right)^{-1} U_\Gamma^T S_{\Gamma\Gamma}$ and the BDDC preconditioner M_{BDDC}. However, the standard coarse space in BDDC is slightly different from the one in GDSW (the extension is performed to the non-primal variables in BDDC and to the interior in GDSW).

The preconditioner

$$\begin{aligned} M_{\text{GDSW}-\text{hybrid}}^{-1} &= (I-P_0)\,M_{OS1}^{-1}\,(I-P_0)^T + \Phi\left(\Phi^T A \Phi\right)^{-1}\Phi^T \\ &= (I-P_0)\,M_{OS1}^{-1}\,(I-P_0)^T + \Phi A_0^{-1}\Phi^T \end{aligned} \tag{2.8}$$

with $P_0 = \Phi\left(\Phi^T A \Phi\right)^{-1}\Phi^T A$, can be motivated from deflation or balancing. Since the inverse $A_0^{-1} = \left(\Phi^T A \Phi\right)^{-1}$ has to be computed in the coarse correction of

$$M_{\text{GDSW}}^{-1} = M_{OS1}^{-1} + \Phi A_0^{-1}\Phi^T, \tag{2.9}$$

it can be reused for the projections $(I-P_0)$ and $(I-P_0)^T$. However, in a naive implementation, the forward and backward substitution for the coarse solve is performed three times in one application of the hybrid preconditioner (a closer look reveals that this is not necessary), whereas it is performed only once in the standard (additive) GDSW preconditioner. The multiplicative version in (2.8), however, does not allow a completely concurrent solution of the levels.

For a symmetric A, we obtain for the Schwarz operator

$$\begin{aligned} P_{\text{GDSW}-\text{hybrid}} &= (I-P_0)\left(\sum_{i=1}^{N} R_i^T \tilde{A}_i^{-1} R_i\right)(I-P_0)^T A + \Phi A_0^{-1}\Phi^T A \\ &= (I-P_0)\left(\sum_{i=1}^{N} P_i\right)(I-P_0) + P_0. \end{aligned}$$

Thus, this preconditioner is equivalent to the hybrid preconditioner 1 from [197, Section 2.2] with the GDSW coarse space.

$$\left[\begin{array}{c|c} x_1 & \\ x_2 & \\ x_3 & \\ x_4 & \\ x_5 & \\ x_6 & \end{array}\right] \quad \left[\begin{array}{c|c} x_1 & x_1 \\ x_2 & x_2 \\ x_3 & x_3 \\ x_4 & x_4 \\ x_5 & x_5 \\ x_6 & x_6 \end{array}\right] \quad \left[\begin{array}{c|c} x_1 & \\ x_2 & \\ x_3 & \\ & x_4 \\ & x_5 \\ & x_6 \end{array}\right] \quad \left[\begin{array}{c|c} x_1 & \\ & x_2 \\ x_3 & x_3 \\ x_4 & \\ & x_5 \\ x_6 & x_6 \end{array}\right]$$

Figure 2.2: A local distribution, a fully replicated distribution, a linear distribution, and a specific distribution of a vector with some replicated elements for two processors (left: processor 0/right: processor 1). The distribution of parallel vectors is defined by an `Epetra_Map`.

2.3 Implementation

In this section, we discuss our parallel implementation of the GDSW preconditioner as an `Epetra_Operator` based on `Trilinos` 12.0; cf. [109]. Note that, for compatibility with other libraries such as `LifeV` [90, 92], which we use for the finite element discretizations, we use the `Trilinos Epetra` package for the parallel linear algebra. The more recent `Tpetra` package provides the same functionality in templated form with improved support for shared-memory parallelism.

2.3.1 Trilinos Software Library

`Trilinos` is an object oriented C++ library which supports features for handling large-scale, complex multi-physics engineering, and scientific problems [109]. Although it is composed of individual, independently maintained packages, which could also be used separately, `Trilinos` presents these packages within a common framework to facilitate the development of efficient parallel scientific applications. The packages include a basic parallel linear algebra infrastructure (`Epetra`, `EpetraExt`), direct linear solvers (`Amesos`), iterative linear solvers (`AztecOO`, `Belos`), a suite of useful tools (`Teuchos`), and preconditioners such as algebraic overlapping Schwarz (`IFPACK`).

`Trilinos` offers very flexible mechanisms to define the parallel distribution of linear algebra objects by using maps (`Epetra_Map`); see Figure 2.2. The distribution can be arbitrary, and a vector entry can be held redundantly by the processors.

For a distributed matrix, i.e., a specialization of `Epetra_Operator` such as an `Epetra_CrsMatrix`, four maps determine the parallel distribution of the matrix and the communication pattern for the application of the matrix to parallel vectors. In particular, the row and the column map determine the distribution of the rows and the columns of the matrix, respectively, and the domain and the range map correspond to the maps of the source and destination vectors. For the multiplication or the summation of matrices, compatible maps are required as well. Such latter operations are part of the package `EpetraExt`, which also contains I/O support for reading and writing files in, e.g., `MATLAB` or `HDF5` formats, a `PETSc` interface for `Trilinos` preconditioners, or a function to form the explicit transpose of a matrix.

To communicate off-process elements of distributed objects, i.e., of matrices and vectors, `Epetra` provides the classes `Epetra_Export` and `Epetra_Import`. By specifying the source and the target map, the content of an object is transferred to a second object with a different distribution. These operations thus correspond to gather and scatter operations.

The `Amesos` package provides object-oriented interfaces to direct solvers (mostly third-party libraries), like `LAPACK` [6] (`Amesos_Lapack`), `MUMPS` [5] (`Amesos_Mumps`), and `UMFPACK` [59] (`Amesos_Umfpack`), whereas `AztecOO` and `Belos` provide implementations of iterative Krylov methods such as conjugate gradients (CG), generalized minimal residual (GMRES), and biconjugate gradients stabilized (BiCGSTAB). The package `Teuchos` includes, among others, tools for smart pointers (`Teuchos::RCP`), parameter lists (`Teuchos::ParameterList`), timers (`Teuchos::TimeMonitor`), and command line processing (`Teuchos::CommandLineProcessor`).

For finite element based implementations, special vector and matrix classes, i.e., `Epetra_FEVector` and `Epetra_FECrsMatrix`, are provided. These classes simplify the parallel assembly compared to the corresponding standard classes `Epetra_Vector` and `Epetra_CrsMatrix` – but we have not used them in our implementation.

In order to apply an operator to multiple vectors, e.g., the multiplication of an operator with multiple vectors in block-Krylov methods or the solution of a linear system with multiple right-hand sides, `Trilinos` provides the `Epetra_MultiVector` class, where a single `Epetra_MultiVector` can contain any number of vectors with the same length and distribution. The class `Epetra_Vector` is in fact a specialization of an `Epetra_MultiVector`, i.e., a multivector with a single column.

2.3.2 Structure of the GDSW Implementation

Our GDSW implementation is structured as follows: we have partitioned the computational work into two separate classes, i.e., the class `SOS` (special over-

```
class Epetra_Operator
{
public:
  virtual                     ~Epetra_Operator ()
  virtual int                  SetUseTranspose (bool UseTranspose)
      = 0
  virtual int                  Apply (const Epetra_MultiVector &X,
       Epetra_MultiVector &Y) const = 0
  virtual int                  ApplyInverse (const
      Epetra_MultiVector &X, Epetra_MultiVector &Y) const = 0
  virtual double               NormInf () const = 0
  virtual const char         * Label () const = 0
  virtual bool                 UseTranspose () const = 0
  virtual bool                 HasNormInf () const = 0
  virtual const Epetra_Comm  & Comm () const = 0
  virtual const Epetra_Map   & OperatorDomainMap () const = 0
  virtual const Epetra_Map   & OperatorRangeMap () const = 0
};
```

Figure 2.3: Public interface of the abstract `Trilinos` class `Epetra_Operator`.

lapping Schwarz preconditioner) and the class `SOSSetUp` (object to perform the setup of the preconditioner). The class `SOSSetUp` computes

- the local overlapping subdomain matrices $\tilde{A}_i$ and
- the global matrix Φ, which contains the coarse basis functions.

In order to build the matrices $\tilde{A}_i$, we first export the fully assembled matrix A to the overlapping distribution (`Epetra_Map`) and then extract the local portions of the matrix.

To build the matrix Φ, the interface components have to be identified (cf. Section 2.3.3), and the interface values and the local discrete harmonic extensions of the coarse basis functions have to be computed (cf. Section 2.3.4).

The class `SOS` is a specialization of the abstract class `Epetra_Operator`, which defines the general interface for parallel operators in `Trilinos`; see Figure 2.3. In this way, the preconditioner is compatible with, e.g., the iterative solver packages of `Trilinos`. The preconditioner class contains the part of the implementation which relates to the application of the preconditioner, i.e.,

- the computation of the coarse matrix A_0 (cf. Section 2.3.5) from Φ and A,
- the factorizations of local overlapping and the global coarse problems (cf. Section 2.3.6), and
- the handling of the parallel application of the preconditioner (cf. Section 2.3.7).

2D Linear Elasticity					
# Subdom.	4	16	64	256	1024
Time	0.6 s	1.1 s	1.3 s	2.4 s	4.5 s
%	0.5	0.7	0.9	1.5	2.6

3D Linear Elasticity								
# Subdom.	8	27	64	125	216	343	512	729
Time	0.2 s	0.8 s	0.7 s	0.8 s	1.0 s	1.1 s	1.6 s	2.2 s
%	0.3	0.4	0.3	0.3	0.4	0.4	0.5	0.8
# Subdom.	1 000	1 331	1 728	2 197	2 744	3 375	4 096	
Time	3.3 s	5.1 s	7.9 s	12.2 s	18.5 s	27.0 s	39.6 s	
%	1.0	1.6	2.3	3.4	4.6	6.3	8.1	

Table 2.1: Identification of the interface components: time in seconds and as a percentage of the total time to solution for a linear problem; GDSW uses `UMFPACK` for the local problems. The number of subdomains is identical to the number of cores. Results are for linear elasticity in 2D with $H/h = 100$ (top) and linear elasticity in 3D with $H/h = 6$ (bottom) using P2 finite elements. For the corresponding scaling results; see [105], and Figures 2.15 and 2.17.

Thus, the class `SOS` can be used to implement any kind of two-level overlapping Schwarz preconditioner, by modifying or replacing the class `SOSSetUp`.

2.3.3 Identification of Vertices, Edges, and Faces in Parallel

As in BDDC or FETI-DP domain decomposition methods [197, 149], in GDSW methods, for the setup of the coarse level, the vertices, edges, and (in 3D) faces of the nonoverlapping domain decomposition have to be efficiently identified in parallel. The parallel procedure described in the following is thus also of interest for FETI-DP and BDDC type preconditioners [197, 149].

We have decided to implement our procedure building purely on the parallel linear algebra tools from `Trilinos`. First, we transfer for all nodes the subdomain numbers they belong to: for each node $x \in \bar{\Omega}$, we communicate the index set

$$N_x = \{ i : x \in \bar{\Omega}_i \}$$

to all processes. This is implemented using `Epetra_Exporter`s and `Epetra_IntVector`s, i.e., parallel vectors of integers.

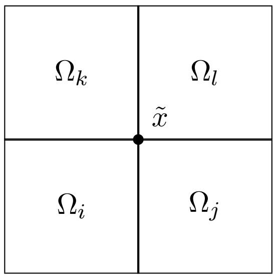


Figure 2.4: A vertex $\tilde{x}$ of the nonoverlapping decomposition.

```
       N      l       k       j       i      1
P1:   0 ... 0 0 0 ... 0 0 0 ... 0 1 0 ... 0 1 0 ... 0
P2:   0 ... 0 0 0 ... 0 1 0 ... 0 0 0 ... 0 0 0 ... 0
P3:   0 ... 0 1 0 ... 0 0 0 ... 0 0 0 ... 0 0 0 ... 0
-----------------------------------------------------
Sum:  0 ... 0 1 0 ... 0 1 0 ... 0 1 0 ... 0 1 0 ... 0
```

Figure 2.5: Computation of the index set N_x for the node $\tilde{x}$ in Figure 2.4: assuming that the subdomains i and j are assigned to process `P1`, the subdomain k is assigned to process `P2`, and the subdomain l is assigned to process `P3`, the local integers $2^i + 2^j$ (`P1`), 2^k (`P2`), 2^l (`P3`), and their summation are shown in binary representation.

In particular, for a node $\tilde{x}$ which belongs to the subdomains Ω_i, Ω_j (assigned to process `P1`), Ω_k (assigned to process `P2`), and Ω_l (assigned to process `P3`), integers are added up in parallel; see Figures 2.4 and 2.5.

Then, the indices i, j, k, and l can be retrieved locally from the binary representation of the sum. In general, the total number of subdomains N is larger than the size of an integer (typically 32 bit or 64 bit), such that multiple integers have to be used for one node.

Since N_x has to be computed for all nodes in $\bar{\Omega}$, `Epetra_IntVector` vectors of length "number of nodes", with each entry of the vector corresponding to one node, have to be added up among neighboring processes. The vectors are distributed according to the map of the nonoverlapping decomposition which overlaps only in the interface degrees of freedom. The parallel summation of the integers is then performed using an `Epetra_Export` object. Therefore, we first export the vectors to a uniquely distributed `Epetra_Map`, summing up the overlapping entries, and then import the vector back to the original map.

When the index sets are available locally on all processes, the multiplicity of each node, which is just the cardinality of N_x, can be computed locally.

To identify the interface components, we categorize all nodes according to the sets N_x, i.e., all nodes which belong to the same subdomains are categorized in the same interface component. A single node which belongs to a set of subdomains (more than one) is a vertex of the decomposition, and all nodes which belong to the same two subdomains form a face. Among the remaining nodes, all nodes which belong to more than two subdomains, reside on an edge of the decomposition. All nodes with multiplicity one are categorized as interior nodes.

This procedure does not require any geometric information and makes use of the `Epetra_Map` of the nonoverlapping decomposition. All operations can be performed locally once the index sets N_x have been communicated. The map is typically available from the partitioning of the mesh (or of the system matrix). However, for standard mesh partitioners such as `ParMETIS` (see [123]), we can typically not guarantee that the interface components are connected. Note that the ordering is not needed for the computation of the GDSW coarse space but could be necessary in order to compute other kinds of basis functions; e.g., in order to compute the *vertex-specific* basis functions of the ACMS coarse space in Chapter 5.

Let us note that the procedure described here has quadratic complexity but with a small constant: the number of export/import operations grows linearly with the number of subdomains, i.e., for 1 024 subdomains 32 export and import operations with 32 bit integers are needed. As this operation tends to be latency-dominated it can be beneficial to increase the block size from 32 bits (int) to 64 bits (long) or even 128 bits.

In Table 2.1, we report the timings for the communication and the identification of the interface components (using 32 bit integers) for linear elastic model problems in 2D and 3D. It can be observed that the timings grow significantly when increasing the number of cores to 4 096. However, the times are still small compared to the total time to solution, especially for nonlinear problems, where this procedure has to be performed only once, in a preprocessing step. For a smaller number of subdomains ($< 1\,000$), the timings are clearly negligible, even for linear problems.

2.3.4 Computation of the Coarse Basis Functions

The coarse basis functions are given by their interface values and the discrete harmonic extension to the interior degrees of freedom of the nonoverlapping subdomains. Thus, when computing the full coarse space, the index sets of the interface components and the list of coordinates of the local mesh partition are needed. Then, the interface values of the coarse space functions can be computed locally, according to (1.48) and (1.49) in 2D, and (1.50) and (1.51) in 3D.

Numerical scalability of the preconditioner can be observed without using rotations in the coarse space; see the numerical results in Section 2.5. For a large number of subdomains, savings in the coarse problem can overcompensate an increase in the number of iterations. When neglecting the rotations in the coarse space, the list of coordinates is not needed.

We insert the interface values into local `Epetra_MultiVectors` $\Phi_\Gamma^{(i)}$. In order to obtain the values at the interior degrees of freedom, discrete harmonic extensions have to be computed, cf. (2.4). We solve the local linear systems

$$-A_{II}^{(i)} \Phi_I^{(i)} = A_{I\Gamma}^{(i)} \Phi_\Gamma^{(i)}$$

for the `Epetra_MultiVectors` $\Phi_I^{(i)}$ using some direct solver, e.g., `MUMPS` (through the `Trilinos` interface `Amesos_Mumps`). The matrices $A_{II}^{(i)}$ and $A_{I\Gamma}^{(i)}$ can be extracted from the fully assembled matrix A.

Having the `Epetra_MultiVectors` $\Phi_\Gamma^{(i)}$ and $\Phi_I^{(i)}$ at hand, we extract the values from the vectors and insert them into the global `Epetra_CrsMatrix` Φ.

2.3.5 Computation of the Coarse Operator

The computation of the coarse operator is a triple matrix product and thus equivalent to the construction of Galerkin coarse operators (RAP product) in Algebraic Multigrid (AMG) methods. This step is currently implemented using the matrix-matrix products from the `EpetraExt` package. Potentially, we could make use of the corresponding routine from the `Trilinos` AMG package `ML` for this operation. However, the `EpetraExt` routines currently seem to outpace the `ML` routines, at least below 4 000 cores [181].

The coarse matrix can be computed either using the fully assembled matrix A, cf. (2.3), or the Schur complement, cf. (2.5). In both cases, global matrix-matrix multiplications have to be performed. Alternatively, A_0 can be computed subdomain-wise, exploiting

$$\begin{aligned} A_0 = \Phi^T A \Phi &= \sum_{i=1}^{N} \Phi^{(i)T} A^{(i)} \Phi^{(i)} \qquad \text{or} \\ A_0 = \Phi_\Gamma^T S_{\Gamma\Gamma} \Phi_\Gamma &= \sum_{i=1}^{N} \Phi_\Gamma^{(i)T} S_{\Gamma\Gamma}^{(i)} \Phi_\Gamma^{(i)}, \end{aligned}$$

```
1 Teuchos::RCP<Epetra_CrsMatrix> A0tmp(new Epetra_CrsMatrix(Copy,Phi
      ->DomainMap(),Phi->ColMap().NumMyElements()));
2 Epetra_CrsMatrix B(Copy,K->RowMap(),K0->NumMyRows());
3 EpetraExt::MatrixMatrix::Multiply(*A,false,*Phi,false,B);
4 EpetraExt::MatrixMatrix::Multiply(*Phi,true,Tmp,false,*A0tmp);
5 Teuchos::RCP<Epetra_Export> Export0(new Epetra_Export(Phi->
      DomainMap(),*CoarseMap));
6 Teuchos::RCP<Epetra_CrsMatrix> A0(new Epetra_CrsMatrix(Copy,*
      CoarseMap,K->NumGlobalNonzeros()/K->NumGlobalRows());
7 A0->Export(*A0tmp,*Export0,Insert);
8 A0->FillComplete();
```

Figure 2.6: Building the coarse matrix A_0 using `Trilinos`.

where $A^{(i)}$ and $\Phi_\Gamma^{(i)}$ are the local subdomain matrix and the restriction of Φ_Γ to the i-th subdomain, respectively, and $S_{\Gamma\Gamma}^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} \left(A_{II}^{(i)}\right)^{-1} A_{I\Gamma}^{(i)}$ is the local Schur complement. Here, the local (Neumann) subdomain matrices $A^{(i)}$ are needed, which cannot be extracted from the fully assembled matrix A. Thus, even though the matrix-matrix multiplications can be computed purely locally, their use leads to a less general implementation since they depend on the availability of the subdomain matrices $A^{(i)}$.

The representation $\Phi_\Gamma^T S_{\Gamma\Gamma}\Phi_\Gamma$ makes better use of a priori knowledge and involves operators with smaller dimension. However, in the $\Phi^T A\Phi$-approach, a better use of a priori knowledge could also be made, i.e., it is known a priori that in (2.5) the upper block in $\begin{bmatrix} 0 \\ S_{\Gamma\Gamma}\Phi_\Gamma \end{bmatrix}$ is zero and, moreover, the product $\Phi_\Gamma A_{I\Gamma} A_{II}^{-1} \cdot 0$ does not need to be computed. Currently, however, we do not exploit this knowledge.

For the computations, we resort to the representation $A_0 = \Phi^T A\Phi$; see (2.3). The corresponding lines of code are listed in Figure 2.6, where the parallel matrix-matrix multiplications $Z := A\Phi$ and $A_0 = \Phi^T Z$ can be found in lines 3 and 4, respectively. The resulting matrix A_0 is stored in a temporary matrix `A0tmp` which is then finally assembled using the unique map `CoarseMap` and an `Epetra_Export` object; see line 7 in Figure 2.6.

Let us briefly comment on the performance of the triple matrix-matrix multiplication $\Phi^T A\Phi$ using `EpetraExt`. For instance, in the scalability study presented in Figure 2.20, for the GDSW preconditioner with full coarse space and 4 096 subdomains and cores, the timings for the computation of the matrix-matrix multiplications are 1.8 s and 2.4 s, respectively. These times correspond to only about 2% of the total runtime.

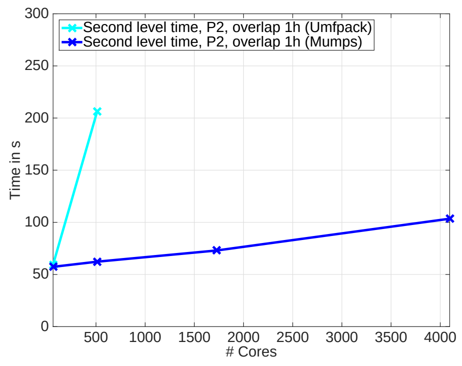


Figure 2.7: Comparison of timings for the coarse level using `MUMPS` (`Amesos_Mumps`) and `UMFPACK` (`Amesos_Umfpack`) as the coarse level solver. Already for a number of $12^3 = 1\,728$ processes, `Amesos_Umfpack` runs out of memory.

2.3.6 Factorizations of the Local and the Coarse Problems

When setting up the preconditioner, factorizations of the local overlapping matrices $\tilde{A}_i$ and the coarse matrix A_0 have to be performed. Note that the matrices $\tilde{A}_i$ are first extracted from A and then stored in local (i.e., sequential) `Epetra_CrsMatrix` objects. On the other hand, A_0 is a parallel, globally distributed matrix.

We perform the factorizations of the local matrices using `MUMPS` (through `Amesos_Mumps`), an MPI parallel multifrontal direct solver, in serial mode. The coarse matrix can be factorized using either `MUMPS` as well, or using `UMFPACK` (`Amesos_Umfpack`), which is a serial multifrontal direct solver. `MUMPS` has a limited range of parallel scalability, especially for three-dimensional problems, due to the superlinear complexity of the (parallel) algorithm. However, `MUMPS` is usually faster (sometimes significantly) than `UMFPACK` and larger systems can be solved as a consequence of lower memory consumption; see Figure 2.7.

The number of processes for solving the coarse problem with `MUMPS` is chosen in accordance to

$$\frac{1}{2}\left(1+\min\left\{\texttt{NumProcs},\max\left\{\texttt{NumRows}/10\,000,\texttt{NumNonzeros}/100\,000\right\}\right\}\right),$$

with `NumNonzeros` being the number of non-zero entries of the replicated coarse matrix in which the off-process entries are not assembled yet. In most cases presented in this thesis, the formula yields roughly the same number of processes as

$$1+\min\left\{\texttt{NumProcs},\max\left\{\texttt{NumRows}/10\,000,\texttt{NumNonzeros}/100\,000\right\}\right\}$$

using the number of non-zero entries of the corresponding assembled coarse matrix. Both formulae are typically dominated by the number of non-zero entries; in Figure 2.22, i.e., for unstructured domain decomposition in 3D, the average number of non-zeros per row can be as large as 820. The formula is proposed in `Amesos_Mumps`, and the distribution and communication can be performed by `Amesos_Mumps`, accordingly.

However, `Amesos_Mumps` distributes the matrices according to a unique linear `Epetra_Map`. Since the coarse matrix of the GDSW preconditioner is distributed according to the partition of the mesh, the redistribution of the matrix in `Amesos_Mumps` can be expensive for large coarse problems. Thus, we have modified `Amesos_Mumps` such that it uses the unique distribution given by our implementation. For the case of 4 096 subdomains in Figure 2.18, in this way, the total time for the factorization phase of the coarse matrix could be accelerated from 11.6 s to 1.3 s.

In order to save memory and computational work, we employ the symmetric mode of `MUMPS` for the coarse problem. In this case, `MUMPS` expects to receive only the lower or the upper triangular part of the matrix (including the diagonal) as an input. This feature is not provided in `Trilinos` version 12.0, and had to be added to `Amesos_Mumps` by the author of this thesis.

In parallel, the symmetric mode of `MUMPS` accelerates the use of `Amesos_Mumps` only by a factor of 1.09 for 4 096 subdomains in the weak scalability run presented in Figure 2.18. However, we would expect a factor of approximately two for `MUMPS` in serial mode. Since `LifeV` does not symmetrize the matrices when applying Dirichlet boundary conditions, we use the general (non symmetric) mode for the local problems.

2.3.7 Application of the Preconditioner to a Vector or Multivector

The application of the preconditioner is implemented in two separate steps, i.e., the application of the first and the application of the second level. The application of the first level to a vector v involves the parallel summation of the local contributions, i.e.,

$$\sum_{i=1}^{N} R_i^T \tilde{A}_i^{-1} R_i v.$$

Here, the multiplications with R_i and R_i^T are just the restrictions to the overlapping subdomains and corresponding prolongations to the whole domain. Again, we use `Trilinos` exporters/importers to handle the corresponding communication. The application of the inverse $\tilde{A}_i^{-1}$ is performed using `MUMPS` in serial or `UMFPACK` on the local vector $R_i v$; cf. Section 2.3.6.

The computation of the coarse correction involves

$$\Phi A_0^{-1} \Phi^T v,$$

i.e., the multiplication of v by Φ^T, the solution of the coarse problem using either a serial direct solver (`UMFPACK`) or a parallel direct solver (`MUMPS`), cf. Section 2.3.6, and the multiplication by Φ.

Note that the default behavior implemented in `Amesos_Mumps` and `Amesos_Umfpack` is to transfer the right-hand side vector to rank 0 before performing the solution step using the corresponding direct solver, i.e., `MUMPS` or `UMFPACK`, respectively. This transfer is, for our use, very costly in terms of time, although only one vector is communicated. Instead, we transfer the right-hand side to an intermediate set of processes reducing the time significantly; we use the same set of ranks as for the distribution of the coarse matrix, cf. Section 2.3.5. For, e.g., the case of 4 096 subdomains, in the weak scalability study in Figure 2.18, the transfer of right-hand side vectors took 128.5 seconds in total, using the default redistribution to rank 0 in `Amesos_Mumps`. However, when manually redistributing the vectors according to our new strategy, the time can be reduced to only 2.1 seconds.

Finally, the corrections computed on both levels are summed,

$$M_{\text{GDSW}}^{-1} v = \Phi A_0^{-1} \Phi^T v + \sum_{i=1}^{N} R_i^T \tilde{A}_i^{-1} R_i v.$$

Concurrent computations of the levels, although not currently implemented, would generally be possible in the additive preconditioner but not in the hybrid version, which, on the other hand, is often numerically more efficient.

2.3.8 User-Interface of the Preconditioner

As shown in the code snippet in Figure 2.8, the use of the preconditioner requires only a few lines of code. In particular, the preconditioner object from the class `SOS` and the set up object from the class `SOSSetUp` have to be created.

To construct these objects, cf. lines 2 and 3, the following data is needed: `M_DomainMap` and `M_RangeMap` are unique domain and range maps of the preconditioner as a specialization of the `Epetra_Operator`. The integers `numVectors` and `numSubdomainsPerProcess` are the number of vectors stored in the `Epetra_MultiVector` to which the preconditioner is applied to and the number of subdomains which are assigned to one process, respectively. When using the preconditioner in an FSI simulation in `LifeV`, both integers

```
 1  if (useFirstLevel || useSecondLevel) {
 2    Teuchos::RCP<SOS::SOS> M_SOS(new SOS::SOS(numVectors,
          numSubdomainsPerProcess,M_DomainMap,M_RangeMap));
 3    Teuchos::RCP<SOS::SOSSetUp> M_SOSSetUp(new SOS::SOSSetUp(
          numSubdomainsPerProcess,dimension,dofs,M_rowMatrixTeuchos,
          M_DomainMap));
 4  }
 5  if (useFirstLevel) {
 6    M_SOSSetUp->FirstLevel(M_ProcessMapOverlap);
 7  }
 8  if (useSecondLevel) {
 9    M_SOSSetUp->SecondLevel(M_ProcessMapNodes,M_ProcessMap,SOS::
          LifeVOrdering,M_LocalDirichletBoundaryDofs,"Mumps",
          useRotations,M_LocalNodeList);
10  }
11  if (useFirstLevel || useSecondLevel) {
12    M_SOSSetUp->SetUpPreconditioner(M_SOS,"Mumps",
          secondLevelSolverParamterList,Type);
13  }
14  if (Print) {
15    M_SOS->Print(std::cout);
16  }
```

Figure 2.8: Lines of code to call the GDSW preconditioner (SOS) inside the FSI code which was implemented using `LifeV`.

are set to one, i.e., each application involves only one vector and we use one subdomain per MPI process. The integers `dimension` and `dofs` correspond to the spatial dimension of the considered problem and to the number of degrees of freedom per node, respectively, and `M_rowMatrixTeuchos` is the pointer (`Teuchos::RCP`) to the `Epetra_CrsMatrix` corresponding to A.

In order to set up the first level of the preconditioner, for the case of one subdomain per process, only the `Epetra_Map` corresponding to the overlapping decomposition is needed, cf. line 6 in Figure 2.8.

To build the second level, cf. line 9, more data is needed: the `Epetra_Map` for the distribution of the nodes, `M_ProcessMapNodes`, the corresponding `Epetra_Map` for the distribution of the degrees of freedom, `M_ProcessMap`, a local list of Dirichlet boundary degrees of freedom, `M_LocalDirichletBoundaryDofs` (they are treated as interior degrees of freedom), a string stating the solver which is used to compute the discrete harmonic extensions (here `MUMPS`), a boolean variable, `useRotations`, which enables the use of rotations for the coarse space, and the list of coordinates of the local partition of the mesh, `M_LocalNodeList` (needed for the computation of the rotations). Additionally, the parameter `SOS::LifeVOrdering` specifies the ordering of the degrees of freedom in the matrix. In particular, in `LifeV`, the degrees of freedom are ordered dimension-wise (i.e., all x coordinates first), in contrast to most other codes where a nodal ordering is used. `LifeV` provides implementations of finite element discretizations used for the simulations in the next section. It is also the base of the FSI software (see also Chapter 1) which is employed for the simulations in the next chapter, where the parallel implementation of the GDSW preconditioner is used as a preconditioner for the structural block in FSI. Therefore, the ordering of the degrees of freedom used in `LifeV` is supported by our implementation.

Calling the `SetUpPreconditioner` method in line 12 finally sets up the preconditioner object. Here, all information regarding the first and the second level are handed from the `SOSSetUp` object to the preconditioner object, the first level solver is set (here `"Mumps"`), and a `Teuchos::ParameterList secondLevelSolverParamterList` including the specific configuration of the solver for the coarse problem (here `"Mumps"` and its configuration) is specified.

Finally, the parameter `Type` specifies whether the additive or the hybrid version of the preconditioner is used. The lines 14 to 16 are optional. Here, information about the state of the preconditioner object is printed.

2.3.9 Third-Party Libraries

In addition to the software library `Trilinos` [109], we make use of other third-party libraries. For mesh partitioning, we use `ParMETIS` [123]. The problems corresponding to the first level are solved using `MUMPS` [5] (version 4.10.0) in sequential mode if not marked otherwise. The coarse level is always solved using `MUMPS` [5] in parallel mode. For some experiments, we have used `UMFPACK` [59] (version 5.3.0) for the first level problems. Of course, other serial or sequential sparse direct solvers could be used as well.

Note that, for efficiency, some modifications in the `Amesos_Mumps` interface class were performed for the results in Sections 2.5.2 and 2.5.4; see Sections 2.3.6 and 2.3.7. The finite element discretizations, which are used in the next section, are based on the finite element library `LifeV` (version 3.8.8).

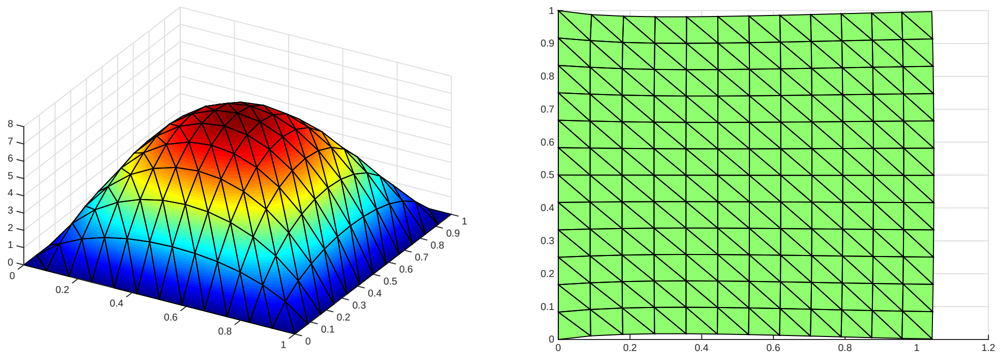


Figure 2.9: Solutions of the Laplacian (left) and the linear elastic (right) model problems using linear finite elements and a mesh with $h = 1/12$.

2.4 Model Problems

As benchmark problems for our implementation of the two-level overlapping Schwarz GDSW preconditioner, we consider different model problems, i.e., a Laplacian and a linear elastic problem in 2D and 3D, and a fluid-structure interaction (FSI) model problem in 3D.

The Laplacian and the linear elastic model problems are rather simple problems on structured grids which are used in order to study the parallel scalability of our software and to identify potential scalability limits. We examine the effectiveness of the coarse space (we also consider the first level only) and structured as well as unstructured domain decompositions. In addition, we compare the additive (standard) and the hybrid version of the preconditioner. The corresponding results are presented in Section 2.5 and include the first results for the performance of our preconditioner from the short proceedings article [105] as well as the more extended set of results (especially, in three dimensions) from the journal publication [107] about the implementation of the GDSW preconditioner.

The FSI problem is a substantially more sophisticated problem, including monolithic coupling, time dependence, and nonlinearities in fluid and (possibly) structure; cf. Chapter 1. We consider different material models for the structure, i.e., a linear elastic (cf. Section 1.2.2), a Neo-Hookean (cf. Section 1.2.3), and a highly nonlinear anisotropic hyperelastic material model (Ψ_A, cf. Section 1.2.4). The FSI problem is considered to study the robustness of our preconditioner with respect to (almost) realistic applications and to investigate whether the use of the GDSW preconditioner can improve the performance of our FSI framework introduced in Chapter 1; previously an algebraic one-level overlapping Schwarz preconditioner has been used. The corresponding results from [106] and [107] are shown in Chapter 3.

2.4.1 Laplacian

We consider a simple scalar elliptic problem in 2D or 3D: find $u \in H^1(\Omega)$, such that

$$\begin{aligned} \Delta u &= 1 && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega \end{aligned} \tag{2.10}$$

with $\Omega = [0;1]^2 \subset \mathbb{R}^2$ or $\Omega = [0;1]^3 \subset \mathbb{R}^3$. The solution for the 2D case is shown in Figure 2.9 (left).

2.4.2 Linear Elasticity

Secondly, we consider, in 2D and 3D: find $u \in (H^1(\Omega))^d$, such that

$$\begin{aligned} \operatorname{div} \boldsymbol{\sigma} &= f && \text{in } \Omega, \\ \mathbf{u} &= 0 && \text{on } \partial\Omega_D \end{aligned} \tag{2.11}$$

with the dimension d (either 2 or 3), Lamé parameters $\lambda = \frac{1.0}{2.6}$ and $\mu = \frac{0.3}{0.52}$, and $\partial\Omega_D = \partial\Omega \cap \{x = 0\}$.

For the 2D case, the right-hand side is $f = [0.1, 0]^T$ and $\Omega = [0;1]^2 \subset \mathbb{R}^2$, whereas for the 3D case, the right-hand side is $f = [0.1, 0, 0]^T$ and $\Omega = [0;1]^3 \subset \mathbb{R}^3$. The solution for the 2D case is shown in Figure 2.9 (right).

2.5 Numerical Results

In this section, we report numerical results for the Laplacian and the linear elastic benchmark problems described in Section 2.4.

In [105], we have focussed on the parallel scalability of our implementation of the GDSW preconditioner for a Laplacian and a linear elastic model problem in 2D. Also, some first 3D results were given, i.e., weak scalability for linear elasticity in 3D. In the computations in [105], `UMFPACK` was used for the first level.

We have significantly extended these results in [107], mostly by results in 3D. In particular, we have investigated the performance of our implementation of the GDSW preconditioner with respect to structured and unstructured decompositions, to reduced coarse spaces, and to the hybrid version of the preconditioner. In addition to that, the implementation has been improved compared to [105] using `MUMPS` on both levels and including modifications to `Amesos_Mumps`; see Sections 2.3.6 and 2.3.7. To highlight the improvements in our implementation, we present the results from [105], cf. Sections 2.5.1 and 2.5.3, as well as the improved results from [107], cf. Sections 2.5.2 and 2.5.4.

If not stated otherwise, in our numerical experiments, we have a one-to-one correspondence of subdomains and processor cores, although this is not necessary in our implementation. For all problems, we use GMRES as a Krylov method with a relative stopping criterion of 10^{-7}, the GMRES iteration is stopped when $\|r_n\|_2/\|r_0\|_2 < 10^{-7}$. With $\| \cdot \|_2$ we refer to the corresponding vector norm. Note that, since `LifeV` does not ensure symmetry of the system matrix when implementing the Dirichlet boundary conditions, we cannot use conjugate gradients (CG) even for our symmetric positive definite problems. The use of GMRES also simplifies comparisons with realistic application problems, where GMRES is often the Krylov method of choice. For the same reasons, we also cannot use the symmetric mode of `MUMPS` for the first level problems. For the coarse problem, however, we use `MUMPS` in symmetric mode for the symmetric positive definite model problems.

The numerical results in this section are generated on the JUQUEEN BG/Q (Blue Gene/Q) supercomputer [191] at JSC Jülich. We use the `clang` 4.7.2 compiler and the Engineering and Scientific Subroutine Library (`ESSL`) 5.1. A node of the JUQUEEN supercomputer has 16 cores (Power BQC, 1.60 GHz) and 16 GB of RAM.

Let us briefly comment on the performance of the sparse direct solvers `UMFPACK` and `MUMPS` on the BG/Q (Power BQC, 16 cores, 1.60 GHz) architecture compared to the Intel Ivy Bridge (Intel Xeon E5-2650 v2 @ 2.60 GHz)

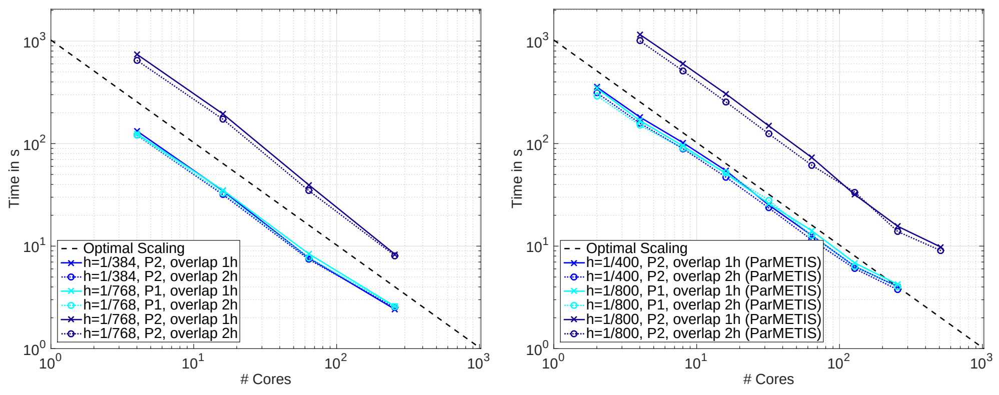


Figure 2.10: Strong parallel scalability on JUQUEEN using the GDSW preconditioner for the model problem of the Laplacian in 2D, cf. (2.10): structured decomposition (left), `ParMETIS` decomposition (right). Using `UMFPACK` for the first level.

architecture for our subdomain matrices in 3D of size around 50 000: using `MUMPS`, the local subdomain matrix (assembled using `LifeV`; $H/h = 12$; piecewise quadratic (P2) finite elements; size $46\,875 \times 46\,875$; on average 59 non-zeros per row) is factorized (including the solution of one right-hand side) on JUQUEEN in 37.30 s and on Intel Ivy Bridge in 9.64 s using 918 MB of memory. Optimized `BLAS` libraries are used, i.e., `ESSL` on JUQUEEN and `MKL` on Ivy Bridge. The performance of `UMFPACK` is, surprisingly, significantly worse for the P2 matrices of this size from `LifeV`: using `UMFPACK`, the same matrix as above is factorized in 203.94 s on JUQUEEN and in 28.43 s on Ivy Bridge using 2027 MB of memory. We therefore use `Mumps` in our numerical experiments, except where explicitly noted otherwise.

2.5.1 Strong Scalability in 2D Using Umfpack as the First Level Solver

Results of strong parallel scalability tests using `UMFPACK` as the first level solver are shown in Figures 2.10 and 2.11 for the Laplacian and linear elasticity, respectively, in 2D. For both problems, we present results for structured and unstructured (`ParMETIS`) domain decompositions. Note that the results are computed using the state of the implementation from [105], whereas only the results for the linear elastic model problem (see Figure 2.11) are presented therein.

For both model problems, we observe very good strong scalability for different problem sizes and with negligible deviations for different sizes of overlap ($1h$ or $2h$). Since we increase the number of subdomains with the number of processes,

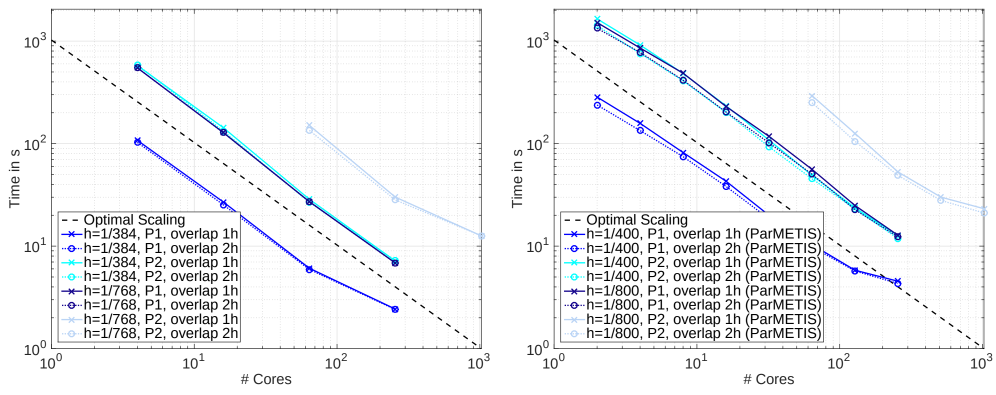


Figure 2.11: Strong parallel scalability using the GDSW preconditioner for the model problem of linear elasticity in 2D, cf. (2.11): structured decomposition (left), `ParMETIS` decomposition (right). Using `UMFPACK` for the first level.

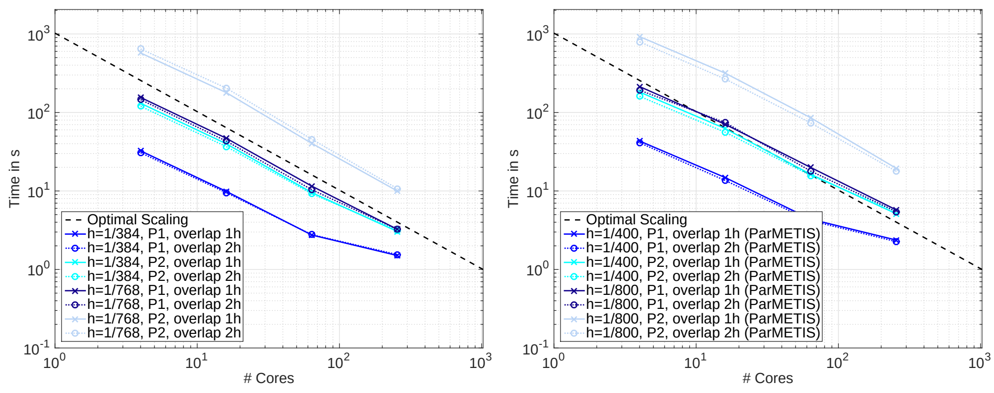


Figure 2.12: Strong parallel scalability on JUQUEEN using the GDSW preconditioner for the model problem of the Laplacian in 2D, cf. (2.10): structured decomposition (left), `ParMETIS` decomposition (right). Using `MUMPS` for both levels.

we benefit from the superlinearly increasing speed of the sparse direct solvers when the subdomain size is decreased.

2.5.2 Strong Scalability in 2D Using Mumps as the First Level Solver

In Figures 2.12 and 2.13, we present results of strong scaling studies for the Laplacian in 2D and for linear elasticity in 2D, respectively, using the improved implementation from [107]. In particular, in contrast to the results from Sec-

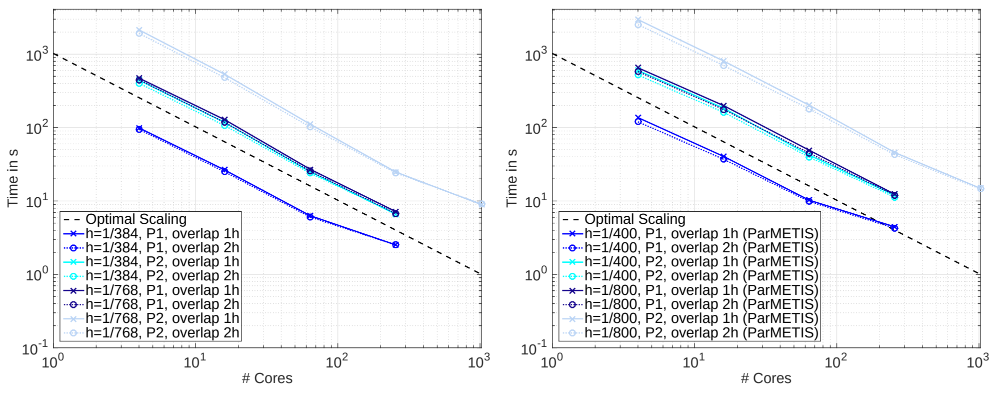


Figure 2.13: Strong parallel scalability on JUQUEEN using the GDSW preconditioner for the model problem of linear elasticity in 2D, cf. (2.11): structured decomposition (left), `ParMETIS` decomposition (right). Using `MUMPS` for both levels.

# Subdomains	4	16	64	256	1024
Total problem, P2 finite elements	160 801	641 601	2 563 201	10 246 401	40 972 801
Avg. first level, P2, overlap 1h	41 207.5	41 612.6	41 815.7	41 917.3	41 968.1
Avg. first level, P2, overlap 2h	42 020	42 837.8	43 248.7	43 454.7	43 557.8
Coarse level	5	33	161	705	2 945
Avg. first level, P2, overlap 1h (`ParMETIS`)	41 581.5	41 841.9	42 101.8	42 225.7	42 263.1
Avg. first level, P2, overlap 2h (`ParMETIS`)	42 686.5	43 243.7	43 752.9	43 999.4	44 077.9
Coarse level (`ParMETIS`)	3	45	241	1 129	4 822

Table 2.2: Number of degrees of freedom of the total mesh, coarse and local space dimensions of the GDSW preconditioner for the weak scaling tests in Figure 2.14.

tion 2.5.1, `MUMPS` is used as the first level solver and improvements have been applied to `Amesos_Mumps`; see Sections 2.3.6 and 2.3.7.

Using the improved implementation, we observe very good strong scalability as well, with `MUMPS` being significantly faster only for larger subdomain problems. When increasing the number of subdomains and decreasing the subdomain sizes, the timings for both solvers are comparable. However, `MUMPS` requires significantly less memory, such that larger problems could be computed (for a small number of cores); cf. Figures 2.11 and 2.13.

2.5.3 Weak Scalability Using Umfpack as the First Level Solver

In this section, we present the weak scalability results from [105] using `UMFPACK` as the solver for the first level of the preconditioner.

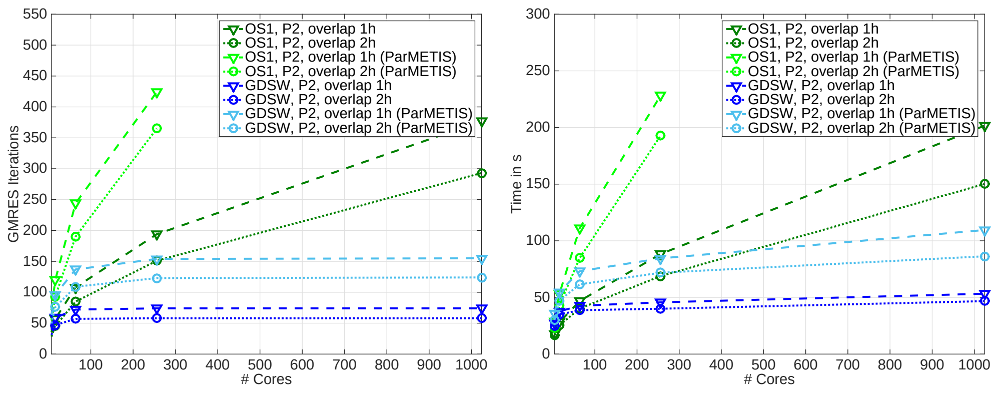


Figure 2.14: Weak parallel scalability on JUQUEEN for model problem of the Laplacian in 2D, cf. (2.10), using P2 finite elements: number of iterations (left), runtimes (right). For the structured and the unstructured decomposition (`ParMETIS`), we have approximately 40 000 degrees of freedom per subdomain. Using `UMFPACK` for the first level.

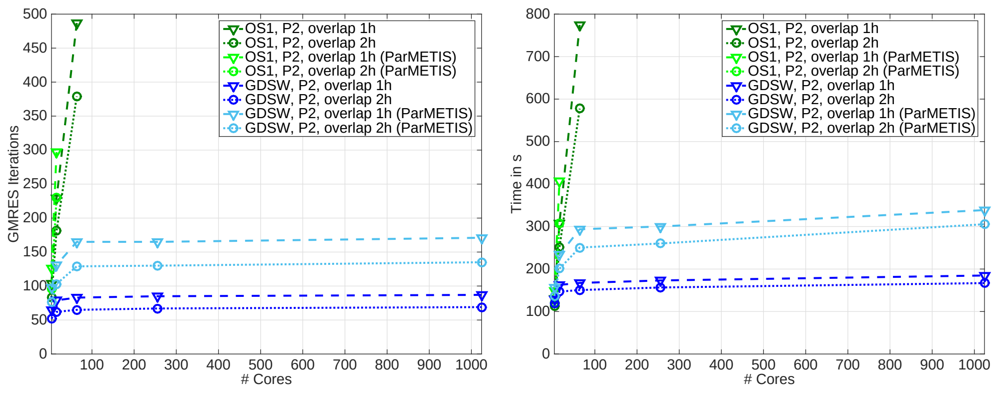


Figure 2.15: Weak parallel scalability on JUQUEEN for model problem of linear elasticity in 2D, cf. (2.11), using P2 finite elements: number of iterations (left), runtimes (right). For the structured and the unstructured decomposition (`ParMETIS`), we have approximately 80 000 degrees of freedom per subdomain. Using `UMFPACK` for the first level.

2.5.3.1 Weak Scalability in 2D Using Umfpack as the First Level Solver

For the weak scalability tests, comparing the GDSW preconditioner with only the first level of the preconditioner (OS1), we use five different meshes with $H/h = 100$ and an increasing number of subdomains; see Tables 2.2 and 2.3 for the corresponding problem sizes. The results of weak scaling tests from 4

# Subdomains	4	16	64	256	1024
Total problem, P2	321 602	1 286 408	5 126 402	20 492 802	81 945 602
Avg. first level, P2, overlap 1h	82 415	83 225.2	83 631.3	83 834.6	83 936.3
Avg. first level, P2, overlap 2h	84 040	85 675.5	86 497.4	86 909.3	87 115.6
Coarse level	14	90	434	1 890	7 874
Coarse level, no rotations	10	66	322	1 410	5 890
Avg. first level, P2, overlap 1h (`ParMETIS`)	83 163	83 683.9	84 203.6	84 451.3	84 526.2
Avg. first level, P2, overlap 2h (`ParMETIS`)	85 373	86 487.4	87 505.8	87 998.7	88 155.9
Coarse level (`ParMETIS`)	9	120	633	2 950	12 567
Coarse level, no rotations (`ParMETIS`)	6	90	482	2 258	9 644

Table 2.3: Number of degrees of freedom of the total mesh, coarse and local space dimensions of the GDSW preconditioner for the weak scaling tests in Figure 2.15 and Figure 2.16.

to 1024 processor cores for both model problems and an overlap of $\delta = 1h$ and of $\delta = 2h$ are presented in Figures 2.14 and 2.15. The GSDW preconditioner is numerically and parallel scalable, i.e., the number of iterations is bounded, both, for structured and unstructured decompositions, and the time to solution grows only slowly. The one-level preconditioner (OS1) does not scale numerically, and the number of iterations grows very fast. Indeed, considering unstructured decompositions for the scalar and the elastic model problems, no convergence is obtained for more than 256 and 16 subdomains, respectively. More precisely, the maximum number of 500 iterations is exceeded in these cases. This is, of course, also due to the comparably small overlap. As a result of the better constant in (2.6), for the GDSW preconditioner, we observe better convergence for structured decompositions. Note that for the case of four (2×2) subdomains, the overlapping subdomains are significantly smaller.

A detailed analysis of different phases of the method is presented for linear elasticity in 2D in Figure 2.16. We consider the standard full GDSW coarse space as well as the GDSW coarse space without rotations, i.e., the rotation is omitted from the coarse space for each edge. This latter case is not covered by the theoretical bound (2.6), but the results indicate numerical and parallel scalability.

2.5.3.2 Weak Scalability for Linear Elasticity in 3D Using Umfpack as the First Level Solver

We present results of weak scalability runs for a linear elastic model problem in 3D from 8 to 4 096 cores. Therefore, we consider a structured decomposition of a cube and use the full GDSW coarse space in 3D. In Figure 2.17, we present the number of iterations and the timings using P2 elements and an overlap δ of one or two elements. The number of iterations seems to be bounded by

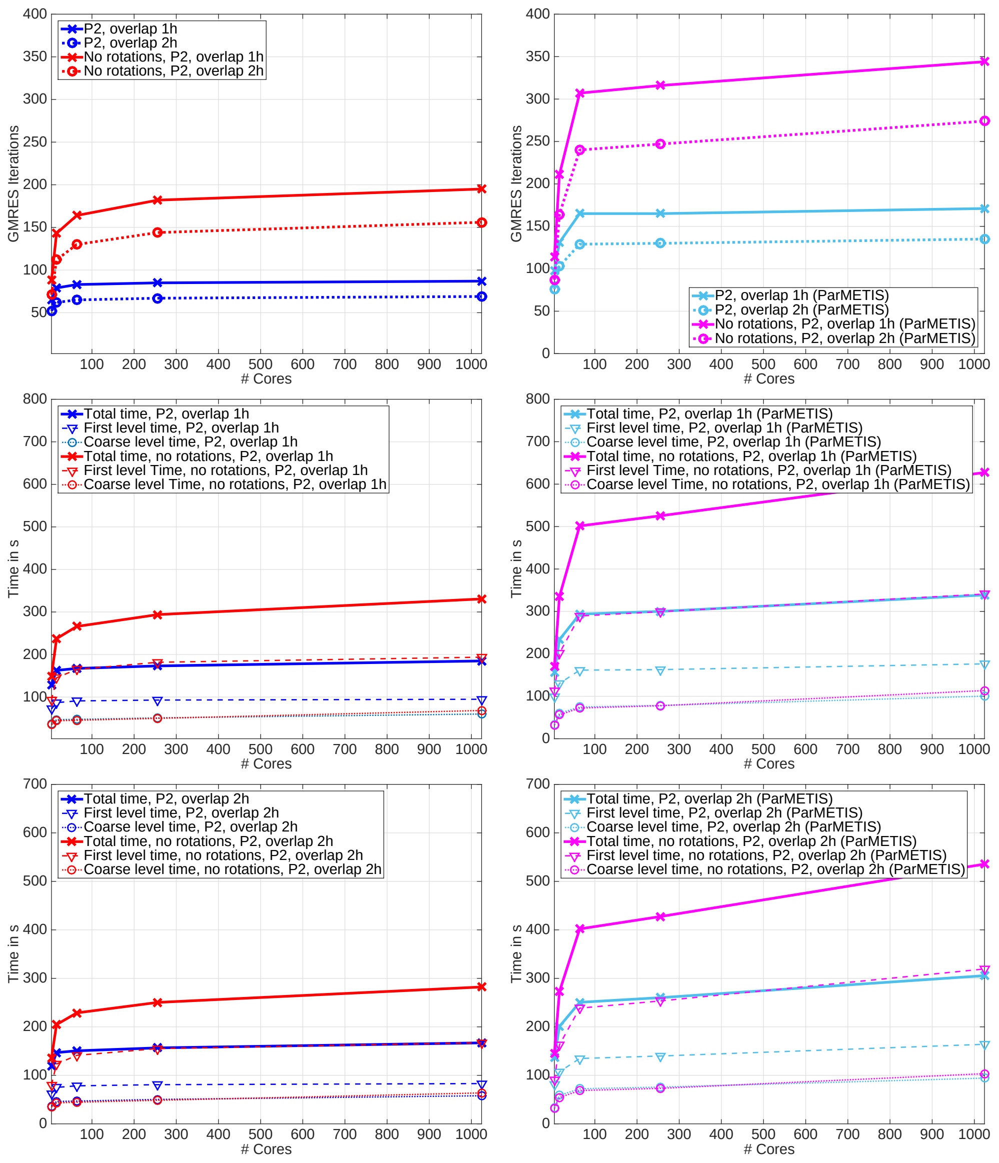


Figure 2.16: Weak parallel scalability using the GDSW preconditioner for the model problem of linear elasticity in 2D, cf. (2.11): structured (left) and unstructured decomposition (right); number of iterations (top), timings for an overlap of $\delta = 1h$ (middle), and timings for an overlap of $\delta = 2h$ (bottom). For the structured and the unstructured (`ParMETIS`) decomposition we use a subdomain size of roughly 40 000 degrees of freedom. Using `UMFPACK` for the first level.

a constant number, whereas the solution time increases, i.e., the cost of the (parallel) sparse direct solver used for the coarse problem is noticeable in 3D.

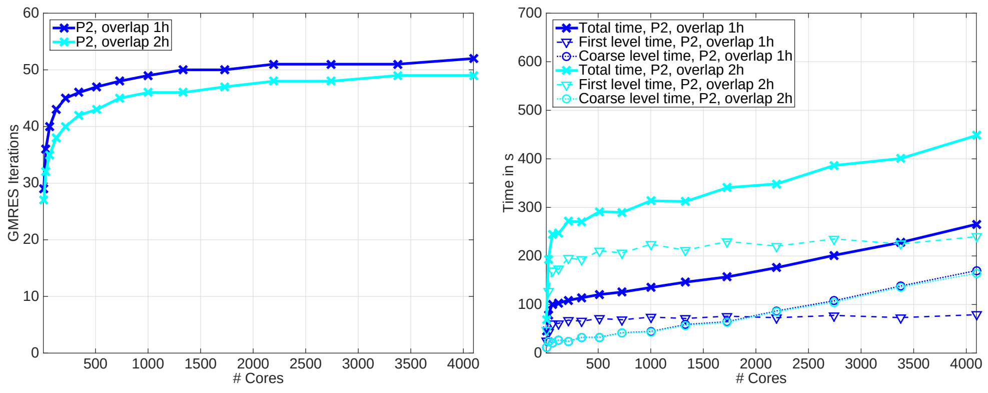


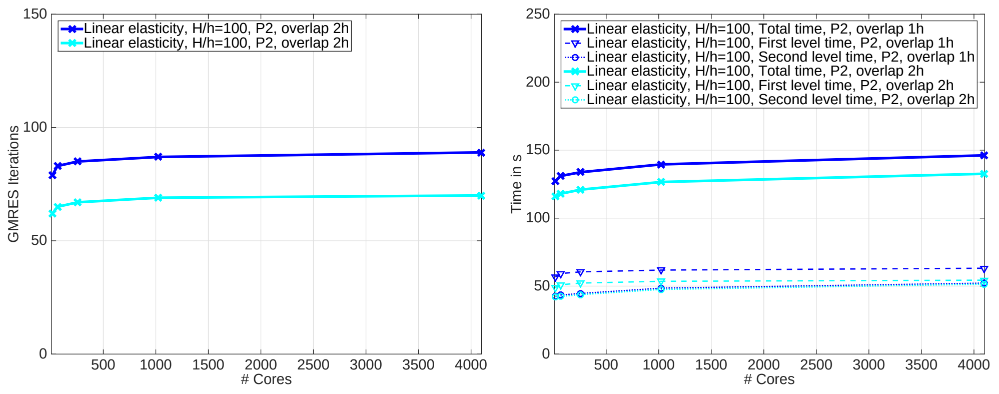
Figure 2.17: Weak parallel scalability using the GDSW preconditioner for the problem of linear elasticity in 3D: number of iterations (left), timings (right). We use a subdomain size of $H/h = 6$ and P2 finite elements. Using `UMFPACK` for the first level.

Note that by using `MUMPS` as the first level solver the total time could be reduced and the parallel scalability could be improved significantly [107]; cf. Section 2.5.4 and, in particular, Figures 2.19 and 2.20 for the corresponding results.

2.5.4 Weak Scalability Using Mumps as the First Level Solver

In this section, we present the improved weak scalability results from [107], where `MUMPS` has been used to solve the first level problems and parts of `Amesos_Mumps` have been modified to improve the parallel performance of the coarse solves; see Sections 2.3.6 and 2.3.7.

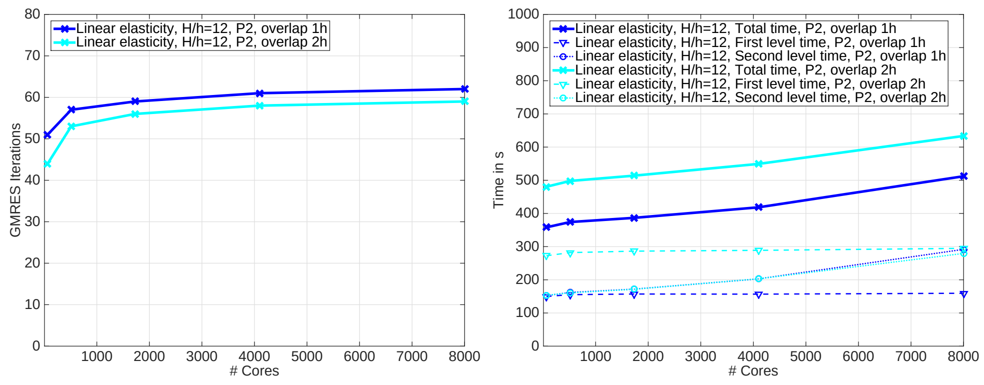
Figures 2.18 and 2.19 show weak parallel scalability for linear elasticity in 2D and 3D, respectively, using structured domain decompositions. For 2D, we achieve a parallel efficiency of 87 % scaling up from a single node (16 cores) to 4 096 cores; see Figure 2.18. We also observe very good weak scalability from 64 to 8 000 processor cores with a parallel efficiency of 70 % for 3D linear elasticity; see, e.g., Figure 2.19 (right). The largest three-dimensional linear elastic problem has 334 million unknowns. In the computations in Figure 2.19, we use four MPI ranks for each node, i.e., we can use up to 4 GB of memory for each MPI rank. Figure 2.20, however, shows weak scalability in three dimensions for smaller overlapping problems, i.e., $H/h = 15$ and P1 finite elements, using 16 MPI ranks for each node, i.e., one MPI rank for each node. The largest problem has 42 million unknowns and can be solved in 102 seconds. Here, the parallel efficiency stays above 55 %.



# Subdomains (# Cores):		16	64	256	1 024	4 096
$\delta = 1h$	Time	127.1 s	131.0 s	133.8 s	139.5 s	146.2 s
	Effic.	91 %	90 %	87 %	83 %	79 %
$\delta = 2h$	Time	116.0 s	117.9 s	121.0 s	126.6 s	132.6 s
	Effic.	**100 %**	98 %	96 %	92 %	87 %

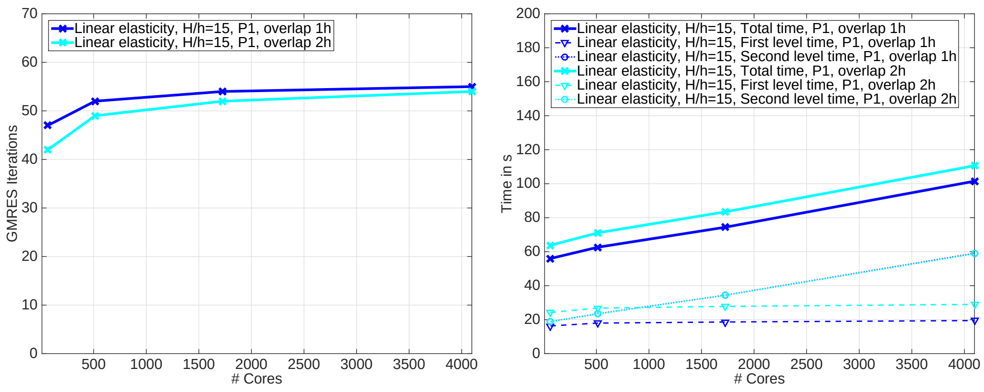
Figure 2.18: Weak parallel scalability on JUQUEEN using the GDSW preconditioner for linear elasticity in 2D: number of iterations (left) and timings (right). We use a subdomain size of $H/h = 100$ and P2 finite elements. Using `MUMPS` for both levels and 16 MPI ranks per node. The baseline for the efficiency is the fastest time on 16 cores.

While the numerical scalability is almost perfect, the parallel efficiency decreases slightly as a result of the increasing time spent on the second level. This is due to the increasing dimension of the coarse space and the superlinear complexity of the parallel direct solver `MUMPS`, which does not exhibit perfect scalability. On the other hand, the time spent on the first level stays almost constant.



# Subdomains (# Cores) :		64	512	1 728	4 096	8 000
$\delta = 1h$	Time	358.1 s	374.1 s	386.7 s	418.4 s	512.1 s
	Effic.	**100 %**	96 %	93 %	86 %	70 %
$\delta = 2h$	Time	480.5 s	497.9 s	513.9 s	549.3 s	633.6 s
	Effic.	75 %	72 %	70 %	65 %	57 %

Figure 2.19: Weak parallel scalability on JUQUEEN using the GDSW preconditioner for linear elasticity in 3D: number of iterations (left), timings (right). We use a subdomain size of $H/h = 12$ and P2 finite elements. Using `MUMPS` for both levels. Four MPI ranks per node. The baseline for the efficiency is the fastest time on 64 cores.



# Subdomains (# Cores) :		64	512	1 728	4 096
$\delta = 1h$	Time	56.0 s	62.6 s	74.4 s	101.5 s
	Effic.	**100 %**	89 %	75 %	55 %
$\delta = 2h$	Time	63.8 s	71.1 s	83.5 s	110.6 s
	Effic.	88 %	79 %	67 %	51 %

Figure 2.20: Weak parallel scalability on JUQUEEN using the GDSW preconditioner for linear elasticity in 3D: number of iterations (left), timings (right). We use a subdomain size of $H/h = 15$ and P1 finite elements. Using `MUMPS` for both levels. 16 MPI ranks per node. The baseline for the efficiency is the fastest time on 64 cores.

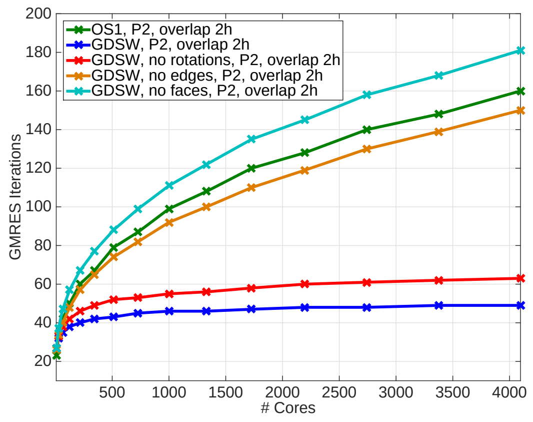


Figure 2.21: Numerical scalability of variants of the GDSW preconditioner for the model problem of linear elasticity in 3D, cf. (2.11): one-level overlapping Schwarz preconditioner (OS1), GDSW with full coarse space, GDSW neglecting rotations, GDSW neglecting edge-based coarse functions, and GDSW neglecting face-based coarse functions. We use structured domain decompositions with $H/h = 6$ and P2 finite elements. The computations are performed on 16 MPI ranks for each node on JUQUEEN and the local overlapping problems are solved using `UMFPACK`.

2.5.4.1 Reduction of the Coarse Space

Figure 2.21 shows a numerical scalability study for the linear elastic model problem in 3D. We present the number of iterations for an overlap of $2h$ for a comparison of the one-level overlapping Schwarz preconditioner, GDSW with the full coarse space, and GDSW neglecting either the rotation-based (cf. (1.49) and (1.51)), the edge-based, or the face-based coarse space functions.

In accordance with the results in [105] for the 2D case, also in 3D, we observe numerical scalability of the GDSW preconditioner even if the rotation-based coarse basis functions are omitted from the coarse space, cf. Figure 2.21. This is remarkable, since the dimension of the coarse space is reduced substantially; see Table 2.4. However, if the coarse space is reduced further (by omitting the edge-based or the face-based basis functions), which is possible in the related FETI-DP and BDDC methods, numerical scalability is lost for the GDSW preconditioner as can be expected from the theory, which is based on a partition of unity; see Figure 2.21.

# Subdomains	Full	No edges	No faces	No rotations
8	105	75	33	57
27	528	348	204	294
64	1 485	945	621	837
125	3 192	1 992	1 392	1 812
216	5 865	3 615	2 625	3 345
343	9 720	5 940	4 428	5 562
512	14 973	9 093	6 909	8 589
729	21 840	13 200	10 176	12 552
1 000	30 537	18 387	14 337	17 577
1 331	41 280	24 780	19 500	23 790
1 728	54 285	32 505	25 773	31 317
2 197	69 768	41 688	33 264	40 284
2 744	87 945	52 455	42 081	50 817
3 375	109 032	64 932	52 332	63 042
4 096	133 245	79 245	64 125	77 085

Table 2.4: Coarse space dimensions of the GDSW preconditioner for the weak scaling tests in Figure 2.21: GDSW with full coarse space, GDSW neglecting edge-based coarse functions, GDSW neglecting face-based coarse functions, and GDSW neglecting rotations.

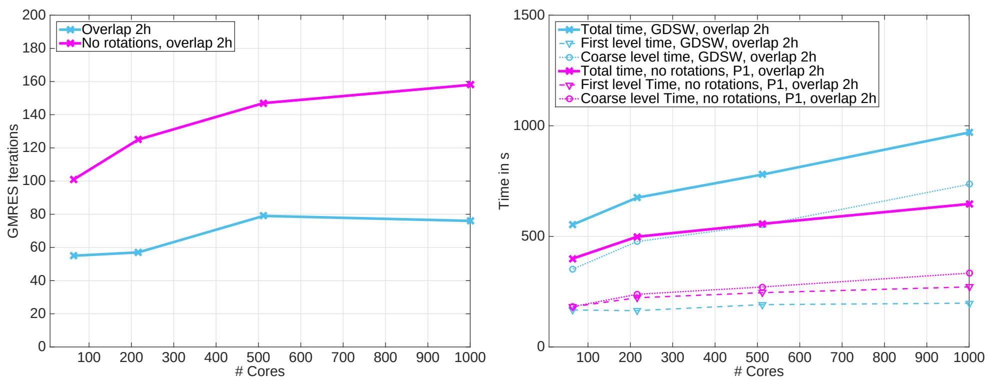


Figure 2.22: Weak parallel scalability using the GDSW preconditioner for the model problem of linear elasticity in 3D for unstructured decompositions, cf. (2.11): number of iterations (left), and timings for overlap $\delta = 2h$ (right). We use P2 finite elements and the subdomain sizes listed in Table 2.5. The computations are performed on 4 MPI ranks for each node on JUQUEEN.

2.5.4.2 Unstructured Domain Decomposition

As expected from the theory, cf. [70, 71], the GDSW preconditioner scales well numerically also for unstructured domain decompositions; see Figure 2.22. For

# Subdomains	64	216	512	1 000
Total problem, P2	1 594 324	5 314 683	12 519 843	24 361 803
Avg. first level, P2, overlap 2h	60 159.8	64 043.3	66 249.1	67 520.5
Full coarse problem	4 611	19 829	53 096	111 296
Coarse problem, no rotations	2 613	11 337	30 516	64 056

Table 2.5: Coarse and local space dimensions of the GDSW preconditioner for the weak scaling tests in Figure 2.22, i.e., linear elasticity in 3D for an unstructured decomposition (`ParMETIS`).

GDSW with the full coarse space, the number of iterations is only slightly larger than for structured domain decompositions, cf. Figure 2.19. Neglecting rotations, the GDSW preconditioner is numerically less robust with respect to an unstructured decomposition.

It is important to note that the dimension of the coarse space grows by more than a factor of three compared to the case of a structured decomposition as a consequence of a much higher number of faces, edges, and vertices in the decomposition; see Figures 2.4 and 2.5. Thus, the time spent on the coarse level grows substantially, and, for a large number of cores and the full coarse space, it starts to dominate the total time of the computation. However, when omitting the rotations from the coarse space, the dimension of the coarse space is reduced significantly (here, by more than a factor of 1.7). Thus, even though the number of iterations is increased by a factor of approximately two, the total time is reduced significantly compared to the full coarse space.

2.5.4.3 Hybrid GDSW Preconditioner

The hybrid version of the GDSW preconditioner (additive on the first level, multiplicative between levels; cf. Section 2.2), involves additional computational work in each iteration, compared to the standard (additive) GDSW preconditioner. However, as can be seen in Figure 2.23, the number of iterations is reduced significantly, i.e., by approximately 10 to 20 iterations. As a consequence, the total computation time for both preconditioners is roughly the same, whereby the time for the hybrid version depends to some extent on the implementation.

In particular, when using the naive implementation of the hybrid preconditioner in the simulation of 8 000 subdomains with an overlap of $1h$ in Figure 2.19, the total simulation time is 538.3 s. Combining the coarse level solve for the application of the coarse Schwarz operator P_0 and the first projection $(I - P_0)$, the time can be reduced to 506.1 s by saving one solve on the coarse level. Fol-

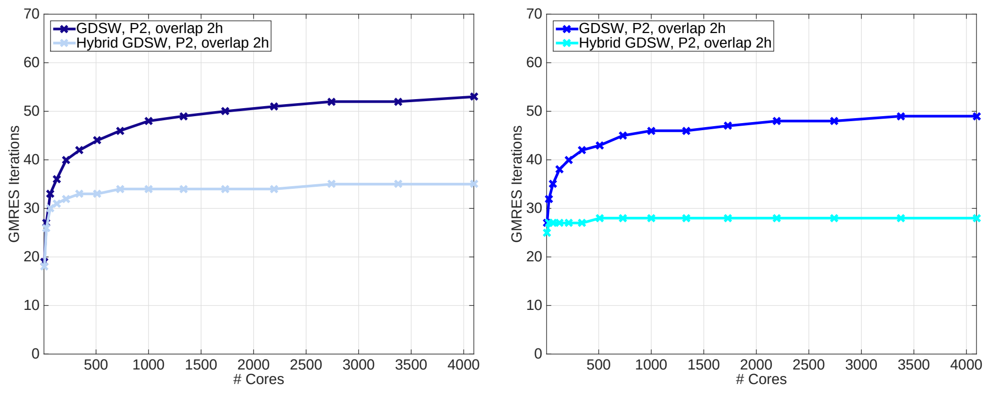


Figure 2.23: Numerical parallel scalability using the GDSW preconditioner for the model problems of the Laplacian in 3D (left), cf. (2.10), and of linear elasticity in 3D (right), cf. (2.11). We use structured domain decomposition with $H/h = 12$ for the Laplacian, $H/h = 6$ for linear elasticity, and P2 finite elements. The computations are performed on 16 MPI ranks for each node on JUQUEEN and the local overlapping problems are solved using `UMFPACK`.

lowing a deflation formulation, the computational cost could be reduced further by omitting the rightmost projection and removing the balancing term from the iteration.

2.6 Conclusion

Our parallel implementation of the GDSW preconditioner is strongly and weakly scalable to thousands of cores for two- and three-dimensional elasticity problems. Very good numerical scalability can be observed, even when neglecting the linearized rotations in the coarse space. For a large number of subdomains (and cores) the cost of the coarse problem becomes significant, even if a parallel sparse direct solver is used. This is especially the case for unstructured domain decompositions, where the coarse space grows faster and is more dense. A two-stage partitioning of the computational domain, as proposed in [142], may help to obtain decompositions with better quality. Techniques to further reduce the size of the coarse space [72] could also be helpful.

Hybrid MPI/OpenMP parallization, i.e., a threaded sparse solver using, e.g., four to eight threads on each subdomain [128] (see also [102, 122]) can serve to extend the scalability beyond the range presented in this thesis by allowing larger subdomains. A simple approach such as using a threaded `BLAS` with `MUMPS` or `UMFPACK` will not be successful.

A parallel multilevel extension may also seem like a natural next step. Improvements in constructing the coarse problem, which are also most important for the unstructured case, may also still be possible, e.g., building on the discussion in Section 2.3.5.

However, even in very challenging model problems (e.g., as a preconditioner for the structural block in FSI simulations) the preconditioner proves to be very robust and significantly faster than our previous algebraic default preconditioner, while being constructable from the assembled system matrix; see Section 3.2. The implementation is very flexible and the use of the preconditioner requires just a few lines of code in the `Trilinos` framework; see Section 2.3.8.

Adaptive coarse spaces may also be of interest to be used for the coarse space of a parallel two-level overlapping Schwarz preconditioner. Therefore, we refer to Chapter 5, where the interface basis functions from the finite element space of a special (multiscale) finite element method are employed in the coarse space of a two-level Schwarz preconditioner on structured decompositions in 2D. There, a serial `MATLAB` implementation of the GDSW preconditioner is used to test the robustness with respect to heterogeneous problems. Whereas, for some heterogeneous problems, the GDSW preconditioner seems to be robust, for more difficult problems additional enrichment of the coarse space is necessary to sustain the robustness. The results for such enriched coarse spaces presented in Chapter 5 are still preliminary, but also a version which can be used in a purely algebraic way, just utilizing the fully assembled global stiffness matrix, is presented. In this regard, basis functions of this type would fit well with the parallel implementation presented here.

3 Application of the GDSW Preconditioner to Fluid-Structure Interaction Problems

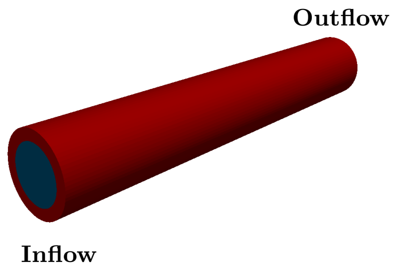
In the simulations in Chapter 1, more precisely, in Section 1.5, one-level algebraic additive overlapping Schwarz preconditioners have been used to approximate the inverses of the fluid, the structural, and the geometry blocks in Fluid-Structure Interaction (FSI). In particular, `Trilinos IFPACK` [173], i.e., a parallel algebraic one-level overlapping Schwarz preconditioner, has been used. However, it is well-known from theory [186, 197] and it has been observed in the last chapter, in Section 2.5, that one-level overlapping Schwarz preconditioners are not numerically scalable. In addition to that, the algebraic overlap makes `IFPACK` less robust than overlapping Schwarz preconditioners with a geometric overlap.

The GDSW preconditioner, a two-level Schwarz preconditioner with geometric overlap, has been proven to be numerically scalable and robust with respect to unstructured decompositions in elasticity problems; see Section 2.5. Thus, it seems natural to utilize our implementation of the GDSW preconditioner, which has already been described in Chapter 2, as a preconditioner for the structure in FSI simulations. Especially for sophisticated material models, cf., e.g., Sections 1.2.4 and 1.2.5, the use of a suitable preconditioner would improve the performance of the FSI algorithm significantly.

In [105], first numerical results of our parallel GDSW implementation for a Neo-Hookean structure in FSI have been presented, comparing the GDSW preconditioner to our default preconditioner for the structural block, i.e., `Trilinos IFPACK`; see also [23, 24] and Chapter 1. In [106], the comparison has been extended to a whole study for pressure wave driven FSI simulations: for different time steps and material models, `IFPACK` has been compared to the first level of our implementation, the GDSW preconditioner neglecting rotations in the coarse space, and the GDSW preconditioner with the full coarse space. These results have been recalled and extended by the hybrid version of the preconditioner and by results using a smooth ramp at the inflow in [107]. In addition to that, in [106], we showed strong parallel scalability results, and we discussed that the shape of the geometry can have a significant influence on the scalability of the FSI solver.

Here, we report on all results for our parallel implementation of GDSW used as a preconditioner for the structural block in FSI from [105], [106], and [107]: a comparison of different overlapping Schwarz preconditioners for the structural block in FSI simulations with different settings, cf. Section 3.2, and strong scalability studies of FSI simulations using GDSW as the preconditioner for the structural block, cf. Section 3.3. The settings for the simulations are described in Section 3.1, and a conclusion is given in Section 3.4.

The computations in this chapter have been performed the JUQUEEN BG/Q supercomputer [191] at JSC Jülich and on the Cray XT6m at Universität Duisburg-Essen. On the JUQUEEN supercomputer, we use the `clang` 4.7.2 compiler and the Engineering and Scientific Subroutine Library (`ESSL`) 5.1. A node of the JUQUEEN supercomputer has 16 cores (Power BQC, 1.60 Ghz) and 16 GB of RAM. On the Cray XT6m supercomputer, we use the `Intel compiler` 11.1 and the Cray Scientific Library (`libsci`) 10.4.4. A node of the Cray XT6m supercomputer has 24 GB of RAM and two sockets, each with 12 cores (Opteron 6168, 1.9 GHz).



Mesh #1:	Interior radius of the structure	0.15 cm
	Outer radius of the structure	0.21 cm
	Length	2.5 cm
Mesh #2:	Interior radius of the structure	0.08 cm
	Outer radius of the structure	0.1 cm
	Length	5 cm
Mesh #3:	Interior radius of the structure	0.08 cm
	Outer radius of the structure	0.11 cm
	Length	10 cm

Figure 3.1: Geometry of the FSI problem. The number of degrees of freedom is almost identical for all geometries and well-balanced between fluid (F) and structure (S); cf. Table 3.1.

3.1 Simulation Settings

We consider the FSI problem, as described in Section 1.1. In contrast to [23, 24] and Section 1.5, where a monolithic Convective Explicit (CE) time discretization scheme was used, a monolithic fully implicit (FI) scheme is used here; see [13, 56, 65] and Section 1.1.2. For the spatial discretization, we use P2-P1 elements for the fluid, P2 elements for the structure, and P2 elements for the geometry problem. This corresponds to the “P2” discretization used in [24] and Chapter 1. The fluid and the structural meshes are, again, conforming on the FSI interface.

We solve the linearized systems using a GMRES iteration with the FaCSI preconditioner [65], which is based on a factorization of the Dirichlet-Neumann preconditioner matrix P_{DN}; cf. Section 1.1.3. The fluid block is treated further by static condensation of the interface degrees of freedom and the use of a SIMPLE [160] preconditioner for the fluid block; see [65] and Section 1.1.3. The inverses appearing in the application of the FaCSI preconditioner are then replaced by overlapping Schwarz preconditioners for geometry, fluid, and structure, separately. The default preconditioner for all blocks is `IFPACK`. The systems from `LifeV` use block coordinate numbering, i.e., all x variables first. Our parallel preconditioner has two potential advantages over `IFPACK`: it uses a geometric overlap and it can use a coarse space for better robustness and improved numerical scalability. For the strong scalability tests, we consider three different meshes with different geometries; cf. Figure 3.1 and Table 3.1, whereas we only use Mesh #1 for the comparison of the preconditioners for the structure.

We apply zero-displacement Dirichlet boundary conditions to the structure at the inlet and the outlet as wells as an inflow boundary condition to the fluid. In particular, we use two different types of inflow conditions, i.e., a pressure

Mesh	Velocity (F)	Pressure (F)	Displacement (S)	Displacement (G)
#1	393 903	17 261	379 080	393 903
#2	401 763	17 775	373 032	401 763
#3	376 623	17 352	346 320	376 623

Table 3.1: Number of degrees of freedom of the discretization of the tube in Figure 3.1.

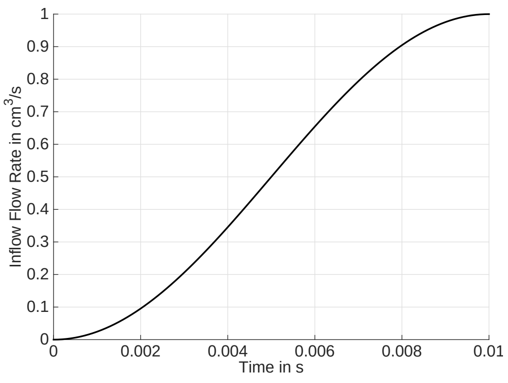


Figure 3.2: Cosine-type inflow boundary condition.

wave and a cosine-type ramp inflow boundary condition. The pressure wave results from a constant normal stress $\sigma \cdot n = 1.33\,\text{kPa}$, which is applied to the fluid inflow for $t \leq 0.003\,\text{s}$. In the cosine-type inflow boundary condition, a parabolic inflow velocity profile is prescribed such that the inflow flow rate Q is given by

$$Q(t) = \frac{Q_{STEADY}}{2}\left(1 - \cos\left(\frac{\pi}{T}t\right)\right) \tag{3.1}$$

for $0 \leq t \leq T$; the shape of the profile is also shown in Figure 3.2. Here, we use $T = 0.01\,\text{s}$; cf. [105, 106, 107].

We use three different material models for the arterial wall, i.e., a linear elastic (cf. Section 1.2.2), a Neo-Hookean (cf. Section 1.2.3), and a sophisticated, anisotropic material model [26], which was denoted as Ψ_A in [41] (cf. Section 1.2.4).

The hyperelastic energy Ψ_A has the form

$$\Psi_A = c_1\left(\frac{I_1}{I_3^{1/3}} - 3\right) + \sum_{a=1}^{2} \alpha_1 \left\langle I_1 J_4^{(a)} - J_5^{(a)} - 2\right\rangle^{\alpha_2} + \varepsilon_1\left(I_3^{\varepsilon_2} + \frac{1}{I_3^{\varepsilon_2}} - 2\right),$$

where $I_1 = \operatorname{tr}\boldsymbol{C}$, $I_3 = \det\boldsymbol{C}$, $J_4^{(a)} = \operatorname{tr}[\boldsymbol{C}M]$, $J_5^{(a)} = \operatorname{tr}[\boldsymbol{C}^2 M]$, and $\boldsymbol{C} := \boldsymbol{F}^T\boldsymbol{F}$; $\boldsymbol{F} := \nabla\boldsymbol{\varphi}$; $\boldsymbol{M} := a \otimes a$ (structural tensor). It has already been used to model arterial walls in FSI; see, e.g., [23, 24] and Chapter 1. For linear elasticity, we use

c_1 [kPa]	ε_1 [kPa]	ε_2	α_1 [kPa]	α_2
17.5	499.8	2.4	30 001.9	5.1

Table 3.2: Parameters for the nonlinear Ψ_A material model used

$E = 400\,\text{kPa}$ and $\nu = 0.3$, for Neo-Hooke, $\mu = 77.2\,\text{kPa}$ and $\kappa = 3833\,\text{kPa}$, and for the Ψ_A model, we use the parameters from [41, 20, (Ψ_A Set 2)], cf. Table 3.2.

Our stopping criterion for Newton is a mixed criterion with a relative and absolute tolerance of 10^{-8}, i.e., the Newton iteration is stopped when $\min\{\|r_n\|_\infty, \|r_n\|_\infty/\|r_0\|_\infty\} < 10^{-8}$, and for GMRES, we use a relative tolerance of 10^{-6}, i.e., the GMRES iteration is stopped when $\|r_n\|_2/\|r_0\|_2 < 10^{-6}$. With $\|\cdot\|_\infty$ and $\|\cdot\|_2$ we refer to the corresponding vector norms, and r_n denotes the residual in the correspond n-th iteration step.

3.2 Comparison of Preconditioners for the Structural Block

In this section, we discuss the performance of the GDSW preconditioner as a preconditioner for the structural block in the monolithic system in FSI; see Section 1.1.

The inverses appearing in the application of the FaCSI preconditioner are replaced by overlapping Schwarz preconditioners; see Section 1.1.3. We use algebraic one-level overlapping Schwarz preconditioners (`IFPACK`) for the geometry block and for the fluid block. For the structure, we consider different preconditioners and compare the resulting performance of the complete monolithic FSI simulation. In particular, we compare the performance using our default preconditioner for the structural block in [23, 24], i.e., `IFPACK`, a geometric one-level overlapping Schwarz preconditioner (OS1), the GDSW preconditioner neglecting rotations (GDSW-nr), the GDSW with full coarse space (GDSW), and the hybrid version of the GDSW preconditioner (GDSW-B). Note that we use the naive implementation of the hybrid version here; cf. Section 2.2. For `IFPACK` as well as for our overlapping Schwarz methods, the local subdomain problems are solved using `UMFPACK`. We perform the comparison using 128 cores of a Cray XT6m. On the Cray (Opteron 6168, 12 cores, 1.9 GHz) architecture, the performance of `MUMPS` and `UMFPACK` is often similar, especially for small matrices. In this section, the subdomain problems have only a few thousand degrees of freedom. We specify an overlap of $\delta = 2h$ for all overlapping Schwarz methods. As inflow conditions, we consider the pressure wave as well as the cosine-shaped ramp for all three (linear and nonlinear) material models introduced in Section 3.1.

3.2.1 Time to Solution - Pressure Wave Inflow Condition

Using the pressure wave inflow condition, we consider the time steps Δt = 0.0001s, 0.0002 s, 0.0004 s, and 0.0005 s, i.e., we solve 100, 50, 25 or 20 monolithic nonlinear systems; see also Figure 3.3. The average number of Newton iterations needed to solve the nonlinear problems for the different combinations of material model and time step size are listed in Table 3.6. The corresponding number of iterations and computing times are presented in Table 3.3 and Figure 3.4.

In Table 3.3, for a small time step, all preconditioners show a very similar performance with respect to the number of GMRES iteration as well as the timings. However, for a larger time step, where the weight in front of the mass

Δt	Mat.	IFPACK		One-level Schwarz (OS1)		GDSW without rot. (GDSW-nr)		GDSW		Hybrid-GDSW (GDSW-B)	
		Time	GMRES its.	Time	GMRES its.	Time	GMRES its.	Time	GMRES its.	Time	GMRES its.
0.0001 s	LE	**5.0 min**	53.4	5.1 min	50.8	5.4 min	51.8	5.3 min	**50.8**	5.6 min	54.1
	NH	8.6 min	89.8	**6.8 min**	59.3	7.1 min	55.3	7.0 min	**52.7**	7.6 min	57.7
	Ψ_A	19.7 min	214.7	**9.9 min**	82.0	10.5 min	81.0	10.6 min	**79.1**	11.1 min	84.9
0.0002 s	LE	8.9 min	95.8	7.8 min	74.5	7.0 min	60.7	**6.8 min**	**58.0**	7.3 min	63.2
	NH	14.2 min	152.4	9.8 min	87.5	9.6 min	77.2	**9.0 min**	**66.0**	9.4 min	68.2
	Ψ_A	33.3 min	316.7	**13.2 min**	96.9	13.8 min	94.1	13.9 min	**90.7**	14.8 min	99.6
0.0004 s	LE	15.3 min	147.2	14.1 min	124.5	10.9 min	84.4	**9.6 min**	**71.9**	10.3 min	77.3
	NH	24.7 min	226.5	17.8 min	145.7	16.2 min	117.9	13.6 min	88.4	**13.5 min**	**86.1**
	Ψ_A	63.0 min	399.9	27.0 min	145.4	27.1 min	135.5	**23.5 min**	**108.5**	24.3 min	113.2
0.0005 s	LE	19.4 min	169.0	17.7 min	142.0	13.0 min	93.7	**11.3 min**	**76.3**	11.9 min	79.9
	NH	33.5 min	261.5	24.2 min	171.0	20.9 min	133.2	17.1 min	96.1	**16.7 min**	**90.4**

Table 3.3: Average computing time per time step (in minutes) on the Cray XT6m and average number of GMRES iterations per Newton step for the *pressure wave in a tube* problem discretized on Mesh#1; see Figure 3.4 for the total runtimes. Linear elasticity (LE), Neo-Hooke (NH), and a nonlinear, anisotropic hyperelastic material law (Ψ_A) to model the arterial wall; see also Figure 3.3. The time step is Δt and the final simulation time is $T = 0.01$ s. We compare `IFPACK` with the one-level overlapping Schwarz preconditioner (OS1), the GDSW preconditioner with and without rotations (GDSW/GDSW-nr), and the hybrid version of the preconditioner (GDSW-B) on 128 cores of a Cray XT6m. No convergence for Ψ_A and $\Delta t = 0.0005$ s. Best numbers in **bold face**; cf. [106].

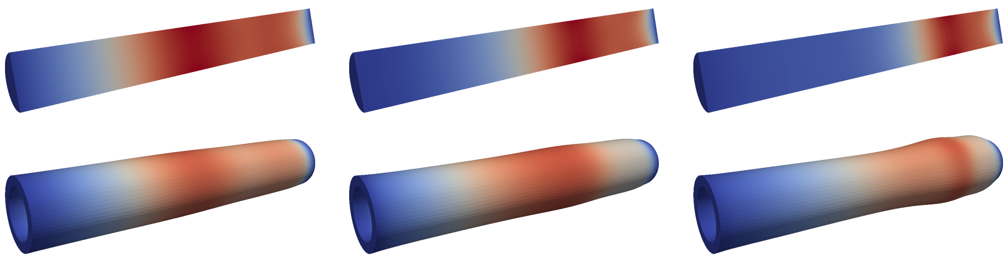


Figure 3.3: Fluid pressure (top) and structural deformation (bottom) for the linear elastic (left), the Neo-Hookean (middle), and the Ψ_A (right) material model at $t = 0.003\,\text{s}$. The structural displacement is magnified by a factor of 10. The figure also illustrates the significantly different behavior for the material models; cf. [106].

Δt	Mat.	Newton its.	
		Pressure Wave	Cosine Ramp
0.0001s	LE	5.1	3.9
	NH	5.6	3.9
	Ψ_A	6.6	3.9
0.0002s	LE	6.1	4.2
	NH	6.3	4.1
	Ψ_A	7.9	4.1
0.0004s	LE	7.4	-
	NH	7.9	-
	Ψ_A	11.9	-
0.0005s	LE	8.4	5.0
	NH	9.5	5.0
	Ψ_A	-	5.1

Table 3.4: Average number of Newton iterations per time step for FSI for the pressure wave and the cosine ramp inflow boundary condition in the tube. Linear elasticity (LE), Neo-Hooke (NH), and a nonlinear, anisotropic hyperelastic material law (Ψ_A); cf. [106, 107].

matrix is small, the number of iterations and the timings for `IFPACK` quickly deteriorate. The other methods, which use a geometric overlap, show a better performance. The use of a coarse space gives further improvements: for the two largest time steps the GDSW preconditioner is the fastest method.

We observe that, when using nonlinear material models, the methods with geometric overlap perform much better than `IFPACK`. On the other hand, when increasing the time step size, a second level is needed to obtain the best performance.

In particular, we observe that for linear elasticity and the smallest time step all preconditioners show a comparable performance. On the contrary, for a large time step $\Delta t = 0.0004\,\text{s}$ and the highly nonlinear material model (Ψ_A), the GDSW preconditioner, in the standard as well as in the hybrid version, is

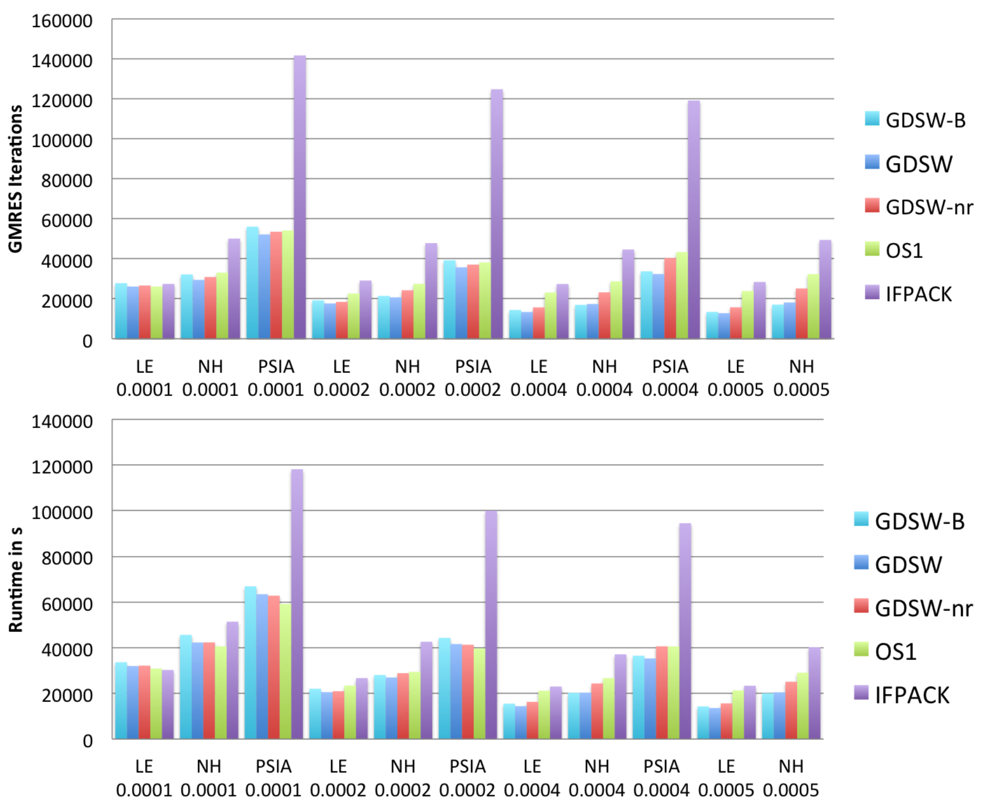


Figure 3.4: Total number of GMRES iterations (top) and total runtime on the Cray XT6m (bottom) for the *pressure wave in a tube* FSI problem for different time step sizes using Mesh #1 and 128 cores; see also Table 3.3. We use different preconditioners for the structure block. "OS1" is the one-level Schwarz preconditioner, "GDSW-nr" is the GDSW preconditioner without rotations, "GDSW" is the GDSW preconditioner with full coarse space, and "GDSW-B" is the hybrid version of the GDSW preconditioner. Linear elasticity (LE), Neo-Hooke (NH), and a nonlinear, anisotropic hyperelastic material law to model the arterial wall (Ψ_A/PSIA); cf. [106].

more than 2.5 times faster than `IFPACK`. For the Neo-Hookean material model and large time steps ($\Delta t = 0.0004\,\mathrm{s}$ and $\Delta t = 0.0005\,\mathrm{s}$), where the improvement through the second level (GDSW vs. OS1) is most noticeable, we observe the best performance for the hybrid version of the preconditioner. For smaller time steps, where the coarse level is less beneficial, the performance of the hybrid preconditioner (here, in its naive implementation) is worse.

In Figure 3.5, the variation of the computing time over the simulation time for the time step $\Delta t = 0.0005\,\mathrm{s}$ and the Neo-Hookean material, is depicted.

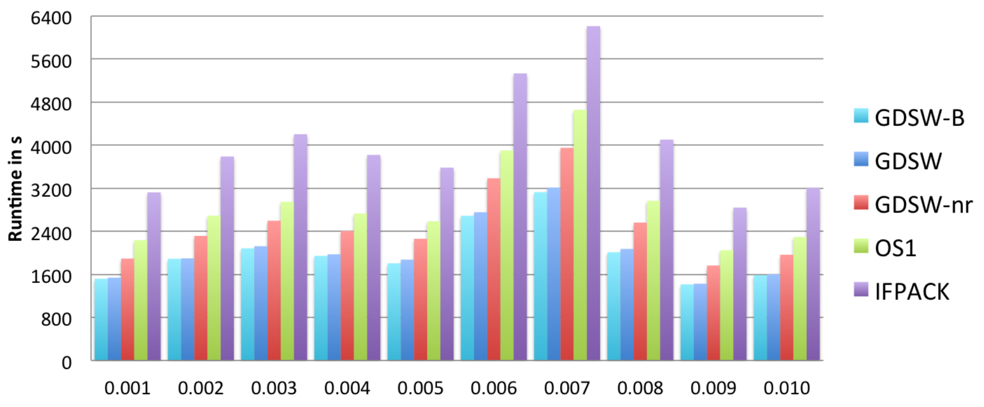


Figure 3.5: Runtimes for the monolithic FSI simulation on the Cray XT6m using a Neo-Hookean material, a time step of 0.0005 s, and a pressure wave inflow condition. For clarity, the runtimes of two subsequent time steps of size $\Delta t = 0.0005\,\mathrm{s}$ are combined. All bars belonging to one preconditioner sum up to the corresponding total runtime; "OS1" is the one-level Schwarz preconditioner, "GDSW-nr" is the GDSW preconditioner without rotations, and "GDSW" is the GDSW preconditioner with full coarse space; cf. [105].

The variation over the simulation time, as a result of the propagation of the pressure wave in the tube, is qualitatively the same for all methods. In this case, GDSW and the hybrid GDSW are almost twice as fast as `IFPACK`.

In contrast to the results for unstructured domain decompositions in Section 2.5.4.2, here, the improvement from a reduction of the coarse space (GDSW-nr) does not compensate the increase in the number of iterations. The timings using GDSW-nr are mostly in between the timings for OS1 and GDSW.

3.2.2 Time to Solution - Cosine Ramp Inflow Condition

In Figure 3.7 and Table 3.5, we present the corresponding results for the cosine-type inflow boundary condition. A similar inflow condition is used in Chapter 1 and in [24] in order to prestress the artery before the simulation of heartbeats. For this settings, we use time steps $\Delta t = 0.0001\,\mathrm{s}$, $0.0002\,\mathrm{s}$, and $0.0005\,\mathrm{s}$. As can be observed in Table 3.6, the nonlinear problems in each time step are easier to solve than for the pressure wave problem, cf. Section 3.2.1. Also, when increasing the time step size, the number of Newton iterations grows only slightly, in contrast to the observations for the pressure wave. Nevertheless, we observe the same qualitative behavior with respect to the different preconditioners.

Δt	Mat.	IFPACK		One-level Schwarz (OS1)		GDSW without rot. (GDSW-nr)		GDSW		Hybrid-GDSW (GDSW-B)	
		Time	GMRES its.	Time	GMRES its.	Time	GMRES its.	Time	GMRES its.	Time	GMRES its.
0.0001 s	LE	**3.9 min**	53.7	**3.9 min**	51.4	4.0 min	52.0	4.0 min	**51.0**	4.3 min	54.1
	NH	6.0 min	89.1	**4.8 min**	59.5	5.0 min	55.6	5.0 min	**52.6**	5.3 min	57.7
	Ψ_A	11.6 min	211.1	**6.1 min**	85	6.3 min	82.6	6.4 min	**79.0**	6.6 min	83.8
0.0002 s	LE	6.2 min	96.9	5.5 min	76.4	4.9 min	61.8	**4.8 min**	**58.9**	5.1 min	64.2
	NH	9.4 min	153.1	6.6 min	89.6	6.4 min	77.5	**6.0 min**	**66.1**	6.1 min	67.6
	Ψ_A	17.1 min	313.3	**6.9 min**	97.3	7.2 min	94.4	7.2 min	**89.7**	7.7 min	98.7
0.0005 s	LE	11.3 min	163.1	10.7 min	143.2	**7.0 min**	94.4	**7.0 min**	**79.6**	7.5 min	84.6
	NH	17.4 min	256.9	12.7 min	168.8	10.9 min	130.7	9.0 min	95.4	**8.8 min**	**90.7**
	Ψ_A	26.7 min	400	12.3 min	160.7	11.8 min	142.9	**9.9 min**	**110.1**	10.1 min	111.9

Table 3.5: Average computing time per time step (in minutes) on the Cray XT6m and average number of GMRES iterations per Newton step for the *cosine ramp in a tube* problem discretized on Mesh#1; see Figure 3.4 for the total runtimes. Linear elasticity (LE), Neo-Hooke (NH), and a nonlinear, anisotropic hyperelastic material law (Ψ_A) to model the arterial wall; see also Figure 3.3. The time step is Δt and the final simulation time is $T = 0.01$ s. We compare `IFPACK` with the one-level overlapping Schwarz preconditioner (OS1), the GDSW preconditioner with and without rotations (GDSW/GDSW-nr), and the hybrid version of the preconditioner (GDSW-B) on 128 cores of a Cray XT6m. Best numbers in **bold face**; cf. [107]

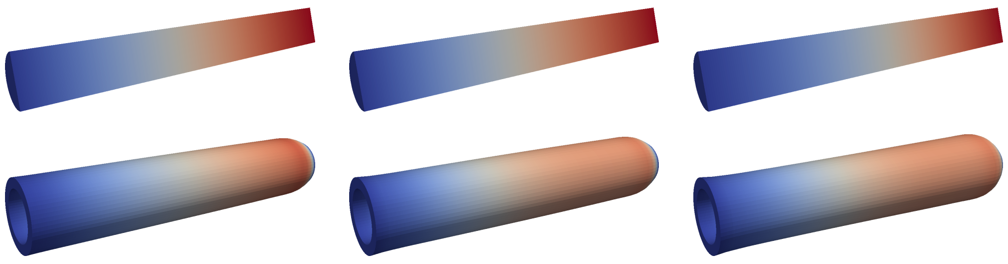


Figure 3.6: Fluid pressure (top) and structural deformation (bottom) for the linear elastic (left), the Neo-Hookean (middle), and the anisotropic Ψ_A (right) material model at $t = 0.008\,\mathrm{s}$ using a cosine ramp inflow condition. The structural displacement is magnified by a factor of 40. The figure also illustrates the significantly different behavior for the material models; cf. [106].

Again, a geometric overlap improves the iterative solution of the linearized monolithic systems where nonlinear material laws have been used in the models. This can also be observed in Figure 3.8, where the computation times for single time steps are plotted for the Ψ_A material model and a time step of $0.0002\,\mathrm{s}$. To improve the performance with respect to larger time steps, the second level of the GDSW preconditioner becomes significant; see Figure 3.7 and Table 3.5.

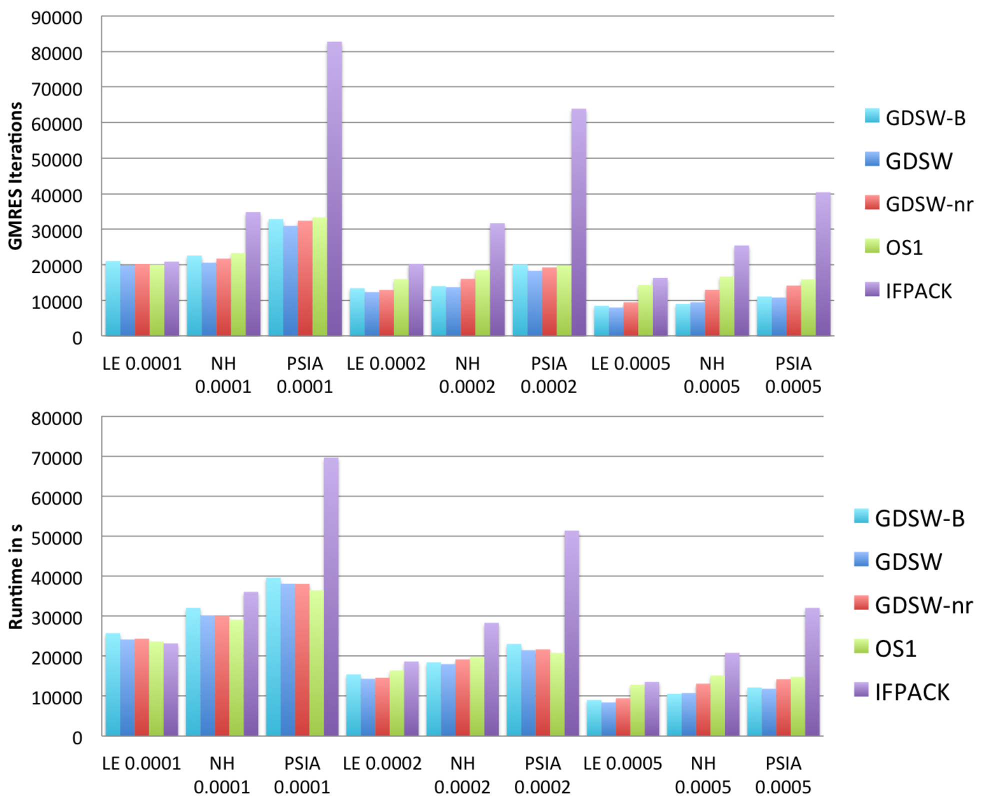


Figure 3.7: Total number of GMRES iterations (top) and total runtime on the Cray XT6m (bottom) for the *cosine ramp in a tube* FSI problem for different time step sizes using Mesh#1 and 128 cores; see also Table 3.5. We use different preconditioners for the structure block. "OS1" is the one-level Schwarz preconditioner, "GDSW-nr" is the GDSW preconditioner without rotations, "GDSW" is the GDSW preconditioner with full coarse space, and "GDSW-B" is the hybrid version of the GDSW preconditioner. Linear elasticity (LE), Neo-Hooke (NH), and a nonlinear, anisotropic hyperelastic material law to model the arterial wall (Ψ_A/PSIA); cf. [107].

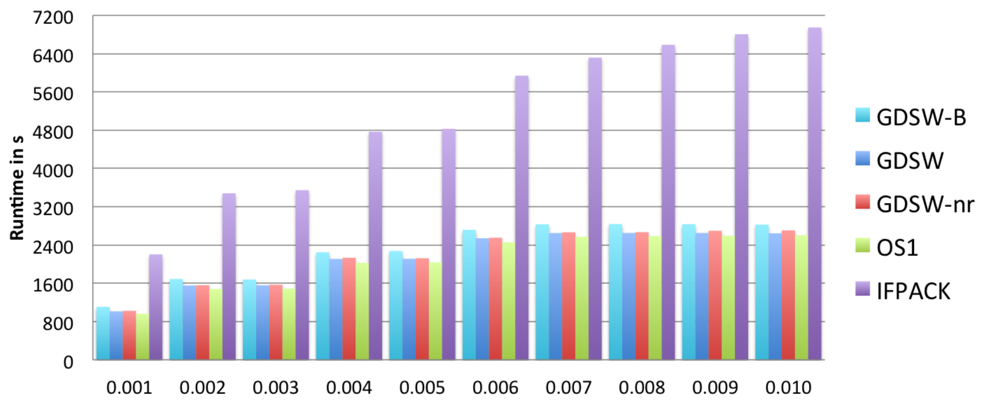


Figure 3.8: Runtimes for the monolithic FSI simulation on the Cray XT6m using the Ψ_A material model, a time step of 0.0002 s, and a cosine-type ramp inflow condition. For clarity, the runtimes of five subsequent time steps of size $\Delta t = 0.0002\,\mathrm{s}$ are combined. All bars belonging to one preconditioner sum up to the corresponding total runtime; "OS1" is the one-level Schwarz preconditioner, "GDSW-nr" is the GDSW preconditioner without rotations, and "GDSW" is the GDSW preconditioner with full coarse space; cf. [107].

Mesh \ number of processor cores	16	32	64	128	256	512
Mesh #1, overlap $1h$	3	4	4	4	5	4
Mesh #1, overlap $2h$	3	4	4	4	4	4
Mesh #2, overlap $1h$	4	4	4	4	4	4
Mesh #2, overlap $2h$	4	4	4	4	4	4
Mesh #3, overlap $1h$	3	3	4	4	4	4
Mesh #3, overlap $2h$	3	3	4	4	4	4

Table 3.6: Numbers of Newton steps for the strong scaling results shown in Figure 3.10.

3.3 Strong Scaling for the Fluid-Structure Interaction Problem

In Figures 3.9 and 3.10, we present strong parallel scaling results for the first time step for the *pressure wave in a tube* problem using time steps of size $\Delta t = 0.0001\,\text{s}$ and $\Delta t = 0.0002\,\text{s}$, respectively, for a linear elastic tube. For the structure, we use the GDSW preconditioner including rotations with overlaps of $\delta = 1h$ and $\delta = 2h$. For the fluid and the geometry blocks, we again use the `IFPACK` preconditioner with an overlap of $\delta = 2h$. We present the GMRES iterations per Newton step and the total runtime for one time step. The timings are for the first time step of the fully coupled FSI simulation.

For all cases, we observe good scalability results, with slightly worse scaling for a time step of 0.0002 s. This is partially a result of the number of Newton iterations, which varies from three to five. We also observe a significant influence of the shape of the geometry on the performance of the FSI solver. For Mesh #3,

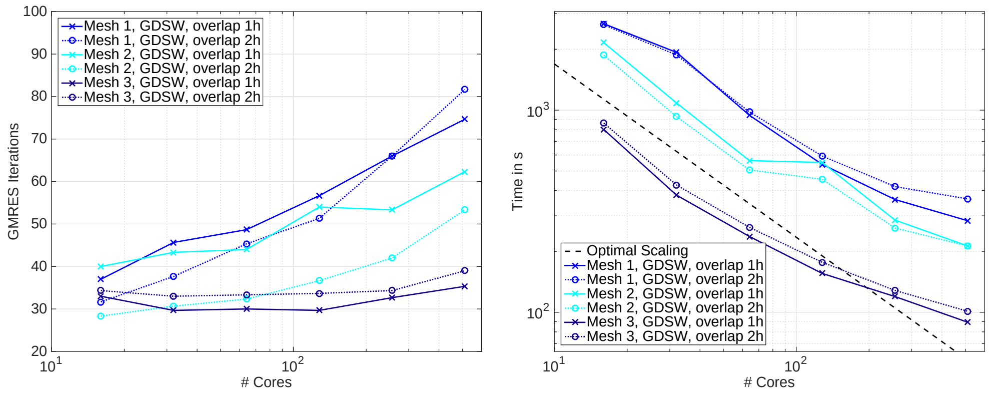


Figure 3.9: Strong parallel scalability on JUQUEEN (16 to 512 cores) for FSI using linear elasticity and $\Delta t = 0.0001\,\text{s}$. The computing time for the first time step is shown. Always 3 Newton steps; cf. [106].

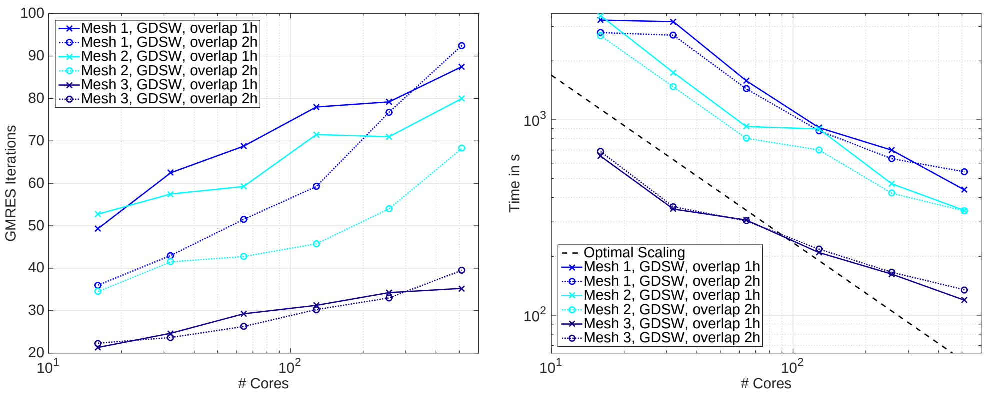


Figure 3.10: Strong parallel scalability on JUQUEEN (16 to 512 cores) for FSI using linear elasticity and $\Delta t = 0.0002\,\mathrm{s}$. The computing time for the first time step is shown. The numbers of Newton steps are shown in Table 3.6; cf. [106].

we observe the lowest number of iterations, the best numerical scalability, the lowest computing times, and the best parallel scalability.

3.4 Conclusion

By applying our GDSW preconditioner, we are able to improve the performance of the FSI simulation by more than a factor of two, compared to the use of `IFPACK`. This is especially remarkable since, in our monolithic preconditioner, we only exchange the preconditioner for the structural block, whereas the timings are for the complete FSI simulation.

From our results, the use of the GDSW preconditioner with the full coarse space can be recommended as the new default preconditioner for our FSI environment, especially when sophisticated nonlinear material models are used to describe the structure appropriately, e.g., in hemodynamics; cf. Chapter 1 and [24]. Thus, using the GDSW preconditioner, the simulation time of FSI simulations of several heartbeats could be reduced significantly since highly nonlinear material models (e.g., the Ψ_A model) are used. The deformations depicted in Figure 3.3 illustrate the significantly different behavior of the sophisticated (anisotropic, almost incompressible, polyconvex, nonlinear, hyperelastic) material model Ψ_A compared to more standard Neo-Hookean hyperelasticity or linear elasticity, i.e., the deformation is significantly more localized for Ψ_A. This is an interesting result by itself.

Hereby, we conclude our considerations of FSI problems, and focus on the discretization and preconditioning (using two-level Schwarz preconditioners) of heterogeneous problems in Chapters 4 and 5, respectively.

4 A Special Finite Element Method Based On Approximate Component Mode Synthesis

We consider problems of the form

$$\begin{aligned} -\nabla \cdot (A(x)\nabla u(x)) &= f(x) \quad \text{in } \Omega \subset \mathbb{R}^2, \\ u &= 0 \qquad\quad \text{on } \partial\Omega, \end{aligned} \tag{4.1}$$

where the coefficient matrix A is rough or highly varying on a small scale. Such problems are also often referred to as multiscale problems. Multiscale problems are challenging to solve with standard finite element methods since very fine meshes are needed in order to resolve the features of the solution on the fine scale. The large number of degrees of freedom then leads to high demands with respect to memory and computational resources. One way to overcome these issues is to introduce methods which explicitly take into account the information on the small scale without resorting to a brute force discretization. By brute force discretization we refer to a very fine discretization using standard finite elements. The incorporation of information on the small scale can be achieved, e.g., by including the coefficient information into the basis functions. Various approaches have been proposed in this field, including multiscale finite element (MsFEM) [77, 114], mixed multiscale finite element [7], heterogeneous multiscale finite element [76], adaptive multiscale [159], generalized finite element [8, 9, 10], and Component Mode Synthesis (CMS) [54, 117, 118] methods.

The special finite element method considered in this chapter, which is based on [103], was introduced by Hetmaniuk and Lehoucq in [111]. Additional theory has been proven by Hetmaniuk and Klawonn in [110]. The method is designed as an approximation of the CMS method using three different types of basis functions in order to do so. It combines *bubble-type* eigenmodes, *vertex-specific* energy minimizing extensions of nodal trace functions, and *coupling edge-based* eigenmodes. For a detailed description of the eigenmode problems, see Section 4.1.2, especially Formulae (4.19) and (4.20), as well as Figure 4.2. We refer to these basis functions as ACMS (Approximate Component Mode Synthesis)[1] shape functions or coarse basis functions. An important property of these basis functions, in addition to their approximation properties, is, in contrast to the basis functions used in the CMS method, their local support. The resulting linear system is therefore sparse and the construction of the coarse basis functions parallelizes well. Note that the numerical construction of the ACMS shape functions can be computationally expensive and, thus, parallelization is crucial for the efficiency of the ACMS method. In this chapter, we therefore investigate the computational cost of using an ACMS discretization in a parallel context.

Using the ACMS method, in order to achieve a comparable accuracy, the discretized system can be smaller by one to two orders of magnitude compared to a brute force discretization; see also Section 4.4.2. However, these systems can still be large and ill-conditioned. Hence, we combine the ACMS discretization with a parallel FETI-DP domain decomposition method as iterative solution method. We show that the FETI-DP method applied to ACMS discretizations is numerically scalable and converges in a small number of iterations.

The remainder of this chapter is organized as follows. In Section 4.1, we first describe the ACMS method as it was introduced in Hetmaniuk and Lehoucq [111]. Then, we briefly discuss the approximation properties of eigenfunctions and then provide an error estimate for the ACMS method which was derived in Hetmaniuk and Klawonn [110]. Next, we describe our parallel implementation of the ACMS method and conclude this section with an algorithmic description of the solution of the eigenvalue problems needed in the ACMS approach. In Section 4.2, we first provide a general algorithmic description of the FETI-DP domain decomposition method, first for standard finite elements and then applied to the ACMS special finite element discretization. In Section 4.3, we introduce several different model problems: the Poisson equation and two second-order diffusion problems, one with a slightly varying coefficient matrix and one with a highly oscillating coefficient matrix. We also consider two additional model problems, where one is even more heterogeneous than the ones considered before and the other one has a discontinuous coefficient function with high jumps. Finally, in Section 4.4, we present numerical results using our parallel implementation of the ACMS method which show that the assembly of this special finite element method is parallel scalable. We also provide numerical results which show that the FETI-DP domain decomposition method applied to the ACMS discretization is weakly parallel scalable. The conclusion of this chapter is provided in Section 4.5.

[1] The letter "A" in ACMS stands for "Approximate" and emphasizes the approximation of a CMS technique.

4.1 Approximate Component Mode Synthesis Discretizations

The model problem (4.1) can be transformed into the variational formulation: find $u \in H_0^1(\Omega)$, such that

$$a\,(u,v) = L(v) \qquad \forall v \in H_0^1(\Omega) \tag{4.2}$$

with the bilinear form and the linear functional

$$a\,(u,v) = \int_\Omega (\nabla u(x))^T A(x) \nabla v(x)\,dx \quad \text{and} \quad L\,(v) = \int_\Omega f(x)v(x)\,dx,$$

respectively, where $f \in L^2(\Omega)$. We assume that the matrix A is uniformly symmetric positive definite and that it satisfies

$$0 < \alpha_{min}\xi^T\xi \;\le\; \xi^T A(x)\xi \;\le\; \alpha_{max}\xi^T\xi \qquad \forall x \in \bar{\Omega} \text{ and } \xi \in \mathbb{R}^2 \setminus \{0\}\,,$$

with constants α_{min}, α_{max} independent of x. For the convergence theory developed in [110], it is assumed that the coefficients a_{ij} of the matrix $A = (a_{ij})_{i,j}$ are in $\mathcal{C}^1(\overline{\Omega})$. The ACMS method might be applied with less strict regularity assumptions on A but no convergence estimate is known so far for this case. Although the method can be applied in a more general setting, for the theory in [110], a two-dimensional polygonal domain is assumed.

In order to define the finite element space of our special finite element method, we consider a family $(\tau_h)_h$ of conforming partitions of Ω into triangles or convex quadrilaterals. The elements of the partition are assumed to be open sets, and the intersection of the closure of two distinct elements T and T' is either empty, a vertex, or a complete edge with two vertices. This partition introduces

$$\Gamma = \left(\bigcup_{T \in \tau_h} \partial T \right) \setminus \partial\Omega \tag{4.3}$$

and thus we have

$$\Omega = \left(\bigcup_{T \in \tau_h} T \right) \cup \Gamma. \tag{4.4}$$

Additionally, we define discrete harmonic, i.e., energy-minimizing, extensions of trace functions on Γ. By $W_\Gamma \subset H^{1/2}(\Gamma)$ we denote the subspace of trace functions on Γ of all functions in $H_0^1(\Omega)$. Thus, a discrete harmonic extension

$E_\Omega(\tau)$ of $\tau \in \mathrm{W}_\Gamma$ is characterized either by the minimization problem

$$\inf_{v \in H_0^1(\Omega)} a(v,v) \quad \text{with} \quad v|_\Gamma = \tau \tag{4.5}$$

or equivalently by

$$\begin{aligned} -\nabla \cdot (A(x)\nabla E_\Omega(\tau)) &= 0 \quad \text{in } T,\ \forall T \in \tau_h, \\ E_\Omega(\tau) &= \tau \quad \text{on } \Gamma, \\ E_\Omega(\tau) &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.6}$$

Based on the partition τ_h of Ω, subspaces

$$V_T = \left\{ v \in H_0^1(\Omega) : v|_T \in H_0^1(T) \text{ and } v|_{\Omega \setminus \bar{T}} = 0 \right\} \subset H_0^1(\Omega) \tag{4.7}$$

for all $T \in \tau_h$ and

$$V_\Gamma = \left\{ \mathrm{E}_\Omega(\tau) \in H_0^1(\Omega) : \tau \in \mathrm{W}_\Gamma \right\} \subset H_0^1(\Omega) \tag{4.8}$$

can be introduced, such that

$$H_0^1(\Omega) = \left(\bigoplus_{T \in \tau_h} V_T \right) \oplus V_\Gamma. \tag{4.9}$$

This decomposition is orthogonal with respect to $a(\cdot,\cdot)$ and is standard in domain decomposition theory; see also [110, eq. (2.4)] and the subsequent discussion given there.

Based on this decomposition, problem (4.2) can equivalently be written as: find $u_T \in V_T$ and $u_\Gamma \in V_\Gamma$, such that

$$\begin{aligned} a(u_T, v_T) &= L(v_T) \quad \forall T \in \tau_h, \forall v_T \in V_T, \\ a(u_\Gamma, v_\Gamma) &= L(v_\Gamma) \quad \forall v_\Gamma \in V_\Gamma \end{aligned} \tag{4.10}$$

with

$$u = \sum_{T \in \tau_h} u_T + u_\Gamma \quad \in H_0^1(\Omega). \tag{4.11}$$

It is, of course, important that our special finite element method can be implemented efficiently. Therefore, we choose basis functions with local support. These shape functions are designed as local approximations to the CMS finite element space, which we will describe in the next section.

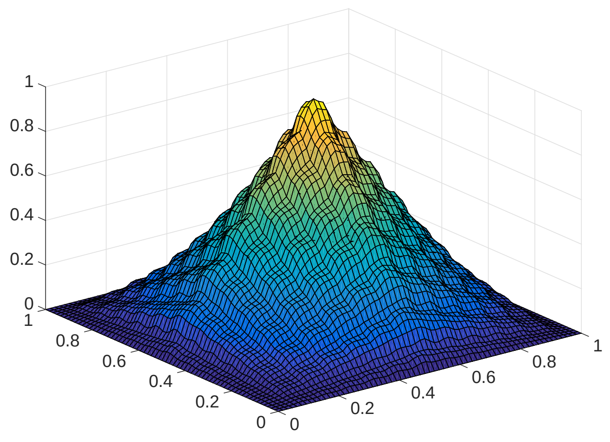


Figure 4.1: A *vertex-specific* coarse basis function with oscillating interface values (see Problem 3) used in the ACMS method. The basis function is defined on a fine mesh of width h_f.

4.1.1 Discretization Spaces Based on Eigenfunctions

The composed problem (4.10) can be solved by using a discretization space based on eigenfunctions. This idea leads to the Component Mode Synthesis (CMS) finite element method which was introduced in [54, 117, 118]. The CMS discretization has very good approximation properties but at high computational costs.

As already known for some time, eigenfunctions have good approximation properties for the solution of variational problems as they are optimal with respect to the so-called n-th width, which has been introduced by Kolmogoroff [141]. This means that they have optimal approximation properties compared to other subspaces of the same dimension; see, e.g., [141, 163, 8] for more details.

Namely, for the variational problem

$$a(u,v) = L(v) \qquad \forall v \in H_0^1(\Omega), \tag{4.12}$$

let z_i be the eigenfunction and λ_i the corresponding eigenvalue of the eigenvalue problem

$$a(z_i,v) = \lambda_i (z_i,v)_{L^2(\Omega)} \qquad \forall v \in H_0^1(\Omega), \tag{4.13}$$

and let those eigenvalues be sorted in non-descending ordering, i.e., $0 < \lambda_1 \leq \lambda_2 \leq ....$ Then, the space span$\{z_1, z_2, ..., z_n\}$ has the best approximation properties among all subspaces of dimension n.

Based on this, the basis functions of the CMS discretization space are obtained by solving the following eigenvalue problems:

$$\begin{aligned} a\left(z_{*,T}, v\right) &= \lambda_{*,T}\left(z_{*,T}, v\right)_{L^2(\Omega)} \quad \forall v \in V_T, \forall T \in \tau_h, \\ a\left(z_{*,\Gamma}, v\right) &= \lambda_{*,\Gamma}\left(z_{*,\Gamma}, v\right)_{L^2(\Omega)} \quad \forall v \in V_\Gamma. \end{aligned} \tag{4.14}$$

Using these eigenfunctions, we assemble the CMS finite element space

$$V_{CMS} = \left(\bigoplus_{T \in \tau_h} \operatorname{span}\left\{ z_{i,T} : 1 \leq i \leq I_T \right\} \right) \oplus \operatorname{span}\left\{ z_{i,\Gamma} : 1 \leq i \leq I_\Gamma \right\} \tag{4.15}$$

with integers $I_T > 0$, $\forall T \in \tau_h$, and $I_\Gamma > 0$.

As can be seen from the studies in [111], the CMS method provides very good approximation properties compared to standard, multiscale or special finite element methods. However, with respect to computational costs, the CMS method is very expensive and it is not clear how to use this method efficiently in a parallel context. This is due to the global support of the interface basis functions.

Therefore, in very recent works of Smetana [184, 185], local approximation spaces for CMS finite element spaces were combined with the Reduced Basis Element (RBE) method in order to overcome this drawback of the CMS method. These local approximation spaces are closely related to the ACMS approach in [111], which is described here.

4.1.2 Description of the ACMS Method

In order to define the finite element space of the special finite element method, different types of basis functions are used, i.e., *vertex-specific*, *edge-based*, and *fixed-interface* shape functions. We refer to the third type of shape functions also as interior *bubble-type* functions.

The *vertex-specific* and the *edge-based* basis functions form an approximation of the subspace

$$\operatorname{span}\left\{ z_{i,\Gamma} : 1 \leq i \leq I_\Gamma \right\} \tag{4.16}$$

of the CMS space (4.15), whereas the *fixed-interface* basis functions are chosen to be exactly the functions $z_{*,T}$ from the CMS Space since they already have local support.

First, we briefly introduce the *vertex-specific* basis functions, which are of *MsFEM basis function* type, cf. [77, 114], and then describe local eigenvalue problems which lead to the two other types of shape functions.

A basis function which corresponds to a vertex P of the partition τ_h is given by the following boundary value problem

$$\begin{aligned} -\nabla \cdot (A(x)\nabla\varphi_P) &= 0 && \text{in } T,\ \forall T \in \tau_h, \\ \varphi_P &= 0 && \text{on } \partial\Omega, \\ \varphi_P &\neq 0 && \text{on } \Gamma, \\ \varphi_P(P') &= \delta_{P,P'} && \text{on } \Gamma, \end{aligned} \tag{4.17}$$

where P' is also a vertex of the partition and $\delta_{P,P'}$ is the *Kronecker delta* function. The MsFEM basis functions are therefore discrete harmonic extensions of trace functions defined on Γ, and thus $\varphi_P \in V_\Gamma$.

It remains to define the trace values of φ_P on Γ, and there are different possibilities to do so. The easiest way is to define φ_P on each edge as a linear function between the two endpoints of the edge, cf. Figure 4.4. The corresponding finite element space is called $V_{\text{ACMS}-\text{L}}$ in [111]. This choice of the trace incorporates the oscillations of the coefficient matrix A with respect to the inner nodes of the elements, i.e., by means of the discrete harmonic extension. However, the oscillations on the edge are ignored.

Instead, the trace can be defined differently, i.e., we require the values on an edge $e \subset \Gamma$ to satisfy

$$\begin{aligned} \frac{\partial}{\partial \tau}\langle A(x)\tau, \nabla\varphi_P(x)\rangle &= 0 && \text{on } e, \\ \varphi_P(P') &= \delta_{P,P'} && \text{on } \Gamma, \end{aligned} \tag{4.18}$$

where τ denotes the tangential vector of the edge with $\|\tau\| = 1$ and $\langle\cdot,\cdot\rangle$ is the standard l_2-inner product. This energy minimal extension onto the adjacent edges can also be seen in Figure 4.1. This leads, together with the eigenmodes defined below, to the finite element space $V_{\text{ACMS}-\text{O}}$ which, for the sake of simplicity, is also just called V_{ACMS}. When the matrix A is a constant multiple of the identity matrix, both spaces are identical. A numerical comparison of these two spaces is presented in [111].

The other two types of basis functions are defined by eigenvalue problems. The so-called *fixed-interface* shape functions are given by: find $(z_{*,T}, \lambda_{*,T}) \in V_T \times \mathbb{R}$ such that

$$a\left(z_{*,T}, v\right) = \lambda_{*,T}\left(z_{*,T}, v\right)_{L^2(\Omega)} \qquad \forall v \in V_T, \tag{4.19}$$

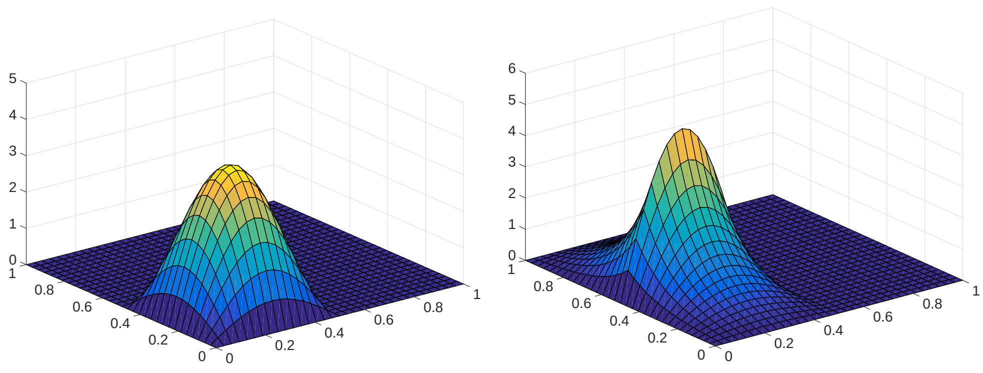


Figure 4.2: Eigenmodes for Problem 1 (2×2 ACMS elements: $1/h_f = 2$, $h/h_f = 16$): a *fixed-interface* basis function (left) and a *coupling* basis function (right).

on each of the $T \in \tau_h$; cf. Figure 4.2 and 4.3. This corresponds to first type of eigenvalue problems in (4.14). For any open edge $e \subset \Gamma$ we define the *edge-based coupling* basis function by the corresponding eigenvalue problem in the space of harmonic extensions: find $(\tilde{\tau}_{*,e}, \lambda_{*,e}) \in H_{00}^{1/2}(e) \times \mathbb{R}$ such that

$$a\left(E_\Omega(\tilde{\tau}_{*,e}), E_\Omega(\tilde{\eta})\right) = \lambda_{*,T}\left(E_\Omega(\tilde{\tau}_{*,e}), E_\Omega(\tilde{\eta})\right)_{L^2(\Omega)} \qquad \forall \eta \in H_{00}^{1/2}(e) \tag{4.20}$$

with $\tilde{\eta}$ being the trivial extension of η by zero to $\Gamma \setminus e$; cf. Figures 4.2 and 4.5. The eigenvalues $\{\lambda_{i,T}\}_{i=1}^{\infty}$ and $\{\lambda_{i,e}\}_{i=1}^{\infty}$ are assumed to be ordered nondecreasingly, and the corresponding eigenmodes accordingly. The eigenmodes form orthonormal bases for the L^2-inner product of V_T and of V_Γ on the element T and on the edge e, respectively.

The finite element space of the special finite element method is then given by

$$\begin{aligned} V_{\text{ACMS}} = &\left(\bigoplus_{T \in \tau_h} \operatorname{span}\{z_{i,T} : 1 \le i \le I_T\}\right) \\ &\oplus \left(\bigoplus_{P \in \Omega} \operatorname{span}\{\varphi_P\}\right) \oplus \left(\bigoplus_{e \subset \Gamma} \operatorname{span}\{E_\Omega(\tilde{\tau}_{i,e}) : 1 \le i \le I_e\}\right) \end{aligned} \tag{4.21}$$

with positive integers I_T, $\forall T \in \tau_h$, and I_e corresponding to the number of eigenmodes used as basis functions. Note, that the Dirichlet boundary conditions are naturally built into the V_{ACMS} space.

4.1.3 Error Estimate

Recently, an a priori error estimate for this special finite element method has been given by Hetmaniuk and Klawonn in [110, Prop. 3.4]; see also [110, Prop. 3.6] for an a posteriori error indicator.

Under the assumption that the coefficients a_{ij} of $A = (a_{ij})$ are in $\mathcal{C}^1(\bar{\Omega})$ and that the solution u of (4.2) belongs to $H_0^1(\Omega) \cap H^{s_0}(\Omega)$, with $s_0 > \frac{3}{2}$, the error between the solution u and the approximate solution $u_{\mathrm{ACMS}} \in V_{\mathrm{ACMS}}$ satisfies

$$a\left(u - u_{ACMS}, u - u_{ACMS}\right) \leq \sum_{T \in \tau_h} \frac{\|f\|^2_{L^2(T)}}{\lambda_{I_T,T}} + C_{s_0,\sigma,A} h^{2s_0-3} \sum_{T \in \tau_h} \frac{\|u\|^2_{H^{s_0}(T)}}{\min_{e \subset \partial T \cap \Gamma} \lambda_{I_e,e}},$$

where the constant $C_{s_0,\sigma,A}$ does not depend on u and h. For further details, see [110, Prop. 3.4] and the related proof.

4.1.4 Parallel Implementation of the ACMS Discretization

Hetmaniuk and Lehoucq [111, Section 5.1] explain how to numerically compute the ACMS basis functions. In this section, we briefly describe the parallel computation of these coarse basis functions on a refined nested mesh, with mesh size $h_f < h$, using bilinear (Q1) Lagrangian finite elements. Only the fine mesh size h_f is chosen small enough such that the important features of the partial differential equation are resolved. Our special finite element method on the mesh of size h then uses the basis functions constructed above.

For the implementation of the algorithm, the library PETSc 3.2-p7 [15, 17] and MPI are used. Particularly, we make use of the matrix, vector, and solver structures which are provided therein. The discrete harmonic extensions occurring in the ACMS method are computed using the sparse Cholesky decomposition implemented in PETSc.

The first step in the construction of the ACMS system is the assembly of the local Q1 fine elements. This step is local to a processor core and can be performed in parallel without communication: the local stiffness-matrices and right-hand sides are built, and it is sufficient to store them locally. The same is valid for the local mass matrices which have to be assembled in order to be used within the generalized eigenvalue problems for the computation of the eigenmodes; see Section 4.1.2. We conclude that this part of the implementation needs no communication, and thus we expect it to be perfectly scalable.

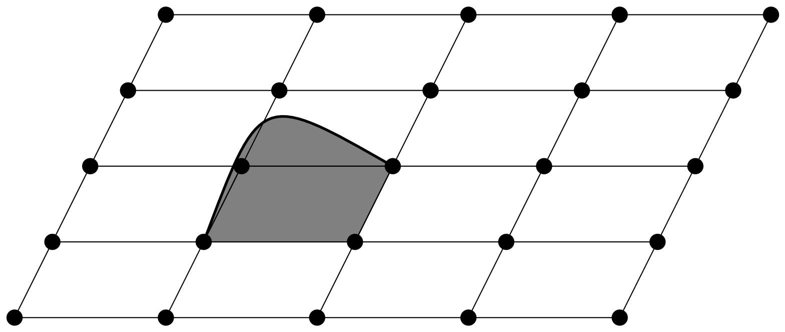


Figure 4.3: Support and shape of a *fixed-interface* basis function (first eigenmode) on a rectangular partition. A concrete such fixed-interface basis function is depicted in Figure 4.2.

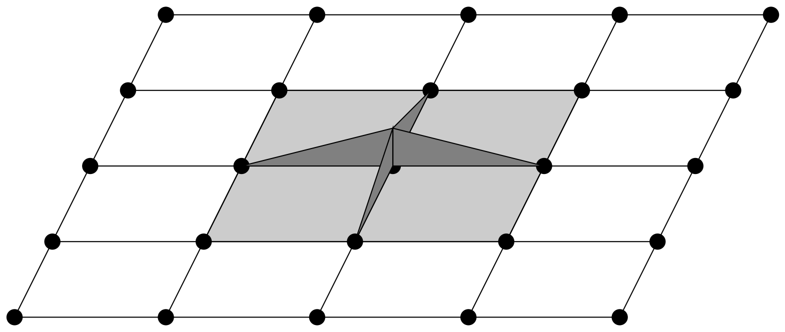


Figure 4.4: Support and trace of a *vertex-specific* basis function on a rectangular partition. A concrete such vertex-specific basis function is depicted in Figure 4.1.

Second, we construct the basis functions for the ACMS elements. Basis functions with local support are used, i.e., the support only contains a bounded number of coarse elements: the *fixed-interface* basis functions $z_{i,K}$ have non-zero values only on one coarse element, cf. Figure 4.3. The computation of these *bubble-type* basis functions is independent of other coarse elements and can therefore be performed locally without any communication. Thus, this part is scalable as well. Although the *vertex-specific* basis functions φ_P are non-zero on several coarse elements (see, e.g., Figure 4.4 for a rectangular coarse mesh) the values can be computed separately and in parallel on each of those elements. For another parallel implementation of the *vertex-specific* basis functions being the basis functions of the MsFEM, see [31].

The next step involves nearest neighbor communication if an edge is shared by two different processes. The support of *edge-based coupling* basis functions $\tau_{i,e}$ consists of two coarse elements. If both adjoint coarse elements reside on the same process the computation does not require any communication. Otherwise

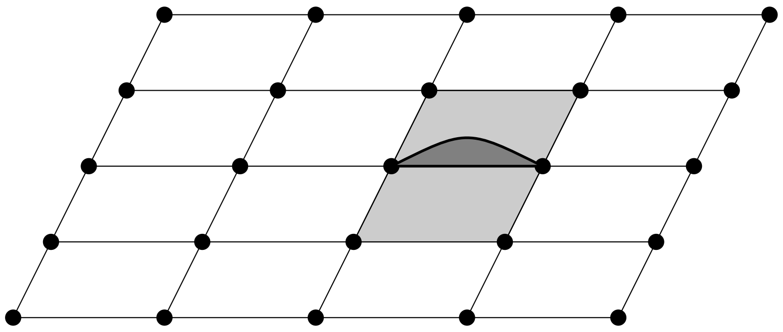


Figure 4.5: Support and trace of a *edge-based* basis function (first eigenmode) on a rectangular partition. A concrete such edge-based basis function is depicted in Figure 4.2.

submatrices corresponding to the generalized eigenvalue problem have to be communicated.

We consider two neighboring subdomains Ω_1 and Ω_2 and two ACMS elements $T_1 \subset \Omega_1$ and $T_2 \subset \Omega_2$ which share an ACMS element edge $e \subset \partial\Omega_1 \cap \partial\Omega_2$. We denote by K_{11} and K_{22} the corresponding local stiffness matrices of the interior degrees of freedom of the ACMS elements T_1 and T_2, respectively, and analogously, by M_{11} and M_{22} the corresponding local mass matrices. By K_{ie} and $M_{ie}, i = 1, 2$, we denote the local stiffness and mass matrices that are formed from basis functions in T_i and on the edge e.

In order to compute the trace η of an edge-based basis function on the edge e the generalized eigenvalue problem

$$\begin{aligned} &\begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix}^T \begin{pmatrix} K_{11} & 0 & K_{1e} \\ 0 & K_{22} & K_{2e} \\ K_{1e}^T & K_{2e}^T & K_{ee} \end{pmatrix} \begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix} \eta \\ &= \lambda \begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix}^T \begin{pmatrix} M_{11} & 0 & M_{1e} \\ 0 & M_{22} & M_{2e} \\ M_{1e}^T & M_{2e}^T & M_{ee} \end{pmatrix} \begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix} \eta \end{aligned} \tag{4.22}$$

has to be solved. We can derive

$$\begin{aligned} &\begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix}^T \begin{pmatrix} K_{11} & 0 & K_{1e} \\ 0 & K_{22} & K_{2e} \\ K_{1e}^T & K_{2e}^T & K_{ee} \end{pmatrix} \begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix} \\ &= -K_{1e}^T K_{11}^{-1} K_{1e} - K_{2e}^T K_{22}^{-1} K_{2e} + K_{ee} \\ &= S_e^{(1)} + S_e^{(2)} \end{aligned}$$

with $S_e^{(i)} = -K_{ie}^T K_{ii}^{-1} K_{ie} + K_{ee}^{(i)}$ and $K_{ee}^{(i)}$ being the local contribution to K_{ee} from the element T_i. The matrices $S_e^{(i)}$ can be computed locally on both coarse elements without any communication. Analogously, we have

$$\begin{aligned}
&\begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix}^T \begin{pmatrix} M_{11} & 0 & M_{1e} \\ 0 & M_{22} & M_{2e} \\ M_{1e}^T & M_{2e}^T & M_{ee} \end{pmatrix} \begin{pmatrix} -K_{11}^{-1}K_{1e} \\ -K_{22}^{-1}K_{2e} \\ I \end{pmatrix} \\
&= K_{1e}^T K_{11}^{-1} M_{11} K_{11}^{-1} K_{1e} - K_{1e}^T K_{11}^{-1} M_{1e} - M_{1e}^T K_{11}^{-1} K_{1e} + K_{2e}^T K_{22}^{-1} M_{22} K_{22}^{-1} K_{2e} \\
&\quad - K_{2e}^T K_{22}^{-1} M_{2e} - M_{2e}^T K_{22}^{-1} K_{2e} + M_{ee} \\
&= \tilde{S}_e^{(1)} + \tilde{S}_e^{(2)}
\end{aligned}$$

with $\tilde{S}_e^{(i)} = K_{ie}^T K_{ii}^{-1} M_{ii} K_{ii}^{-1} K_{ie} - K_{ie}^T K_{ii}^{-1} M_{ie} - M_{ie}^T K_{ii}^{-1} K_{ie} + M_{ee}^{(i)}$ and $M_{ee}^{(i)}$ being the local contribution to M_{ee} from the element T_i. Also the matrix $\tilde{S}_e^{(i)}$ involves only local computations on the corresponding coarse element.

If an edge e is shared by two different processes, the Schur complement matrices $S_e^{(i)}$ and $\tilde{S}_e^{(i)}$ are computed independently and in parallel. We then solve the edge eigenvalue problem on the process with the lower rank. Communication is therefore involved when transferring $S_e^{(i)}$ and $\tilde{S}_e^{(i)}$ to the process responsible for the eigenvalue problem. This is implemented by standard MPI calls. Subsequently, the computed eigenmodes are communicated back to the process with the higher rank. In the results of our numerical experiments, the time for this communication is visible; see Section 4.4.3.

Finally,

$$\begin{pmatrix} -K_{ii}^{-1}K_{ie} \\ I \end{pmatrix} \eta \tag{4.23}$$

computes the local portion of the basis function on the coarse element T_i.

4.1.5 Computation of the Eigenvalue Problems

The *edge-based* eigenvalue problem (4.22) and the *fixed-interface* eigenvalue problem, respectively, are generalized eigenvalue problems of relatively small size. The latter has the dimension of the interior fine degrees of freedom of a single coarse element, the first one only of the number of degrees of freedom on a single edge.

Thus, all matrices needed for the computation can be stored as dense matrices, and the generalized eigenvalue problems are computed directly using the `LAPACK` [6] routine `DSYGVX`. This routine has a cubic complexity. Since our eigenvalue problems are defined on the edges and the interior of the coarse elements only, this remains affordable. Approximate iterative eigensolvers could, of course, also be used for larger eigenvalue problems. In our current implementation, the most expensive step is the computation of the *fixed-interface* interior bubble functions. In 3D, the cost assessment has to be revisited for the computation of the *fixed-interface* interior bubble functions. It also has to be taken into account that additional eigenvalue problems associated with subdomain faces have to be computed.

The underlying algorithm of `DSYGVX` is a Cholesky decomposition in order to reduce the generalized eigenvalue problem to a standard eigenvalue problem. Then the resulting matrix is reduced to tridiagonal form using an orthogonal similarity transformation and a QR-algorithm is employed for the computation of the eigenvectors.

We recall that some MPI communication is necessary to transfer the necessary information to build the eigenvalue problems. We apply a very simple load balancing approach: for each edge, we always gather the information on the process with the lower rank and we also solve the corresponding eigenvalue problem on this process.

4.2 The FETI-DP Method

The FETI-DP domain decomposition method is a divide-and-conquer approach to the iterative solution of linear systems discretized by finite elements and has been shown to be scalable and robust for a wide field of applications. It has been introduced by Farhat et al. in [82, 83]. FETI-DP [82, 83, 136, 138, 134, 139] and BDDC type [69, 55, 149, 146, 150] methods use coarse spaces constructed from constraints. These are typically implemented using partial assembly of the finite elements. This approach has facilitated the extension of the scalability of these methods; see, e.g., [199, 152, 133, 135, 153, 187]. Among the extensions are inexact FETI-DP methods which where introduced in [133]. Their parallel scalability has been demonstrated in [136, 169] for up to 65 000 processors. Recently, new scalable nonlinear versions of the FETI-DP method have been introduced in [125], and inexact FETI-DP variants scaled up to 786 432 cores on Mira BG/Q; see [126, 127]. For an introduction to domain decomposition methods, see, e.g., [197, 186]. The parallel FETI-DP implementation used in this thesis is based on [134, 169] and uses `PETSc` [15, 17] and `UMFPACK` [58]. There is proven robustness of FETI-DP for standard finite element discretizations of second-order self-adjoint elliptic partial differential equations, including (almost incompressible) linear elasticity, when the discontinuities occur only inside of each subdomain; see Gippert, Klawonn, and Rheinbach [100]. The second level or coarse space of FETI-DP has to be enhanced in order to obtain a robust iterative method for more general coefficient distributions. Such an enhancement of the second level of FETI-DP with suitable local eigenvectors could be done, e.g., along the lines of Klawonn, Radtke, and Rheinbach [132]; see also [151, 124]. Let us note that we are not considering robustness of FETI-DP for ACMS in this thesis. For overlapping domain decomposition methods and MsFEM, see, e.g., Aarnes and Hou [3] and Buck, Iliev, and Andrä [43, 44].

In FETI-DP methods the domain Ω is decomposed into N nonoverlapping subdomains $\{\Omega_i\}_{i=1,...,N}$. The corresponding local stiffness matrices $K^{(i)}$ and right hand sides $f^{(i)}$ are assembled for $i = 1, ..., N$. The system

$$Ku = \begin{pmatrix} K^{(1)} & & \\ & \ddots & \\ & & K^{(N)} \end{pmatrix} \begin{pmatrix} u^{(1)} \\ \vdots \\ u^{(N)} \end{pmatrix} = \begin{pmatrix} f^{(1)} \\ \vdots \\ f^{(N)} \end{pmatrix} = f \tag{4.24}$$

has no unique solution because the local stiffness matrices $K^{(i)}$ are not invertible for subdomains with $\partial\Omega_i \cap \partial\Omega = \emptyset$.

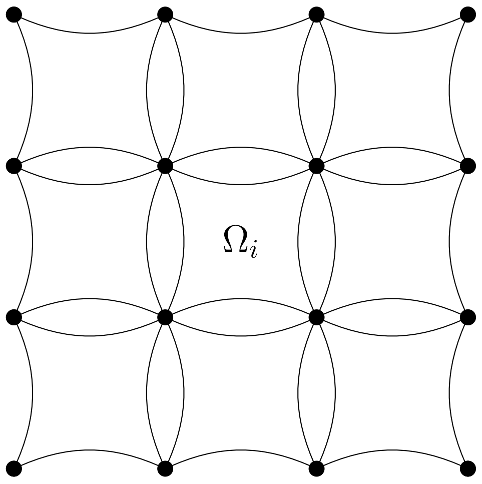


Figure 4.6: Our FETI-DP method uses primal vertices, i.e., we have continuity constraints for the FETI-DP iterates at all subdomain vertices.

To obtain a unique and continuous solution, we partition the interface $\Gamma' = \bigcup_{i=1}^{N} \partial\Omega_i \setminus \partial\Omega$ into dual (Δ) and primal variables (Π) first. We strongly enforce continuity in the primal variables by global assembly of the corresponding degrees of freedom. Continuity in the dual variables is enforced by the additional constraint $Bu = 0$. Here, B is the standard FETI jump operator; see [197].

By introducing Lagrange multipliers λ we can now formulate the *FETI-DP master system* which is a saddle point problem of the form

$$\begin{pmatrix} \widetilde{K} & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ 0 \end{pmatrix}. \tag{4.25}$$

Here, we have

$$\widetilde{K} = \begin{pmatrix} K_{BB}^{(1)} & & & \widetilde{K}_{\Pi B}^{(1)T} \\ & \ddots & & \vdots \\ & & K_{BB}^{(N)} & \widetilde{K}_{\Pi B}^{(N)T} \\ \widetilde{K}_{\Pi B}^{(1)} & \dots & \widetilde{K}_{\Pi B}^{(N)} & \widetilde{K}_{\Pi\Pi} \end{pmatrix} \quad \text{and} \quad \tilde{f} = \begin{pmatrix} \tilde{f}^{(1)} \\ \vdots \\ \tilde{f}^{(N)} \end{pmatrix}, \tag{4.26}$$

with B corresponding to all interior (I) and dual (Δ) variables.

If a sufficient number of degrees of freedom are chosen as primal variables, the matrix $\widetilde{K}$ is invertible. Here, we use primal vertices; see Figure 4.6.

The system can then be written as

$$F\lambda = d \tag{4.27}$$

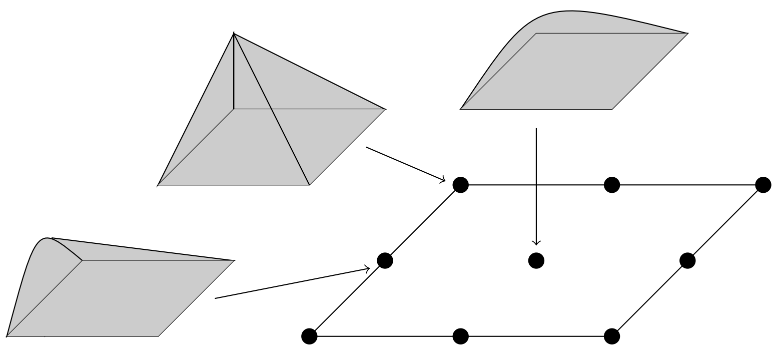


Figure 4.7: For the definition of the FETI-DP method we introduce nodes on the ACMS element and identify the ACMS degrees of freedom with these nodes.

with $F = B\widetilde{K}^{-1}B^T$ and $d = B\widetilde{K}^{-1}\tilde{f}$. By eliminating the interior variables first and using the Schur complement, F can also be written as $B_\Gamma \widetilde{S}_\Gamma^{-1} B_\Gamma^T$.

The Dirichlet preconditioner

$$M_D^{-1} = B_{D,\Gamma}\widetilde{S}B_{D,\Gamma}^T \tag{4.28}$$

can then be defined using $B_{D,\Gamma}$, which is a scaled version of B_Γ. We use simple multiplicity scaling, i.e., we scale by the inverse of the multiplicity of a node. For the many other possibilities of scaling, we refer to the literature; see, e.g., Toselli and Widlund [197] and the references therein and [73].

Typically, condition number bounds of the type

$$\kappa\left(M_D^{-1}F\right) \leq C\left(1 + \log\left(H/h\right)\right)^2, \tag{4.29}$$

can be shown for the preconditioned FETI-DP system where h is the size of the finite elements and H is the size of the subdomains. The constant C is independent of h, H, and possible coefficient jumps. Such estimates have been shown for finite element discretizations as well as higher order, spectral element, and isogeometric analysis discretizations. In all of these cases, the upper bound implies that the number of conjugate gradient iterations is bounded independently of the number of subdomains and thus is independent of the problem size.

4.2.1 FETI-DP Methods for ACMS Discretizations

Since all our data is distributed, i.e., our mesh is distributed and the ACMS shape functions are constructed in parallel, we may also apply a parallel solver building on the parallel distributed data. We use a parallel FETI-DP domain decomposition method where the subdomains are defined from the distribution of the ACMS elements.

A step essential to the fast convergence of the FETI-DP method is the selection of appropriate primal degrees of freedom. To get a better idea of how these are chosen, we identify the basis functions with nodes lying in the corresponding element; see Figure 4.7. One FETI-DP subdomain, in general, contains several ACMS elements, and following Figure 4.7, we see that the *fixed-interface* basis functions always correspond to interior degrees of freedom. In our FETI-DP method, for the sake of simplicity, we choose only the vertices to be primal, and thus only the *vertex-based* basis functions may correspond to primal nodes. The *vertex-based* basis functions which are not primal correspond, together with the *edge-based* basis functions, to the dual degrees of freedom. For standard finite element spaces in 2D, a vertex coarse space is sufficient to obtain numerical scalability in the sense of a $(1 + \log(H/h))^2$ condition number bound.

4.3 Model Problems

For the numerical experiments, we consider only two-dimensional problems on $\Omega = [0;1]^2$. We first employ the three example problems studied in [111]. We refer to [111] for comparisons with other special finite element methods. Additionally, we consider one coefficient function with much stronger oscillations, and another one which is even discontinuous. We use homogeneous Dirichlet boundary conditions for all problems. Note that homogeneous Neumann boundary conditions can be implemented by treating the nodes on the Neumann boundary like interior nodes.

4.3.1 Laplace Equation (Problem 1)

The first model problem is the Laplace equation,

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{4.30}$$

with $f(x,y) = 2x(1-x) + 2y(1-y)$. This corresponds to a coefficient matrix $A(x,y) = I$, where I denotes the identity matrix. The exact solution is given by

$$u(x,y) = x(1-x)y(1-y). \tag{4.31}$$

4.3.2 Equation with a Varying Coefficient (Problem 2)

The second model problem,

$$\begin{aligned} -\nabla \cdot \left(\frac{1}{1.2 + \cos(32\pi x(1-x)y(1-y))} \nabla u(x,y) \right) &= f(x,y) \quad \text{in } \Omega, \\ u &= 0 \qquad\quad\; \text{on } \partial\Omega, \end{aligned} \tag{4.32}$$

with $f(x,y) = 64\pi\,(x(1-x) + 2y(1-y))$, is equipped with a varying coefficient-matrix

$$A(x,y) = \left(\frac{1}{1.2 + \cos(32\pi x(1-x)y(1-y))} \right) I, \tag{4.33}$$

where I denotes the identity matrix; see also Figure 4.8. The exact solution is given by

$$u(x,y) = (1.2 \cdot 32\pi)x(1-x)y(1-y) + \sin(32\pi x(1-x)y(1-y)). \tag{4.34}$$

4.3.3 Equation with a Highly-Oscillating Coefficient (Problem 3)

Here, we consider a model problem with a highly-oscillating coefficient matrix. The model problem is given by

$$\begin{aligned} -\nabla \cdot (A(x,y)\nabla u(x,y)) &= -1 \quad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{4.35}$$

with $A(x,y) = \left(\frac{2+1.8\sin(25\pi x)}{2+1.8\cos(25\pi y)} + \frac{2+\sin(25\pi y)}{2+1.8\sin(25\pi x)}\right) I$, where I denotes the identity matrix; see also Figure 4.8.

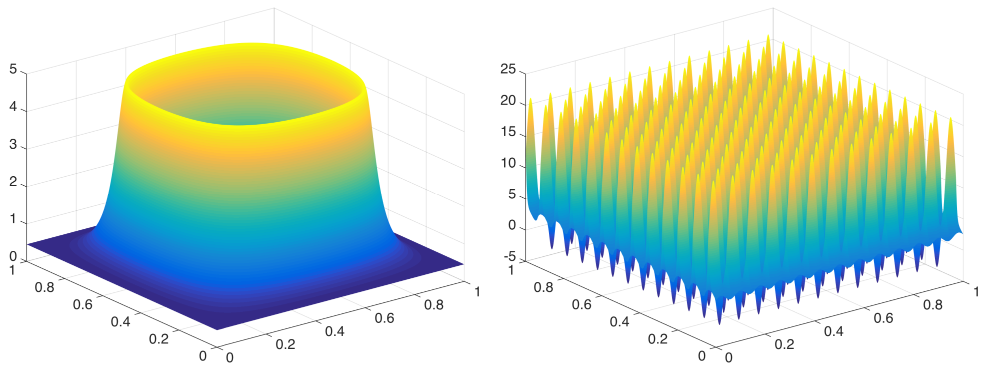


Figure 4.8: Coefficients of Problems 2 (left) and 3 (right); see Section 4.3.2 and Section 4.3.3, respectively.

4.3.4 Another Equation with a Highly-Oscillating Coefficient (Problem 4)

We modify the coefficient function of Problem 3 (cf. Figure 4.8) such that the ratio of the maximum and the minimum coefficient value is much higher. Therefore, we choose

$$c(x,y) := \left(\frac{2+1.99\sin(25\pi x)}{2+1.99\cos(25\pi y)} + \frac{2+\sin(25\pi y)}{2+1.99\sin(25\pi x)}\right) \tag{4.36}$$

as the coefficient function, such that $A(x,y) = c(x,y)I$, where I denotes the identity matrix; see Figure 4.9. The contrast of the coefficient function is

$$\frac{\max_{(x,y)\in[0;1]\times[0;1]} c(x,y)}{\min_{(x,y)\in[0;1]\times[0;1]} c(x,y)} \approx 230.78. \tag{4.37}$$

The fourth problem is then given by (4.35) with the coefficient function $A(x,y)$ given above.

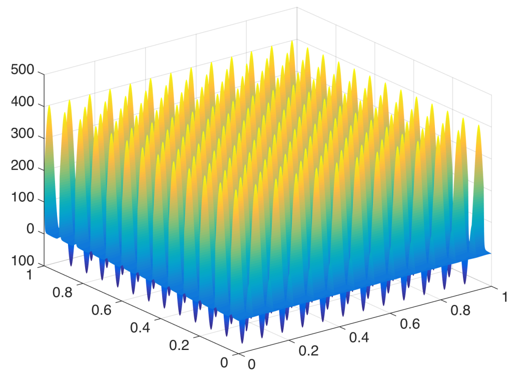


Figure 4.9: Coefficient function of Problem 4; see Section 4.3.4.

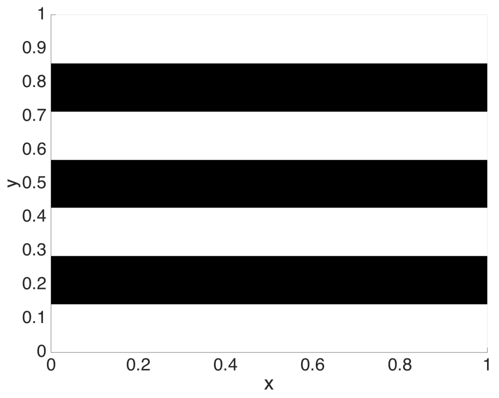


Figure 4.10: Discontinuous coefficient function A. The black channels correspond to a coefficient value of 10^6, the remaining white parts correspond to a coefficient value of one.

4.3.5 An Equation with Discontinuous Coefficients (Problem 5)

We also consider the following problem with discontinuous coefficients

$$\begin{aligned} -\nabla \cdot (A(x,y)\nabla u(x,y)) &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{4.38}$$

with $A(x,y) = c(x,y)I$, where I denotes the identity matrix. The values of the discontinuous coefficient function c are depicted in Figure 4.10, where black corresponds to a value of 10^6 and white to a value of one. Let us note that the entries of this matrix A are not in $\mathcal{C}^1(\overline{\Omega})$ anymore and thus the convergence theory from [110] does not apply.

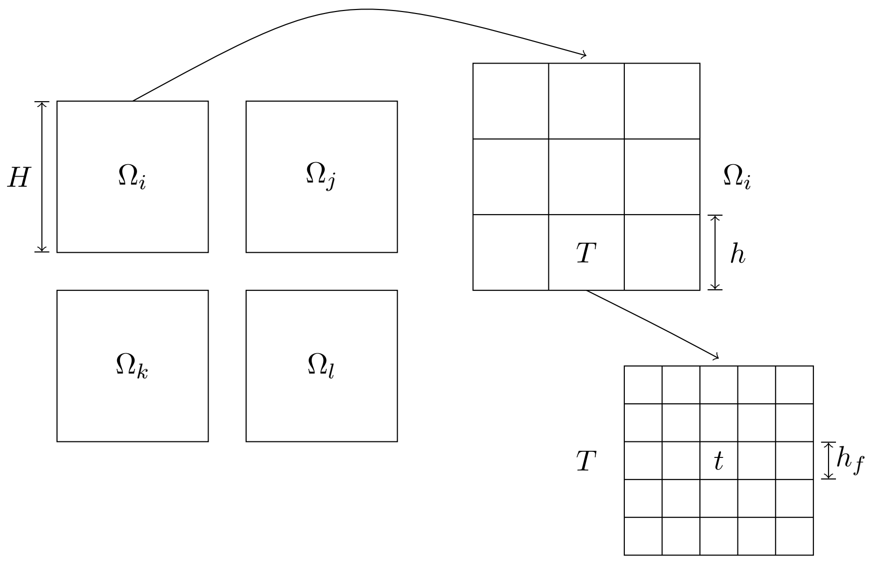


Figure 4.11: Scales and mesh sizes involved in our FETI-DP approach for ACMS discretizations. The subdomains $\Omega_i, \Omega_j, \Omega_k$, and Ω_l are the FETI-DP subdomains of diameter H. Each subdomain contains $(H/h)^2$ ACMS elements. Each ACMS element of diameter h uses shape functions defined on a fine Q1 mesh of size h_f.

4.4 Numerical Results

The linear systems arising from ACMS discretizations can be ill-conditioned as seen in Table 4.1. Here, for a fixed ratio $h/h_f = 30$, the number of ACMS elements is increased for Problem 2; see Section 4.3.2. The estimated condition number seems to grow accordingly to $(1/h)^2$ (see also Figure 4.12) and approaches $1.80 \cdot 10^4$ for $1/h = 512$. The use of an efficient preconditioner for the solution of the ACMS system is thus advisable. In this chapter, we apply the FETI-DP domain decomposition method. Many other parallel preconditioners, such as the GDSW preconditioner (cf. Chapter 2), are, of course, also possible.

For all computations, for the sake of simplicity, we have used $I_T = I_e = 1$ for all $T \in \tau_h$ and for all $e \subset \Gamma$, i.e., we always use the first eigenmode only. Higher values of I_T and I_e could also be used to further improve the approximation properties, at the cost of a larger discretization space.

In order to compare the accuracy of the discretized solutions, we apply the energy functional

$$\mathcal{E}(v) = \frac{a(v,v)}{2} - L(v) = -\frac{a(v,v)}{2} = -\frac{L(v)}{2}; \tag{4.39}$$

cf. [111, eq. (5.4)]. The functional is minimized for the solution of the variational problem u, and thus is a measure for the quality of the approximate solutions.

$1/h$	Cond.
32	55.69
64	269.22
128	1 098.15
256	4 407.93
512	17 686.94

Table 4.1: Estimated condition number for different ACMS discretizations for $h/h_f = 30$ (Problem 2). The condition number was estimated from the Lanczos process [171].

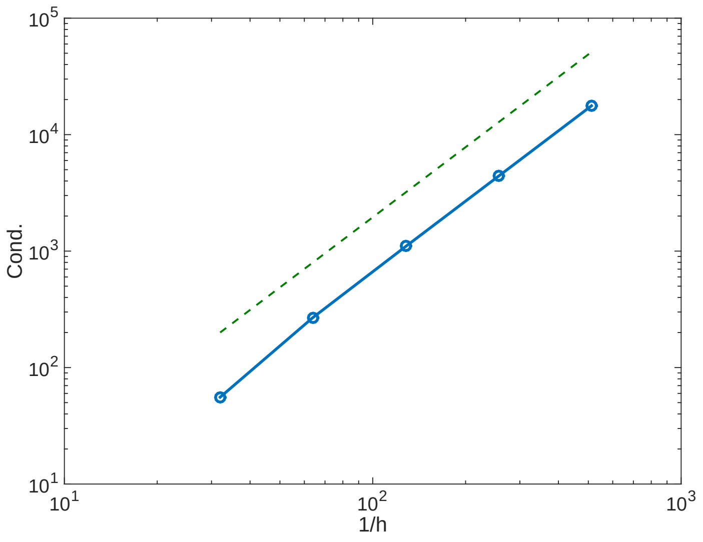


Figure 4.12: Estimated condition number for different ACMS discretizations for $h/h_f = 30$; see Table 4.1. The dashed line represents the slope of $1/h^2$ growth.

4.4.1 Numerical Scalability of the FETI-DP Method for ACMS Discretizations

In Table 4.2 and Figure 4.13 the condition number of the preconditioned FETI-DP system for the ACMS method applied to Problem 2 is presented. Figure 4.13 also includes a least square fit of a second-order polynomial in $\log(H/h)$ to the data. These numerical results strongly suggest a $(1+\log(H/h))^2$ bound for this problem with highly varying coefficients inside ACMS elements and thus inside subdomains. No theory is currently known for the setting presented here. The numerical results are therefore very encouraging.

Additionally, in Tables 4.4, 4.5, 4.6, and 4.7, we observe that the condition number of the preconditioned FETI-DP operator ("FETI-DP"/"Cond.") and thus also the number of FETI-DP conjugate gradient iterations ("FETI-DP"/"It.") stay bounded for a growing number of subdomains $(1/H)^2$ if H/h and h/h_f are kept constant; see Section 4.4.3.

We can thus conclude that we obtain numerical scalability for the FETI-DP method for our ACMS model problems.

H/h	Cond.
4	3.52
8	4.60
12	5.31
16	5.81
20	6.20
24	6.52
28	6.80

Table 4.2: Estimated condition number of the preconditioned FETI-DP system for the ACMS discretization for $1/H = 16$ and $h/h_f = 20$ (Problem 2).

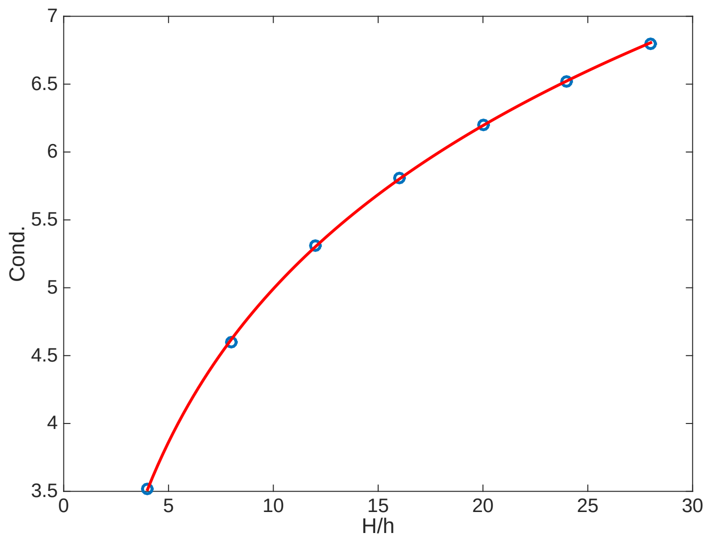


Figure 4.13: Estimated condition number for different H/h for ACMS discretizations with $h/h_f = 20$ (Problem 2) and a fit of a second order polynomial in $\log(H/h)$ to the data shown in Table 4.2.

4.4.2 Using Different Fine Discretizations in ACMS

In Table 4.3 and Figure 4.14 a brute force Q1 discretization (“Q1”) is compared with different ACMS discretizations (“ACMS”) for Problem 2 using the same number of (coarse) degrees of freedom (dof). We vary $h/h_f = 5, 10, 20, 30$, i.e., we consider a different number of (internal) fine degrees of freedom. For our comparison we observe the convergence of the energies $\mathcal{E}_{\mathrm{ACMS}}$ and $\mathcal{E}_{\mathrm{Q1}}$ to the (known/extrapolated) energy $\mathcal{E}^*$ of the exact solution.

In Figure 4.14 we see that the expected convergence order is achieved for all methods. For the same number of (coarse) degrees of freedom, the accuracy of the ACMS method is always significantly higher than that of the brute force method.

Results in Table 4.3 show that, for $h/h_f = 30$, the error in the energies of the (brute force) Q1 discretization falls below that of the ACMS discretization only if between 60 and 80 times more degrees of freedom are invested. Of course, a sufficiently fine underlying Q1 mesh is needed in order to approximate the ACMS basis functions accurately. This is essential for the improvement of the accuracy compared to a standard Lagrangian finite element discretization. However, the number of nodes on the underlying fine Q1 mesh also corresponds to the amount of computational work needed for the generalized eigenvalue problems to be solved. It also corresponds to the memory needed to store the local data, such as the corresponding local stiffness and mass matrices, on the ACMS elements.

The situation is similar for $h/h_f = 20$ although in Figure 4.14 one can clearly see that the horizontal distance of the black (ACMS) and the blue (Q1) curve is smaller for $h/h_f = 20$ than for $h/h_f = 30$. For $h/h_f = 5$ and $h/h_f = 10$, the quality of the approximation is reduced further but we still do not see a break down. This may be caused by the structure of Problem 2. For problems with a clearly defined micro scale we would expect a sudden drop in the quality of the approximation as soon as the ACMS fine mesh fails to resolve the micro scale properly.

In Figure 4.14, we also see that the ACMS discretization profits from the use of a fine underlying Q1 mesh in order to approximate the ACMS basis functions. Of course, in our current implementation, the computational work to compute the ACMS shape functions increases superlinearly for an increasing ratio h/h_f. In the following sections, we therefore choose $h/h_f = 20$ as a compromise between accuracy and computational cost.

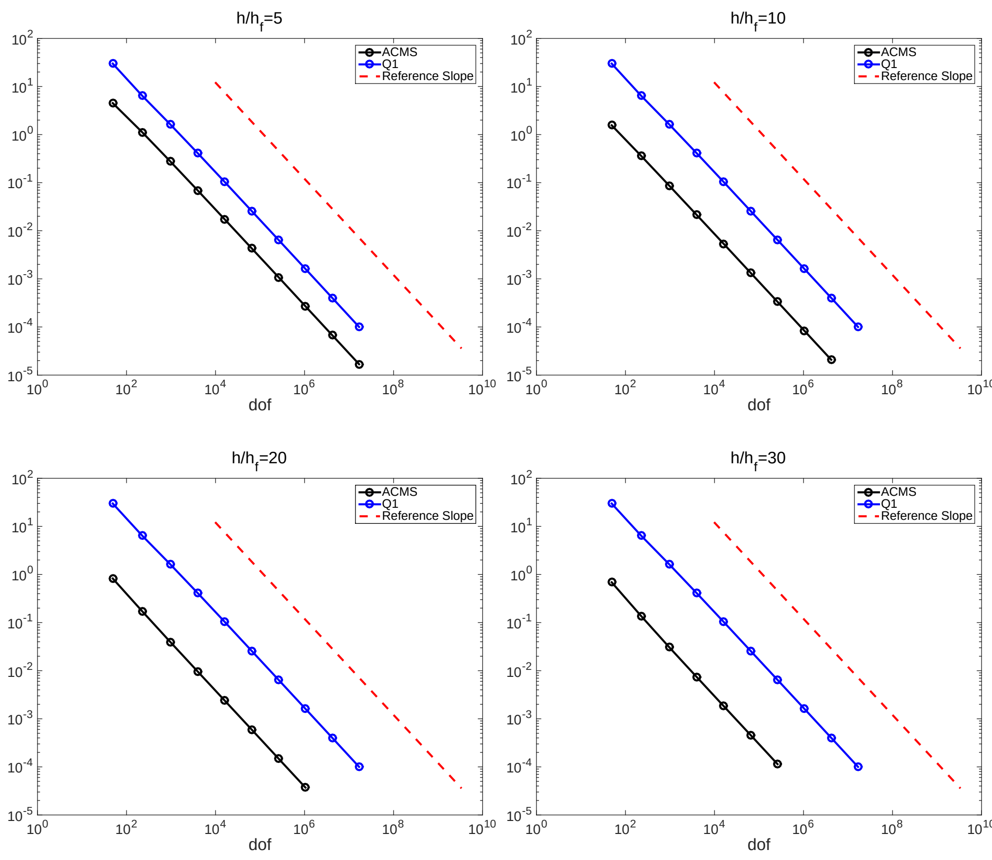


Figure 4.14: Comparison of $\mathcal{E}_{\mathrm{ACMS}} - \mathcal{E}^*$ ("ACMS") and $\mathcal{E}_{\mathrm{Q1}} - \mathcal{E}^*$ ("Q1") for the ACMS special finite element discretization and a Q1 discretization; cf. Table 4.3. Here, $\mathcal{E}^*$ is the energy of the (known) exact solution, $\mathcal{E}_{\mathrm{ACMS}}$ the energy of the ACMS solution, and $\mathcal{E}_{\mathrm{Q1}}$ the energy of the (brute force) Q1 solution. The "Reference Slope" refers to the slope of 1/dof.

	$\mathcal{E}_{\text{ACMS}} - \mathcal{E}^*$				$\mathcal{E}_{\text{Q1}} - \mathcal{E}^*$
dof	$h/h_f = 5$	$h/h_f = 10$	$h/h_f = 20$	$h/h_f = 30$	
49	$4.51 \cdot 10^{0}$	$1.56 \cdot 10^{0}$	$8.26 \cdot 10^{-1}$	$6.89 \cdot 10^{-1}$	$2.99 \cdot 10^{1}$
225	$1.12 \cdot 10^{0}$	$3.61 \cdot 10^{-1}$	$1.72 \cdot 10^{-1}$	$1.37 \cdot 10^{-1}$	$6.42 \cdot 10^{0}$
961	$2.77 \cdot 10^{-1}$	$8.69 \cdot 10^{-2}$	$3.96 \cdot 10^{-2}$	$3.08 \cdot 10^{-2}$	$1.63 \cdot 10^{0}$
3 969	$6.90 \cdot 10^{-2}$	$2.14 \cdot 10^{-2}$	$9.61 \cdot 10^{-3}$	$7.42 \cdot 10^{-3}$	$4.11 \cdot 10^{-1}$
16 129	$1.72 \cdot 10^{-2}$	$5.34 \cdot 10^{-3}$	$2.38 \cdot 10^{-3}$	$1.83 \cdot 10^{-3}$	$1.03 \cdot 10^{-1}$
65 025	$4.30 \cdot 10^{-3}$	$1.33 \cdot 10^{-3}$	$5.94 \cdot 10^{-4}$	$4.57 \cdot 10^{-4}$	$2.57 \cdot 10^{-2}$
261 121	$1.08 \cdot 10^{-3}$	$3.33 \cdot 10^{-4}$	$1.48 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$6.43 \cdot 10^{-3}$
1 046 529	$2.69 \cdot 10^{-4}$	$8.33 \cdot 10^{-5}$	$3.71 \cdot 10^{-5}$		$1.61 \cdot 10^{-3}$
4 190 209	$6.72 \cdot 10^{-5}$	$2.08 \cdot 10^{-5}$			$4.02 \cdot 10^{-4}$
16 769 025	$1.68 \cdot 10^{-5}$				$1.00 \cdot 10^{-4}$

Table 4.3: Comparison of the energies for the ACMS special finite element discretization and a Q1 discretization (Problem 2). Here, $\mathcal{E}^*$ is the energy of the (known) exact solution, $\mathcal{E}_{\text{ACMS}}$ the energy of the ACMS solution, and $\mathcal{E}_{\text{Q1}}$ the energy of the (brute force) Q1 solution.

4.4.3 Weak Parallel Scalability

In this section, we present weak parallel scaling results for the parallel ACMS approach as the discretization and using a parallel FETI-DP method as an iterative solver. The results for $h/h_f = 20$ are shown in Tables 4.4, 4.5, and 4.6. We recall that h is the size of an ACMS element and h_f the size of a fine Q1 finite element; see Figure 4.11. In these experiments, since $h/h_f = 20$ is fixed, the number of fine degrees of freedom for each ACMS element is kept constant. Moreover, since $H/h = 28$ is fixed, the number of ACMS elements for each FETI-DP subdomain is also constant. Since $1/H$ is growing, the number of subdomains increases, as well as the number of MPI ranks and processor cores, from $2^2 = 4$ to $32^2 = 1024$.

The number of FETI-DP subdomains, i.e., $(1/H)^2 \in \{4, 16, 64, 256, 1024\}$, is always identical to the number of processor cores, i.e., we use up to 1024 cores of a Cray XT6. We also always have one FETI-DP subdomain for each process or processor core. The Cray XT6 has 24 cores per node (AMD Magny Cours 1.9 GHz).

Since the ACMS systems are small compared to the total number of fine degrees of freedom, the time spent to solve the system with our parallel FETI-DP method is in the order of only one second or less; cf. Tables 4.4, 4.5, and 4.6.

Let us briefly describe the columns of Tables 4.4, 4.5, and 4.6. In "ACMS"/"Fine Q1" we measure the time for the assembly of the Q1 elements on the fine ACMS mesh of size h_f. The fine mesh is needed to compute the ACMS shape functions. As expected, this phase scales perfectly.

The column "ACMS"/"Shape" presents the time needed for the construction of the ACMS shape functions by computing harmonic extensions and solving generalized eigenvalue problems using the local fine Q1 meshes on each ACMS element. Because of MPI communication that is necessary in the construction of the ACMS edge shape functions (see Section 4.1.4), this phase does not scale perfectly, but good scalability is still achieved.

The column "ACMS"/"Ass." presents the time for the assembly of the ACMS system using the ACMS shape functions. Since the system is small, the time spent here is not significant.

The column "FETI-DP"/"Time" denotes the time for the solution of the ACMS system by the FETI-DP method, "FETI-DP"/"It." denotes the number of conjugate gradients iterations, and "FETI-DP"/"Cond." denotes the estimated condition number obtained from the Lanzcos process. Since, again, the system is small compared to the number of processor cores, the time spent here is also not significant.

"Total Time" denotes the complete time to solution, and "Speedup" and "Efficiency" denote the corresponding parallel speedup and efficiency where $1/H = 2$, i.e., 4 processor cores, is the baseline. For perfect weak parallel scalability the "Total Time" should stay constant, resulting in a perfect speedup of 256 on 1024 cores (compared to the baseline of 4 cores) and a perfect parallel efficiency of 100 %. We achieve a parallel efficiency of 79 %, 83 %, and 84 % for 1024 cores in Tables 4.4, 4.5, and 4.6, respectively.

Comparing the column "Fine Q1" in Tables 4.4, 4.5, and 4.6, we see that the time for the assembly of the fine Q1 problem is much larger for Problem 2 (21 s) and again larger for Problem 3 (36 s to 38 s). This is due to the expensive evaluations of the trigonometric functions in the coefficient functions of Problem 2 and 3. One evaluation of a cosine function for each Gauß point is needed for Problem 2 (see Section 4.3.2) and four evaluations of trigonometric functions for Problem 3 (see Section 4.3.3). The parallel scalability is also illustrated in Figure 4.15.

In Table 4.7 we have also included weak parallel scalability for Problem 3 and $h/h_f = 30$. Since the eigenvalue problems are now larger, the dense linear algebra and the computation of the eigenvalue problems by dense QR becomes increasingly inefficient. Indeed, a quick analysis of detailed timers shows that a large amount of the computing time is spent in the computation of the eigenvalue problem for the interior bubble function. These results indicate that, for

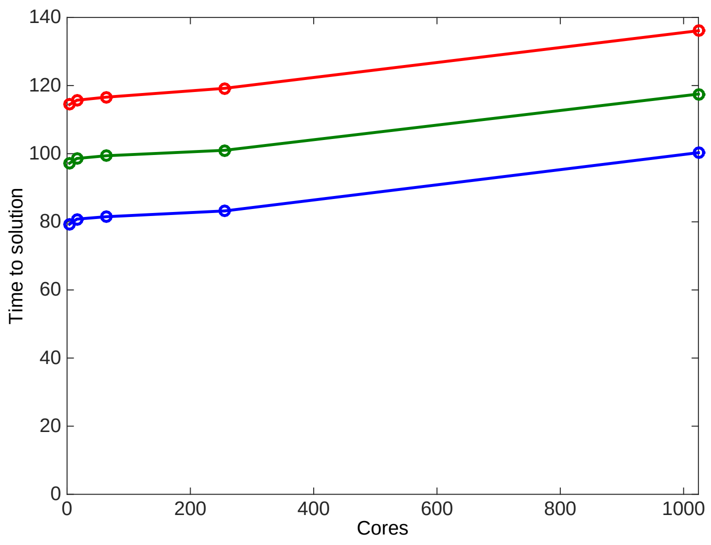


Figure 4.15: Weak parallel scalability from 4 to 1024 processor cores on a Cray XT6; cf. the data in Tables 4.4, 4.5, and 4.6.

three-dimensional problems, an approximate solution of the eigenvalue problems will have to be used.

	ACMS Time	FETI-DP		Total	Parallel
$1/H$	Fine Q1 / Shape / Ass.	Time	It. / Cond.	Time	Speedup / Efficiency
2	2.91 s / 75.65 s / 0.97 s	0.09 s	4 / 2.43	79.33 s	1 / 100 %
4	2.94 s / 77.06 s / 1.05 s	0.11 s	6 / 6.11	80.81 s	3.93 / 98.17 %
8	2.93 s / 77.56 s / 1.71 s	0.33 s	6 / 6.66	81.53 s	15.57 / 97.30 %
16	2.98 s / 78.88 s / 2.17 s	0.67 s	14 / 6.78	83.24 s	61.37 / 95.90 %
32	2.97 s / 95.51 s / 4.06 s	1.10 s	13 / 6.80	100.31 s	202.46 / 79.08 %

Table 4.4: Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 1). The number of MPI ranks is $(1/H)^2$.

4.4.3.1 Discussion of the FETI-DP Solution Phase

The FETI-DP method is known to scale well, even on large supercomputers, and for very large problems; cf., e.g., [127, 126]. In our case the ACMS system is indeed very small compared to the number of processor cores invested.

Let us briefly discuss details. For the FETI-DP method, we see numerical scalability, i.e., the number of conjugate gradient iterations stays bounded for increasing $1/H$. The parallel scalability of the FETI-DP solution phase by itself is far from perfect but this is mainly a result of the very short absolute solution times (0.09 s–1.35 s). The size of the largest ACMS problem solved by FETI-DP in Tables 4.4, 4.5, and 4.6 is only 804 609 degrees of freedom. This is

	ACMS Time	FETI-DP		Total	Parallel
$1/H$	Fine Q1 / Shape / Ass.	Time	It. / Cond.	Time	Speedup / Efficiency
2	21.14 s / 75.30 s / 0.95 s	0.09 s	5 / 2.15	97.20 s	1 / 100 %
4	21.18 s / 76.58 s / 1.04 s	0.11 s	7 / 5.97	98.55 s	3.95 / 98.63 %
8	21.21 s / 77.13 s / 1.73 s	0.34 s	14 / 6.68	99.35 s	15.65 / 97.84 %
16	21.33 s / 78.23 s / 2.83 s	0.71 s	15 / 6.79	100.95 s	61.62 / 96.29 %
32	21.37 s / 94.29 s / 4.22 s	1.16 s	14 / 6.81	117.53 s	211.72 / 82.70 %

Table 4.5: Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 2). The number of MPI ranks is $(1/H)^2$.

	ACMS Time	FETI-DP		Total	Parallel
$1/H$	Fine Q1 / Shape / Ass.	Time	It. / Cond.	Time	Speedup / Efficiency
2	35.94 s / 77.72 s / 0.97 s	0.10 s	6 / 2.69	114.43 s	1 / 100 %
4	35.94 s / 78.99 s / 1.09 s	0.13 s	11 / 8.41	115.74 s	3.95 / 98.87 %
8	36.06 s / 79.47 s / 1.92 s	0.40 s	18 / 8.64	116.62 s	15.70 / 98.12 %
16	36.49 s / 81.27 s / 3.06 s	0.78 s	19 / 8.69	119.24 s	61.42 / 95.97 %
32	37.53 s / 96.52 s / 4.79 s	1.35 s	18 / 8.24	136.12 s	215.21 / 84.07 %

Table 4.6: Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 3). The number of MPI ranks is $(1/H)^2$.

a very small problem for a FETI-DP method running on 1 024 processor cores, resulting in fewer than 800 degrees of freedom for each core. The number of degrees of freedom on the ACMS fine discretization is of course much larger, i.e., 321 million degrees of freedom. But these fine degrees of freedom have been eliminated already in the earlier phase. From a more detailed analysis we have found that the increase of the FETI-DP time almost completely stems from an increasing time spent in the FETI-DP conjugate gradient iteration. This is a result of the growth in conjugate gradient iterations up to $1/H = 8$, i.e., 64 FETI-DP subdomains. Thus, the asymptotic bound is approached only for more than 256 subdomains. This is typical for FETI-DP methods in 2D. Moreover, the conjugate gradient iteration for the FETI-DP system includes global MPI communication as well as MPI collective operations that may indeed add up to a noticeable amount of a fraction of a second on 1 024 cores of a Cray XT6. For larger linear systems, this is usually an insignificant portion of the solution time. We thus expect to achieve good parallel scalability for much larger problems and numbers of cores than presented here.

	ACMS	FETI-DP		Total	Parallel
$1/H$	Fine Q1 / Shape / Ass.	Time	It. / Cond.	Time	Speedup / Efficiency
2	26.27 s / 193.79 s / 0.57 s	0.03 s	6 / 2.41	220.57 s	1 / 100 %
4	26.35 s / 204.12 s / 0.61 s	0.04 s	10 / 7.18	231.00 s	3.82 / 95.48 %
8	26.54 s / 204.32 s / 1.30 s	0.26 s	16 / 7.38	231.62 s	15.24 / 95.23 %
16	26.43 s / 206.80 s / 2.35 s	0.60 s	17 / 7.48	234.35 s	58.46 / 91.35 %
32	26.65 s / 210.85 s / 3.73 s	1.07 s	17 / 7.16	239.08 s	236.18 / 92.26 %

Table 4.7: Weak scaling for $H/h = 16$ and $h/h_f = 30$ (Problem 3).

4.4.3.2 Highly Heterogeneous Problems

Here, we consider Problem 4 and Problem 5 which have been introduced in Section 4.3 and which are much more heterogeneous compared to Problems 1, 2, and 3.

Let us first consider Problem 4; cf. Section 4.3.4. The reference energy, computed using a brute force discretization and Richardson extrapolation, for the corresponding boundary value problem

$$\begin{aligned} -\nabla \cdot (A(x,y)\nabla u(x,y)) &= -1 \quad \text{in } \Omega, \\ u &= 0 \quad\ \ \text{on } \partial\Omega, \end{aligned} \tag{4.40}$$

is $-1.65867679589956 \cdot 10^3$; cf. [111].

As can be seen in Table 4.8, the approximation properties of the ACMS method compared to standard Q1 elements is even better for this coefficient function. The best approximation for ACMS elements cannot be reached using Q1 elements on the same number of MPI processes.

Table 4.9 shows the corresponding numbers of iterations and condition numbers, as well as the timings resulting when applying FETI-DP on the ACMS system. The parallel scaling is as good as for Problem 3, however the condition and iteration numbers are slightly better for Problem 3, cf. Table 4.6.

Let us now consider Problem 5; cf., Section 4.3.5. The corresponding scalability results are shown in Table 4.10. It can be seen, that the weak scalability is comparable to the results of Problem 1, cf. Table 4.4. Especially the condition numbers and timings of the FETI-DP method for the discontinuous case are very good, even though no special scaling has been applied.

	$\mathcal{E}_{\text{ACMS}} - \mathcal{E}^*$				$\mathcal{E}_{\text{Q1}} - \mathcal{E}^*$
dof	$h/h_f = 5$	$h/h_f = 10$	$h/h_f = 20$	$h/h_f = 30$	
49	$5.03 \cdot 10^{-4}$	$3.23 \cdot 10^{-4}$	$1.98 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$	$8.82 \cdot 10^{-4}$
225	$3.48 \cdot 10^{-4}$	$2.24 \cdot 10^{-4}$	$8.66 \cdot 10^{-5}$	$6.34 \cdot 10^{-5}$	$8.13 \cdot 10^{-4}$
961	$2.12 \cdot 10^{-4}$	$7.18 \cdot 10^{-5}$	$3.70 \cdot 10^{-5}$	$2.81 \cdot 10^{-5}$	$4.71 \cdot 10^{-4}$
3 969	$5.95 \cdot 10^{-5}$	$2.45 \cdot 10^{-5}$	$1.23 \cdot 10^{-5}$	$9.94 \cdot 10^{-6}$	$2.56 \cdot 10^{-4}$
16 129	$1.87 \cdot 10^{-5}$	$6.36 \cdot 10^{-6}$	$3.19 \cdot 10^{-6}$	$2.60 \cdot 10^{-6}$	$7.52 \cdot 10^{-5}$
65 025	$5.06 \cdot 10^{-6}$	$1.86 \cdot 10^{-6}$	$1.05 \cdot 10^{-6}$	$8.95 \cdot 10^{-7}$	$2.53 \cdot 10^{-5}$
261 121	$1.38 \cdot 10^{-6}$	$5.65 \cdot 10^{-7}$	$3.60 \cdot 10^{-7}$	$3.22 \cdot 10^{-7}$	$6.98 \cdot 10^{-6}$
1 046 529	$5.01 \cdot 10^{-7}$	$2.95 \cdot 10^{-7}$	$2.43 \cdot 10^{-7}$	$2.34 \cdot 10^{-7}$	$1.94 \cdot 10^{-6}$
4 190 209	$2.87 \cdot 10^{-7}$	$2.35 \cdot 10^{-7}$			$6.50 \cdot 10^{-7}$
16 769 025	$2.33 \cdot 10^{-7}$				$3.25 \cdot 10^{-7}$

Table 4.8: Comparison of the energies for the ACMS special finite element discretization and a Q1 discretization (Problem 4). Here, $\mathcal{E}^*$ is the energy of the (known) exact solution, $\mathcal{E}_{\text{ACMS}}$ the energy of the ACMS solution and $\mathcal{E}_{\text{Q1}}$ using the (brute force) Q1 solution.

	ACMS Time	FETI-DP		Total	Parallel
$1/H$	Fine Q1 / Shape / Ass.	Time	It. / Cond.	Time	Speedup / Efficiency
2	35.94 s / 77.72 s / 0.97 s	0.1 s	6 / 2.93	114.43 s	1 / 100 %
4	35.93 s / 79.10 s / 1.11 s	0.14 s	12 / 19.17	115.85 s	3.95 / 98.77 %
8	36.05 s / 79.83 s / 2.15 s	0.48 s	23 / 17.65	117.05 s	15.64 / 97.96 %
16	36.25 s / 80.82 s / 3.89 s	1.06 s	27 / 18.42	118.82 s	61.64 / 96.31 %
32	37.10 s / 96.56 s / 7.94 s	2.40 s	30 / 22.34	136.76 s	214.20 / 83.67 %

Table 4.9: Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 4). The number of MPI ranks is $(1/H)^2$.

	ACMS Time	FETI-DP		Total	Parallel
$1/H$	Fine Q1 / Shape / Ass.	Time	It. / Cond.	Time	Speedup / Efficiency
2	8.44 s / 75.41 s / 0.96 s	0.09 s	6 / 2.69	84.6 s	1 / 100 %
4	8.47 s / 76.66 s / 1.06 s	0.12 s	11 / 8.41	85.94 s	3.94 / 98.44 %
8	8.65 s / 77.08 s / 1.94 s	0.41 s	18 / 8.64	86.83 s	15.59 / 97.43 %
16	8.68 s / 78.53 s / 3.20 s	0.83 s	19 / 8.69	88.72 s	61.03 / 95.36 %
32	8.69 s / 94.71 s / 5.12 s	1.46 s	18 / 8.24	105.57 s	205.15 / 80.14 %

Table 4.10: Weak scaling for $H/h = 28$ and $h/h_f = 20$ (Problem 5). The number of MPI ranks is $(1/H)^2$.

4.5 Conclusion

We presented a parallel implementation of the ACMS special finite element method, which shows good parallel scalability. Using this implementation, we could perform a comparison with a large brute force discretization using standard bilinear finite elements. These computations allowed us to compare the accuracy of the ACMS method with standard bilinear finite elements for different settings. It was also possible to study the influence of the approximation quality of the eigensystems (fine mesh) on the approximation quality of the ACMS method. We also applied the FETI-DP domain decomposition preconditioner to the ACMS linear system. Our numerical results show that FETI-DP is numerically scalable in this case, i.e., we could see that the computed condition number of the preconditioned system grows quadratic-logarithmically, depending on the size of the subdomain problems. This is the condition number estimate which is usually obtained for FETI-DP applied to standard finite elements and could motivate further theoretical investigations to analytically prove such a condition number estimate. In our present study in two dimensions, direct dense eigensolvers were used. For future ACMS discretizations in three dimensions, iterative sparse eigensolvers are probably necessary.

All three types of the ACMS basis functions are harmonic extensions to the interior nodes, which is very similar to the construction of the coarse basis functions of the GDSW coarse space; cf. Chapter 2. However, due to the generalized eigenvalue problems additional information about the fine scale are introduced in the finite element space. Thus, it seems natural to test the use of the ACMS basis functions in a coarse space of two-level overlapping Schwarz preconditioners to investigate whether this additional information has a positive effect if used in the coarse space of a preconditioner as well. Therefore, we refer to the next chapter, Chapter 5, where we follow this approach and present some preliminary results.

5 Coarse Spaces for Overlapping Schwarz Methods Based on the ACMS Space

In Chapter 2, a parallel implementation of the GDSW preconditioner, i.e., a two-level overlapping Schwarz preconditioner with an energy-minimizing coarse space, has been presented and applied to homogeneous Laplacian and elasticity problems; it has also been applied to fluid-structure interaction problems in Chapter 3. However, for highly heterogenous (multiscale) problems, enriched coarse spaces (by, e.g., eigenfunctions) are required to obtain robust preconditioners for the corresponding systems.

Such coarse spaces, which are typically denoted as adaptive coarse spaces, have been developed for many different domain decomposition methods. Mostly, they involve the solution of generalized eigenvalue problems to facilitate certain estimates which are needed to prove the condition number bounds for the corresponding domain decomposition method. For instance, adaptive coarse spaces are available for two-level Neumann-Neumann methods [37, 36], for overlapping Schwarz algorithms [188, 189, 94, 95] and based thereon for FETI and BDD methods [190, 101], and for FETI-DP and BDDC methods [132, 49, 130, 151, 131, 165].

The ACMS special finite element method (cf. [111] and Chapter 4) features nodal (*vertex-specific*) as well as *edge-based* interface basis functions. The latter are given by the solution of a generalized eigenvalue problem, very similar to the ones used in the construction of some adaptive coarse spaces. The *fixed-interface* basis functions of the ACMS method, however, are not useful for the construction of a coarse space since they vanish on the interface and are therefore not related to the coupling of the subdomains.

In this chapter, we first introduce briefly a coarse space for overlapping Schwarz methods based on the ACMS space; see Section 5.1. Next, we investigate the numerical performance of the preconditioner for some specific model problems in Section 5.2. Therefore, we consider model problems from Chapter 4 and some other heterogeneous model problems with jumping coefficients to compare the overlapping Schwarz preconditioner with ACMS coarse space to other overlapping Schwarz preconditioners. In Section 5.3, we discuss how a space spanned by *edge-based* ACMS basis functions can be approximated to obtain a robust preconditioner, without solving generalized eigenvalue problems. This can be performed by a heuristic procedure in a purely algebraic way. Finally, we provide a conclusion in Section 5.4. This chapter contains ongoing joint work with Axel Klawonn, Jascha Knepper, and Oliver Rheinbach; cf. [104] and Jascha Knepper's master's thesis [140].

Note that the *vertex-specific* ACMS basis functions, i.e., the basis functions of the Multiscale Finite Element Method (MsFEM), have already been used in the coarse space for the overlapping Schwarz method for linear elastic model problems in two and three dimensions by Buck et al. in [43, 44]. In [3], Aarnes et al. used MsFEM basis functions in the second level of an overlapping Schwarz preconditioner for an interface problem applied to scalar multiscale problems.

As we are going to discuss later, for some heterogeneous problems, the MsFEM coarse basis functions outperform the standard Lagrangian coarse basis functions of the standard coarse space, whereas the performance is comparable for homogeneous problems. However, as for the computation of the GDSW basis functions, all ACMS basis functions could also be constructed on unstructured decompositions without the need of an additional coarse triangulation. This is not the case for the standard Lagrangian basis functions.

We restrict our observations to the serial computations performed with `MATLAB` for model problems of the form (4.1). A parallel implementation would be straight-forward using the software from [103]; see also Sections 4.1.4 and 4.1.5, but it is not considered in this thesis.

5.1 Definition of an ACMS-based Coarse Space

Consider model problems of the form (4.1) on a two-dimensional domain Ω. Analogously to Chapter 2, Ω is decomposed into nonoverlapping and overlapping subdomains, such that a two-level overlapping Schwarz preconditioner based on the ACMS finite element space is formally given by

$$M_{\text{ACMS}}^{-1} = \Phi_{\text{ACMS}} A_0^{-1} \Phi_{\text{ACMS}}^T + \sum_{i=1}^{N} R_i^T \tilde{A}_i^{-1} R_i \tag{5.1}$$

with

$$A_0 = \Phi_{\text{ACMS}}^T A \Phi_{\text{ACMS}}. \tag{5.2}$$

Here, Φ_{ACMS} is a matrix containing those ACMS basis functions that are used in the coarse space. Only the matrix Φ_{ACMS} distinguishes this preconditioner from the GDSW preconditioner given in cf. Equation (2.2). To define the ACMS basis functions in this context, the (triangular or quadrilateral) ACMS finite elements $T \in \tau_h$ correspond to the subdomains of the nonoverlapping decomposition.

From the set of all ACMS basis functions (cf. Section 4.1.2),

$$\left(\bigcup_{e \subset \Gamma} \{ E_\Omega (\tilde{\tau}_{i,e}) : 1 \leq i \leq I_e \} \right) \cup \left(\bigcup_{P \in \Omega} \{ \varphi_P \} \right) \cup \left(\bigcup_{T \in \tau_h} \{ z_{i,T} : 1 \leq i \leq I_T \} \right), \tag{5.3}$$

we select only the *edge-based* basis functions, $E_\Omega (\tilde{\tau}_{i,e})$, and the *vertex-specific* basis functions, φ_P. Since they vanish on the interface and are therefore not related to the coupling of the subdomains, we omit the *fixed-interface* basis functions, $z_{i,T}$. Note that the *vertex-specific* basis functions are equal to the standard Lagrangian coarse space functions for tetrahedral or quadrilateral decompositions and piecewise constant coefficients on the subdomains.

The simulations of our studies of the ACMS coarse space are performed with `MATLAB`, using the CG method for the solution of the linear systems, a relative stopping criterion of 10^{-8}, and a maximum number of 500 iterations. If not marked otherwise, we include all *vertex-specific* basis functions and the *edge-based* basis function corresponding to the smallest eigenvalue for each edge from (5.3) in the coarse space, i.e., $I_e = 1$. Therefore, the dimension of the coarse space is the same as for the GDSW preconditioner which, for two-dimensional scalar problems, includes one basis function for each vertex and for each edge; cf. Chapter 2.

For simplicity, we use square fine meshes with mesh size h and corresponding bilinear (Q1) finite element functions. The square subdomains have mesh size H, and the ACMS basis functions are approximated on the fine quadratic mesh. For all overlapping Schwarz preconditioners, we use an overlap of $2h$.

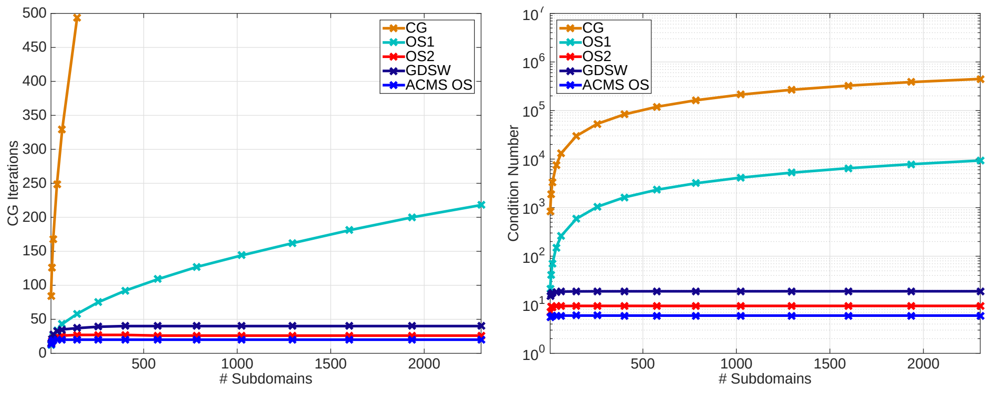


Figure 5.1: Numerical scalability of the unpreconditioned system (denoted as CG), the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS): number of CG iterations (left) and estimated condition numbers (Lanczos, right) for the Laplacian model problem (Problem 1) from Section 4.3.1 with $H/h = 32$. Note that here OS2 is equal to using multiscale finite element method (MsFEM) basis functions in the coarse space.

5.2 Performance of the ACMS-based Coarse Space

In Figures 5.1, 5.2, and 5.3, we consider numerical scalability for the model problems from Sections 4.3.1, 4.3.2, and 4.3.3, i.e., Problems 1–3. We compare the unpreconditioned system (denoted as CG), a one-level overlapping Schwarz preconditioner (OS1), the standard two-level overlapping Schwarz preconditioner (OS2), the GDSW preconditioner (GDSW), and the two-level preconditioner with an ACMS-based coarse space (ACMS OS). The results in Figures 5.1, 5.2, and 5.3 show excellent numerical scalability for OS2, GDSW, and ACMS OS, with the latter showing a slightly better performance than the other two-level preconditioners. As expected, the unpreconditioned CG and the one-level preconditioner are not numerically scalable.

Note that, for Problem 1, the *vertex-specific* basis functions of the ACMS coarse space are just the coarse basis functions of the standard two-level Schwarz preconditioner since the coefficient function is constant. Furthermore, the results suggest that the standard Lagrangian basis functions are already sufficient to obtain good scalability for Problems 1–3.

However, when considering problems with high coefficient jumps, like the coefficient distributions depicted in Figure 5.4, the ACMS coarse space is clearly

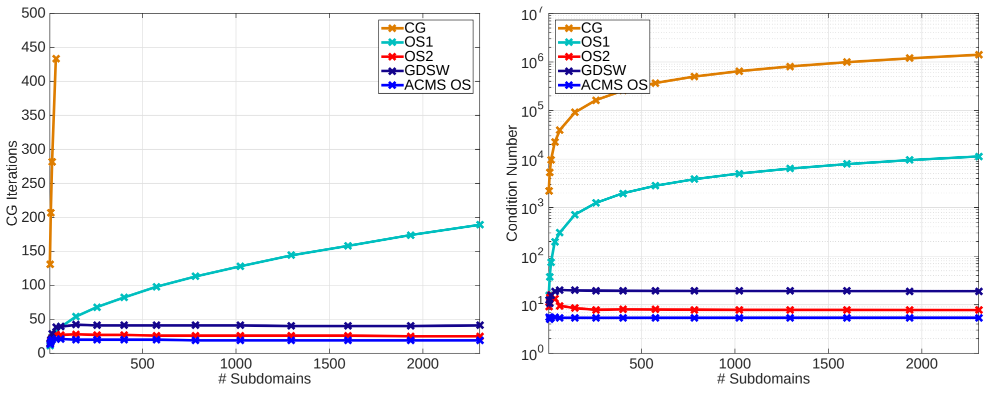


Figure 5.2: Numerical scalability of the unpreconditioned system (denoted as CG), the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS): number of CG iterations (left) and estimated condition numbers (Lanczos, right) for Problem 2 from Section 4.3.2 with $H/h = 32$.

superior to the standard two-level preconditioner. Table 5.1 shows results for model problem (4.1) with the coefficient distribution from Figure 5.4 (left). We observe that the *edge-based* coarse basis functions provide the robustness of the preconditioner; see the number of iterations and the condition number for ACMS Schwarz using only the *edge-based* basis functions (ACMS-E OS). This is intuitively understood since the severity of the problem is induced by the channels cutting through the edges of the decomposition. For this example, the GDSW coarse space is robust as well due to the basis functions belonging to the corresponding edges.

The results in Table 5.2 show that, for the coefficient function in Figure 5.4 (right), the nodal (*vertex-specific*) coarse basis functions yield robustness of the ACMS OS preconditioner (ACMS-V OS), whereas the nodal basis functions of the standard two-level preconditioner and the *edge-based* basis functions (ACMS-E OS) fail to do so. This is remarkable since the number of nodal basis functions and their support are the same for OS2 and ACMS-V OS; only the scaling of the coupling is different because the *vertex-specific* basis functions of the ACMS coarse space are extended by discrete harmonic extensions, incorporating information about the coefficient function in that way.

Note that the edge values of the *vertex-specific* basis functions in (4.18) are not well-defined for the coefficient function depicted in Figure 5.4 (right). This is due to the discontinuities of the coefficient function along the edges. To obtain

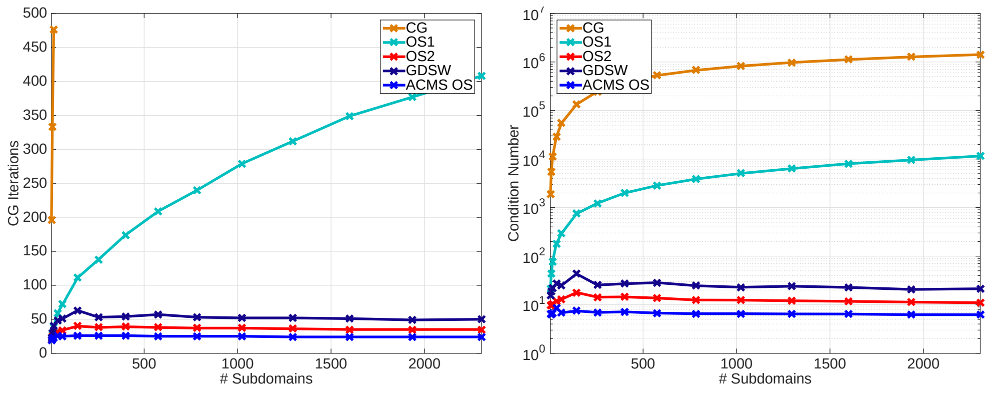


Figure 5.3: Numerical scalability of the unpreconditioned system (denoted as CG), the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS): number of CG iterations (left) and estimated condition numbers (Lanczos, right) for Problem 3 from Section 4.3.3 with $H/h = 32$.

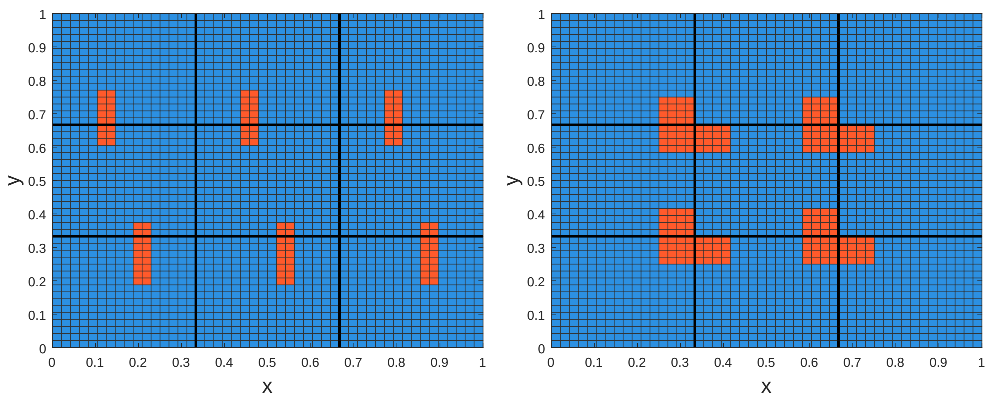


Figure 5.4: Coefficient functions with six short vertical channels (left) and with four inclusions located at the vertices of the decomposition (right). The red coefficient is 10^6, the blue coefficient is one.

a good performance, for each vertex on the edge, the maximum coefficient of all neighboring fine elements has to be chosen.

For the example depicted in Figure 5.5, neither ACMS-E OS, ACMS-V OS, nor ACMS OS is a robust preconditioner; see Table 5.3. It can be observed that the *edge-based* eigenfunctions corresponding to the smallest eigenvalue are not a suitable choice here, even though each edge is cut by only one channel. Remarkably, the GDSW coarse space is robust for this distribution of the coef-

Preconditioner	# Its.	Cond. number	Dim. coarse space
Unpreconditioned CG	n.c.	$3.6 \cdot 10^8$	-
OS1	59	$9.9 \cdot 10^5$	-
OS2	67	$5.3 \cdot 10^5$	4
GDSW	32	14.8	16
ACMS OS	22	6.0	16
ACMS-E OS (only edges)	25	10.9	12
ACMS-V OS (only vertices)	37	$5.5 \cdot 10^5$	4

Table 5.1: Number of iterations and estimated condition number for the unpreconditioned system, the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS), the ACMS coarse space neglecting the *vertex- specific* basis functions (ACMS-E OS), and the ACMS coarse space neglecting the *edge-based* basis functions (ACMS-V OS) solving the model problem (4.1) with the coefficient function displayed in Figure 5.4 (left); for the unpreconditioned system, CG did not converge within the maximum number of 500 iterations: "n.c." stands for "no convergence" and the condition number estimate is given at the time of termination of the CG iteration. The mesh is partitioned into 3×3 subdomains with $H/h = 16$. Note that ACMS-V OS is equal to using MsFEM basis functions in the coarse space.

ficient function, indicating that its basis functions corresponding to the edges are a better choice here.

In the next section, we show that, manually, ACMS *edge-based* eigenfunctions can be chosen such that a good condition number can be recovered. The number of necessary eigenfunctions for this strategy is equal to the number of channels cutting through the corresponding edge. Note that, if the inclusions/channels with high coefficient value touch the Dirichlet boundary, no *edge-based* basis functions are necessary.

Preconditioner	# Its.	Cond. number	Dim. coarse space
Unpreconditioned CG	n.c.	$3.8 \cdot 10^8$	-
OS1	46	$3.9 \cdot 10^6$	-
OS2	54	$8.7 \cdot 10^5$	4
GDSW	52	$1.5 \cdot 10^6$	16
ACMS OS	20	4.2	16
ACMS-E OS (only edges)	49	$3.9 \cdot 10^6$	12
ACMS-V OS (only vertices)	21	5.6	4

Table 5.2: Number of iterations and estimated condition number for the unpreconditioned system, the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS), the ACMS coarse space neglecting the *vertex-specific* basis functions (ACMS-E OS), and the ACMS coarse space neglecting the *edge-based* basis functions (ACMS-V OS) solving the model problem (4.1) with the coefficient function displayed in Figure 5.4 (right); for the unpreconditioned system, CG did not converge within the maximum number of 500 iterations: "n.c." stands for "no convergence" and the condition number estimate is given at the time of termination of the CG iteration. The mesh is partitioned into 3×3 subdomains with $H/h = 16$. Note that ACMS-V OS is equal to using MsFEM basis functions in the coarse space.

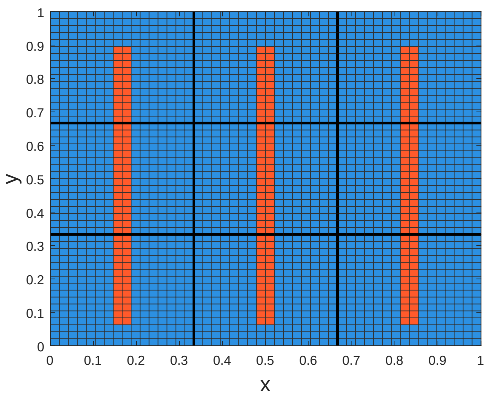


Figure 5.5: Coefficient function with three vertical channels. The red coefficient is 10^6, the blue coefficient is one.

Preconditioner	# Its.	Cond. number
Unpreconditioned CG	n.c.	$3.9 \cdot 10^8$
OS1	38	$6.4 \cdot 10^5$
OS2	41	$3.5 \cdot 10^5$
GDSW	24	10.5
ACMS OS	38	$3.6 \cdot 10^5$

Table 5.3: Number of iterations and estimated condition number for the unpreconditioned system, the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), and the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS) solving the model problem (4.1) with the coefficient function displayed in Figure 5.5; for the unpreconditioned system, CG did not converge within the maximum number of 500 iterations: “n.c.” stands for “no convergence” and the condition number estimate is given at the time of termination of the CG iteration. The mesh is partitioned into 3×3 subdomains with $H/h = 16$.

5.3 Algebraic Approximations of the ACMS Coarse Space

In this section, we first consider an even more severe coefficient function than in the previous section, i.e., a coefficient function with 12 vertical channels of different length cutting through a different number of subdomain edges; see Figure 5.6. As can be observed from the results shown in Table 5.4, the ACMS OS coarse space is not sufficient for this coefficient function. We notice that the eigenfunction corresponding to the smallest eigenvalue captures only the jumps corresponding to the rightmost channel cutting through the four middle horizontal edges; cf. Figure 5.7.

We denote the minimal set of *edge-based* basis functions from (5.3) needed to obtain a low condition number as the optimal set. For the horizontal edges with $y = 0.5$, the optimal basis functions are the eigenfunctions corresponding to the first, the 13th, and the 14th lowest eigenvalues. Using the corresponding coarse space incorporating the optimal *edge-based* basis functions on each edge, we obtain very good results; cf. Table 5.4. Note that each edge contains 23 nodes, and thus, 23 different eigenfunctions can be computed on each edge. We have selected the three optimal functions manually by the method of trial and error because it seems that the choice of the optimal eigenfunctions is not directly related to the magnitude of the corresponding eigenvalue. Thus, even if we were able to select the optimal eigenfunctions, it might be necessary to compute all eigenfunctions first. In a parallel simulation, the solution of the eigenvalue problems is local work but, nonetheless, the computation of all eigenfunctions is costly. The number of basis functions needed to achieve good preconditioning is equal to the number of cuts of channels through subdomain edges, which is not surprising when reviewing the literature on adaptive coarse spaces, e.g., [95, 132].

Looking at the values of the optimal eigenfunctions on the middle edges (cf. Figure 5.7, middle), we observe that each of the eigenfunctions is somehow a disturbed representation of the coefficient jumps corresponding to one of the channels. Thus, it seems natural that the ACMS coarse space with optimal *edge-based* eigenfunctions can also be approximated by reconstructing the jumps on the edges corresponding to the channels manually. The edge values of the basis functions resulting from this strategy are depicted in Figure 5.7 (right), and the corresponding promising results are shown in Table 5.4. Note that, in order to obtain these results, the values of the reconstructed basis functions on the edges are set to zero where the coefficient is low and to one (for means of

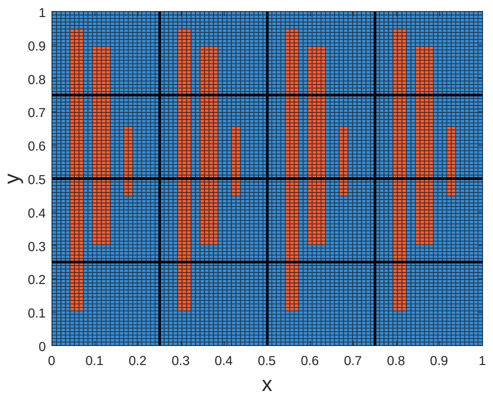


Figure 5.6: Coefficient function with 12 vertical channels. The red coefficient is 10^6, the blue coefficient is one.

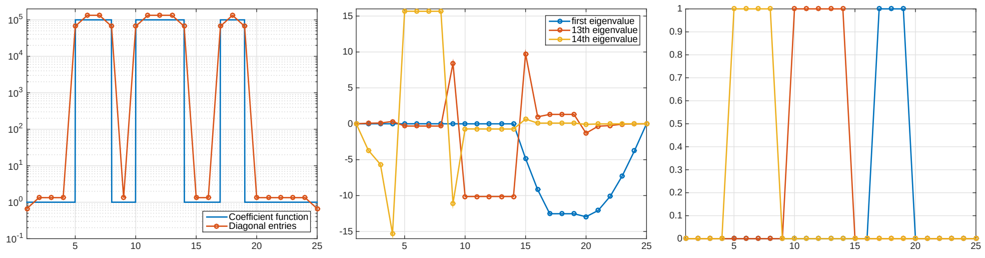


Figure 5.7: Values of the coefficient function and diagonal entries of the stiffness matrix corresponding to the horizontal edges with $y = 0.5$ from Figure 5.6 (left); the corresponding three optimal edge-based eigenfunctions selected for the coarse space (middle); the three reconstructed edge functions (right). The eigenfunctions corresponding to the first, 13th, and 14th lowest eigenvalues have been selected.

normalization) where the coefficient is high. Then, as for the *edge-based* basis functions in the ACMS space, we extend the edge values to the interior nodes by discrete harmonic extensions; cf. Equation (4.6). We denote the resulting overlapping Schwarz preconditioner by ACMS-R OS.

The fact that, following this strategy, we approximate the space spanned by the optimal *edge-based* ACMS basis functions rather than the functions themselves can be observed from the example depicted in Figures 5.9 and 5.10. In this case, each optimal *edge-based* ACMS basis function does not clearly correspond to one of the channels, however the channels can be represented as linear combinations of the basis functions.

Note that, for the coefficient function in Figure 5.6, the dimension of all ACMS based coarse spaces (ACMS OS, ACMS-O OS, and ACMS-R OS) is

Preconditioner	# Its.	Cond. number	Dim. coarse space
Unpreconditioned CG	n.c.	$7.0 \cdot 10^8$	-
OS1	184	$3.4 \cdot 10^6$	-
OS2	222	$9.3 \cdot 10^5$	9
GDSW	153	$2.9 \cdot 10^5$	33
ACMS OS	151	$7.9 \cdot 10^5$	33
ACMS-O OS	32	23.3	33
ACMS-R OS	28	9.7	33

Table 5.4: Number of iterations and estimated condition number for the unpreconditioned system, the one-level overlapping Schwarz preconditioner (OS1), the two-level overlapping Schwarz preconditioner (OS2), the ACMS coarse space with only the *edge-based* basis function corresponding to the smallest eigenvalue (ACMS OS), the ACMS coarse space with the manually selected three optimal *edge-based* basis functions (ACMS-O OS), and the ACMS coarse space with reconstructed *edge-based* basis functions (ACMS-R OS) solving the model problem (4.1) with the coefficient function displayed in Figure 5.6; for the unpreconditioned system, CG did not converge within the maximum number of 500 iterations: “n.c.” stands for “no convergence” and the condition number estimate is given at the time of termination of the CG iteration. The mesh is partitioned into 4×4 subdomains with $H/h = 24$.

the same; however, the results for ACMS-O and ACMS-R OS are significantly better; cf. Table 5.4.

We present two strategies to reconstruct the jumps corresponding to the channels on an edge:

(a) using the values of the coefficient function, or

(b) using the diagonal entries of the stiffness matrix.

The second approach arises from the idea of constructing the coarse space functions algebraically, i.e., without the need of additional information about the geometry or about the coefficient function. This is also a crucial advantage of the GDSW preconditioner since it can be constructed in an algebraic fashion as well; cf. Chapter 2. As can be observed in Figure 5.7 (left), the plot of the diagonal entries of the stiffness matrix has almost the same shape as the coefficient function on the edge. Thus, we can also employ the matrix entries to define the edge values of the *edge-based* basis functions.

Note that, if the width of a channel is equal to or less than the size of one fine element, the detection of the jump may be difficult using the diagonal entries of the stiffness matrix. This is because of the averaging effect of the integrals in the matrix. We do not go into detail here, but in these cases the strategy can be extended by looking at the off-diagonal entries.

For both strategies, we specify a threshold (here, e.g., 100) and go through all nodes of an edge of the decomposition. If the quotient of the coefficient values of two neighboring nodes (or of the corresponding diagonal entries, respectively) is higher than the threshold, we set the values of the *edge-based* basis function to one, until the quotient again gets lower, i.e., lower than the reciprocal value of the threshold; we set the values to zero elsewhere. Then we continue going through the nodes to set the values of the second *edge-based* basis function, and so on. In this way, one basis function is constructed for each channel cutting through the edge which is under consideration. We carry out this procedure for each edge of the decomposition.

Additionally, to obtain a purely algebraic implementation, the values of the coefficient function $A(x)$ in (4.18), i.e., in the definition of the *vertex-specific* basis functions on the edges, can also be approximated using the corresponding diagonal entries of the stiffness matrix; cf. Figure 5.7 (left). As a result, the edge values are well-defined, even if the coefficient function is discontinuous along the edge, as, e.g., in Figure 5.4 (right). For this coefficient function, we obtain 22 CG iterations and an estimated condition number of 5.7 when using the diagonal entries of the stiffness matrix instead of the coefficient function itself. To obtain these results, no *edge-based* basis functions have been used. The results are therefore comparable to the results obtained with the exact values of the coefficient function (ACMS-V OS/MsFEM); cf. Table 5.2.

Remarkably, for an even more complicated example, like the coefficient function depicted in Figure 5.8, three *edge-based* basis function from the ACMS space (see Figure 5.9, right) are still sufficient to obtain a low condition number and a small number of CG iterations: the estimated condition number is 8.53, and 25 CG iterations are needed. However, the basis functions have been selected manually here as well.

For all three approaches for the reconstruction of the jumps which are depicted in Figure 5.10, we obtain excellent results as well. In particular, we observe that three *edge-based* ACMS-R OS functions are sufficient, and thus, no special treatment of the jump from 10^4 to 10^8 is necessary here.

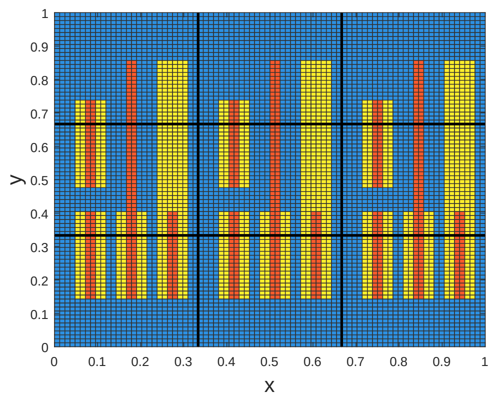


Figure 5.8: Coefficient function with vertical channels and three different values: the red coefficient is 10^8, the yellow coefficient is 10^4, and the blue coefficient is one.

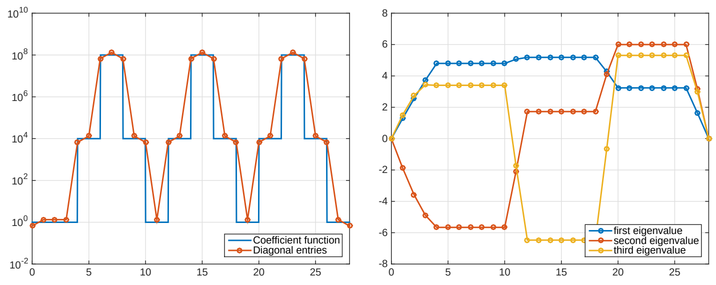
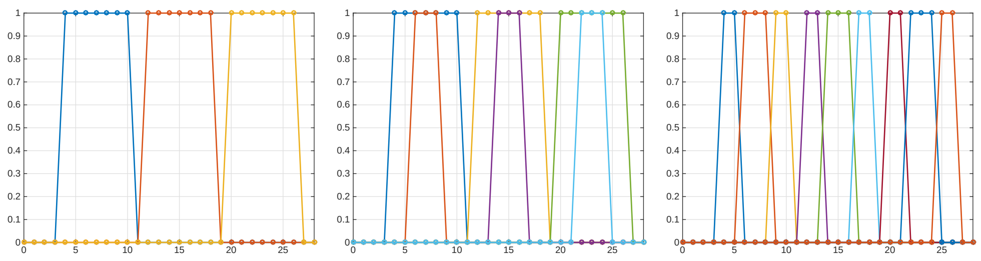


Figure 5.9: Values of the coefficient function and diagonal entries of the stiffness matrix corresponding to the horizontal edges with $y = 1/3$ from Figure 5.8 (left) and the corresponding three optimal edge-based eigenfunctions selected for the coarse space (right); the eigenfunctions corresponding to the three lowest eigenvalues have been selected.



# basis functions	# Its.	Cond. number
1 (left)	28	7.9
2 (middle)	30	7.9
3 (right)	27	7.9

Figure 5.10: Reconstructed *edge-based* basis functions for the the horizontal edges with $y = 1/3$ from Figure 5.8 using one (left), two (middle), or three (right) basis functions for each channel, and the corresponding number of iterations and estimated condition number (Lanczos) for the solution of problem (4.1) with the coefficient function depicted in Figure 5.8.

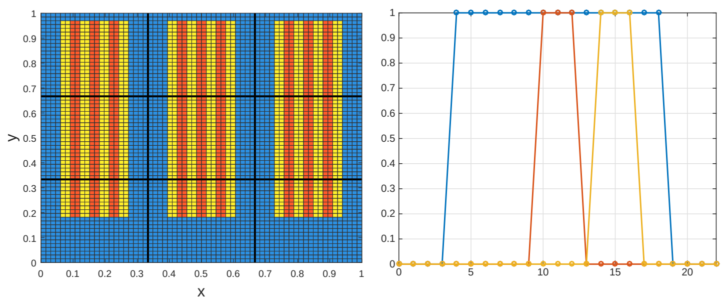


Figure 5.11: Another coefficient function with vertical channels and three different values (left): the red coefficient is 10^8, the yellow coefficient is 10^4, and the blue coefficient is one; the corresponding reconstructed *edge-based* basis functions.

However, for the coefficient function depicted in Figure 5.11 (left), more than one basis function is needed for each edge: three basis functions are needed for each edge to obtain good results, i.e., 23 CG iterations and an estimated condition number of 8.6; cf. Figure 5.11 (right). To fully understand how the edge values have to be chosen for general jumping coefficient functions, further investigation is necessary.

Note that for this coefficient function, also the GDSW coarse space leads to comparably good results, i.e., 50 CG iterations and an estimated condition number of 2 572.4, keeping in mind that only one basis function is employed for each edge.

5.4 Conclusion

We have presented a coarse space for two-level overlapping Schwarz preconditioners which is based on the ACMS space. The ACMS OS preconditioner scales well for the oscillating model problems under consideration. However, for some model problems with high coefficient jumps (e.g., channel problems), the *edge-based* basis functions corresponding to the smallest eigenvalue may neither be optimal nor sufficient. Hence, we have found that an optimal set of *edge-based* eigenfunctions from the full ACMS space can be selected manually to obtain a robust coarse space, where the number of eigenfunctions appears to correspond to the number of channels cutting through the edge. However, no correlation of the magnitude of the eigenvalue to the optimality of the eigenfunction could be observed here.

From investigating the edge values of the optimal *edge-based* basis functions, we could approximate heuristically the space spanned by the optimal *edge-based* basis functions reconstructing the jumps on the edges. Therefore, we use either the values of the coefficient function or, alternatively, the diagonal entries of the stiffness matrix. These coarse basis functions lead to very good condition numbers and do not require the solution of generalized eigenvalue problems, which clearly reduces the computational work needed for construction of the coarse space. However, our heuristic strategy still needs to be extended to general coefficient distributions.

Nonetheless, these strategies are promising as an extension of our parallel implementation of the GDSW preconditioner (cf. Chapter 2): we observed that the AMCS-R OS coarse space may be built in a completely algebraic fashion, i.e., just from the global stiffness matrix. An extension of the GDSW coarse space is not always necessary for heterogeneous problems since the basis functions of the GDSW coarse space which correspond to an edge help to cope with one channel cutting through this edge. However, for more complicated problems, the GDSW coarse space should be enriched by additional basis functions.

Another advantage of the reconstructed basis functions is that, since no generalized eigenvalue problems have to be solved, no mass matrices are needed for their construction. In contrast to that, in the ACMS space the mass matrices have to be build additionally; cf. (4.22).

Conclusion and Future Work

Conclusion

In this thesis, the development of a complete framework for the simulation of Fluid-Structure Interaction (FSI) in coronary arteries has been described. To accurately simulate the distribution of stresses in the arterial wall, the software libraries `LifeV` and `FEAP` have been coupled and a benchmark problem for the testing of the numerical framework and the corresponding software has been set up. An extensive numerical study has been carried out to investigate different boundary conditions and various space discretizations. The results of fully-coupled FSI simulations of several heartbeats using an anisotropic polyconvex hyperelastic material model and an anisotropic viscoelastic material model for the structure have been presented, and the stress distributions in wall of the benchmark geometry have been discussed. For both parts of the simulations, i.e., the ramp phase and the heartbeat phase, extensive mesh convergence studies have been carried out. The results indicate that, for the highly nonlinear material models considered in this thesis, piecewise quadratic or $\bar{\mathrm{F}}$ elements should be preferred instead of piecewise linear elements.

Initiated by the immense computation times required for the simulations, a parallel two-level Schwarz preconditioner has been implemented to be used as a preconditioner for the structural block. Indeed, as it turned out, the structural block is the most crucial part for preconditioning in the presented setting. As a consequence, for larger time steps and for the nonlinear material models which have been used in this thesis, a significant reduction of the simulation time could be achieved, just by replacing the preconditioner for the structural block.

In addition to that, the parallel implementation of the GDSW preconditioner has been tested for Laplacian and linear elastic model problems. Very good numerical and parallel scalability has been observed for the full coarse space as well as for the resulting coarse space when omitting the rotation-based basis functions. These observations involved structured and unstructured domain decompositions in two and three dimensions. The largest three-dimensional linear elastic problem includes 334 million unknowns and has been solved on 8 000 MPI-processes. A hybrid version of the GDSW preconditioner has been presented, and it has been observed that the hybrid version reduces the number of Krylov iterations significantly for the Laplacian and the linear elastic model problems. The parallel implementation is held flexible with respect to the extension of the preconditioner by, e.g., additional coarse basis functions.

A parallel implementation of a special finite element method based on Approximate Component Mode Synthesis (ACMS) in two dimensions has been presented. Convergence results for up to 1 024 MPI ranks and more than 16 million unknowns indicate very good approximation properties of the special finite element method for highly heterogeneous problems. The FETI-DP method has been used to solve the linear system of equations arising from the ACMS discretization, with condition numbers similar to the case of standard Lagrangian basis functions. Excellent parallel scalability could be observed.

Finally, the interface-based basis functions of the ACMS special finite element method have been used to construct a coarse space for overlapping Schwarz preconditioners. First preliminary results show very good scalability and robustness for highly heterogenous problems. In addition, a heuristic strategy to approximate an ACMS-based coarse space, which avoids the solution of generalized eigenvalue problems, has been presented. For channel problems with high coefficient jumps, first results indicate that the ACMS coarse space and the reconstructed ACMS coarse space show a similar performance as adaptive coarse spaces which are already available. However, an adaptive strategy to select the optimal ACMS basis functions for general coefficient functions with jumps has not been presented here yet.

Future Work

In our FSI simulations, we have only considered idealized geometries and only one material layer so far. Thus, to proceed to more realistic configurations, we will take patient-specific geometries, multiple material layers, and surrounding tissue of the artery into account. In particular, the latter will help to reduce oscillations occurring in our simulations and to replace our artificial boundary conditions. Whereas the computational work for the structural part will increase, the parallel implementation of the GDSW preconditioner will enable us to cope with larger structural problems. In addition to that, the movement induced by the pulse of the heart itself should be included as well to obtain more realistic flow rates and pressure over time.

To further reduce the computation time of our FSI simulations, adaptive time stepping schemes as well as parallel-in-time methods, such as Parareal [148] or

MGRIT (Multigrid Reduction in Time) [93], could be considered. Whereas the use of adaptive time stepping schemes is clearly promising, it is not clear yet if and how parallel-in-time approaches can be applied to FSI problems; cf., e.g., [81].

As a next step, the GDSW preconditioner could be applied to the fluid and the geometry blocks as well, and as a preconditioner for the whole monolithic matrix; cf. [129], where two-level overlapping Schwarz preconditioners were applied to saddle point problems. Therefore, the parallel implementation of the GDSW preconditioner should be improved further, by either implementing reduced coarse spaces [72] or by using inexact solvers for the coarse level, e.g., approximating the inverse of the coarse matrix by an Algebraic Multigrid (AMG) preconditioner or, again, by a GDSW preconditioner, which would lead to a multilevel GDSW preconditioner. With these approaches, it could be possible to further improve the parallel scalability up to a larger number of cores.

It would also be interesting to incorporate the *vertex-specific* (MsFEM) basis functions of the ACMS coarse space and the reconstructed ACMS basis functions into the coarse space of our parallel implementation of the GDSW preconditioner. This would enable the simulations of large highly heterogeneous problems in parallel. The ACMS finite element method should therefore be extended to unstructured decompositions, elastic model problems, and three-dimensional problems; cf. [43], where the multiscale finite element method (MsFEM) was used for three-dimensional elasticity problems. With respect to the ACMS coarse space, an adaptive strategy for the selection of the optimal *edge-based* basis functions and the heuristic strategy for the approximation of the ACMS coarse space should be extended to more general coefficient functions, with and without jumps.

Bibliography

[1] Boost C++ Libraries. `http://www.boost.org`.

[2] Hemolab, 2014. http://hemolab.lncc.br/adan-web. [Accessed on November 2014].

[3] Jørg Aarnes and Thomas Y. Hou. Multiscale domain decomposition methods for elliptic problems with high aspect ratios. Acta Math. Appl. Sin. Engl. Ser., 18(1):63–76, 2002.

[4] James Ahrens, Berk Geveci, and Charles Law. ParaView: An End-User Tool for Large Data Visualization. Visualization Handbook, Elsevier, 2005.

[5] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM J. Matrix Anal. Appl., 23(1):15–41, 2001.

[6] E. Anderson, Z. Bai, C. Bischof, L. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[7] Todd Arbogast. Mixed multiscale methods for heterogeneous elliptic problems. In Ivan G. Graham, Thomas Y. Hou, Omar Lakkis, and Robert Scheichl, editors, Numerical analysis of multiscale problems, volume 83 of Lect. Notes Comput. Sci. Eng., pages 243–283. Springer, Heidelberg, Berlin, Heidelberg, 2012.

[8] Ivo Babuška, Uday Banerjee, and John E. Osborn. On principles for the selection of shape functions for the generalized finite element method. Comput. Methods Appl. Mech. Engrg., 191(49-50):5595–5629, 2002.

[9] Ivo Babuška, Uday Banerjee, and John E. Osborn. Generalized finite element methods - main ideas, results and perspective. International Journal of Computational Methods, 1(1):67–103, 2004.

[10] Ivo Babuška and John E. Osborn. Generalized finite element methods: their performance and their relation to mixed methods. SIAM J. Numer. Anal., 20(3):510–536, 1983.

[11] Santiago Badia, Alberto Martin, and Javier Principe. Multilevel balancing domain decomposition at extreme scales. SIAM J. Sci. Comput. In press, 2015.

[12] Santiago Badia, Annalisa Quaini, and Alfio Quarteroni. Modular vs. non-modular preconditioners for fluid-structure systems with large added-mass effect. Comput. Methods Appl. Mech. Engrg., 197(49-50):4216–4232, 2008.

[13] Santiago Badia, Annalisa Quaini, and Alfio Quarteroni. Splitting methods based on algebraic factorization for fluid-structure interaction. SIAM J. Sci. Comput., 30(4):1778–1805, 2008.

[14] Allison H. Baker, Axel Klawonn, Tzanio Kolev, Martin Lanser, Oliver Rheinbach, and Ulrike Meier Yang. Scalability of classical algebraic multigrid for elasticity to half a million parallel tasks. 2015. Submitted 11/2015 to Lect. Notes Comput. Sci. Eng. TUBAF Preprint: 2015-14, http://tu-freiberg.de/fakult1/forschung/preprints.

[15] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, and Hong Zhang. PETSc Web page, 2015. `http://www.mcs.anl.gov/petsc`.

[16] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015. `http://www.mcs.anl.gov/petsc`.

[17] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, Modern Software Tools in Scientific Computing, pages 163–202. Birkhäuser Press, 1997.

[18] John M. Ball. Convexity conditions and existence theorems in nonlinear elasticity. Arch. Rational Mech. Anal., 63(4):337–403, 1977.

[19] Daniel Balzani. Polyconvex Anisotropic Energies and Modeling of Damage Applied to Arterial Walls. Phd thesis, University Duisburg-Essen, Verlag Glückauf Essen, 2006.

[20] Daniel Balzani, Dirk Böse, Dominik Brands, Raimund Erbel, Axel Klawonn, Oliver Rheinbach, and Jörg Schröder. Parallel simulation of patient-specific atherosclerotic arteries for the enhancement of intravascular ultrasound diagnostics. Engineering Computations, 29(8), 2012.

[21] Daniel Balzani, Dominik Brands, Axel Klawonn, and Oliver Rheinbach. Large-scale simulation of arterial walls: mechanical modeling. PAMM, 7(1):4020017–4020018, 2007. Special Issue: Sixth International Congress on Industrial Applied Mathematics (ICIAM07) and GAMM Annual Meeting, Zürich 2007.

[22] Daniel Balzani, Dominik Brands, Axel Klawonn, Oliver Rheinbach, and Jörg Schröder. On the mechanical modeling of anisotropic biological soft tissue and iterative parallel solution strategies. Archive of Applied Mechanics, 80(5):479–488, 2010.

[23] Daniel Balzani, Simone Deparis, Simon Fausten, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, Oliver Rheinbach, and Jörg Schröder. Aspects of arterial wall simulations: Nonlinear anisotropic material models and fluid structure interaction. In Proceedings of the WCCM XI, number WCCM XI, pages 1–12, 2014.

[24] Daniel Balzani, Simone Deparis, Simon Fausten, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, Oliver Rheinbach, and Jörg Schröder. Numerical modeling of fluidstructure interaction in arteries with anisotropic polyconvex hyperelastic and anisotropic viscoelastic material models at finite strains. International Journal for Numerical Methods in Biomedical Engineering, pages 1–41, 2015.

[25] Daniel Balzani, Friedrich Gruttmann, and Jörg Schröder. Analysis of thin shells using anisotropic polyconvex energy densities. Comput. Methods Appl. Mech. Engrg., 197(9-12):1015–1032, 2008.

[26] Daniel Balzani, Patrizio Neff, Jörg Schröder, and Gerhard A. Holzapfel. A polyconvex framework for soft biological tissues. Adjustment to experimental data. Internat. J. Solids Structures, 43(20):6052–6070, 2006.

[27] Daniel Balzani, Jörg Schröder, Dominik Brands, Axel Klawonn, and Oliver Rheinbach. Computer simulation of damage in overstretched atherosclerotic arteries. Proceedings of ICCB 2007 - III. International Congress on Computational Bioengineering, 2007.

[28] Daniel Balzani, Jörg Schröder, and Dietmar Gross. Simulation of discontinuous damage incorporating residual stresses in circumferentially overstretched atherosclerotic arteries. Acta Biomaterialia, 2(6):609–618, 2006.

[29] Andrew T. Barker. Parallel monolithic fluid-structure interaction algorithms with application to blood flow simulation. PhD thesis, University of Colorado, 2009.

[30] Andrew T. Barker and Xiao-Chuan Cai. Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling. J. Comput. Phys., 229(3):642–659, 2010.

[31] Peter Bastian, C. Engwer, J. Fahlke, M. Geveler, D. Göddeke, Oleg Iliev, O. Ippisch, R. Milk, J. Mohring, S. Müthing, Mario Ohlberger, D. Ribbrock, and Stefan Turek. Advances concerning multiscale methods and uncertainty quantification in EXA-DUNE . In Proceedings of the SPPEXA Symposium 2016. Springer Berlin Heidelberg, 2016. Submitted.

[32] Yuri Bazilevs, V. M. Calo, T. J R Hughes, Y. Zhang, M.-C. Hsu, and M. A. Scott. Isogeometric fluid-structure interaction analysis with emphasis on non-matching discretizations, and with application to wind turbines. Comput. Methods Appl. Mech. Engrg., 249/252(1):28–41, 2012.

[33] Yuri Bazilevs, Kenji Takizawa, and Tayfun E. Tezduyar. Computational Fluid–Structure Interaction. Methods and Applications. Wiley, 2013.

[34] Michele Benzi and Maxim A. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. SIAM J. Sci. Comput., 28(6):2095–2113, 2006.

[35] Michele Benzi, Maxim A. Olshanskii, and Zhen Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. Internat. J. Numer. Methods Fluids, 66(4):486–508, 2010.

[36] Petter E. Bjørstad, Jacko Koster, and Piotr Krzyzanowski. Domain decomposition solvers for large scale industrial finite element problems. In PARA2000 Workshop on Applied Parallel Computing, PARA '00, pages 373–383, London, UK, UK, 2001. Lecture Notes in Computer Science 1947, Springer-Verlag.

[37] Petter E. Bjørstad, Piotr Krzyzanowski, and Piotr Krzy. A flexible 2-level neumann-neumann method for structural analysis problems. In Proceedings of the th International Conference on Parallel Processing and Applied Mathematics-Revised Papers, PPAM '01, pages 387–394, London, UK, UK, 2002. Springer-Verlag.

[38] J.-P. Boehler. Introduction to the invariant formulation of anisotropic constitutive equations. In Applications of tensor functions in solid mechanics, volume 292 of CISM Courses and Lectures, pages 13–30. Springer, Vienna, 1987.

[39] Dirk Böse, Sarah Brinkhues, Raimund Erbel, Axel Klawonn, Oliver Rheinbach, and Jörg Schröder. A simultaneous augmented lagrange approach for the simulation of soft biological tissue. In Randolph Bank, Michael Holst, Olof Widlund, and Jinchao Xu, editors, Domain Decomposition Methods in Science and Engineering XX, volume 91 of Lecture Notes in Computational Science and Engineering, pages 369–376. Springer Berlin Heidelberg, 2013.

[40] Dietrich Braess. Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie, volume 25 of Springer-Lehrbuch Masterclass. Springer Berlin Heidelberg, 2013.

[41] Dominik Brands, Axel Klawonn, Oliver Rheinbach, and Jörg Schröder. Modelling and convergence in arterial wall simulations using a parallel feti solution strategy. Comput. Methods Biomech. Biomed. Engin., 11(5):569–583, 2008.

[42] Sarah Brinkhues, Axel Klawonn, Oliver Rheinbach, and Jörg Schröder. Augmented lagrange methods for quasi-incompressible materials - applications to soft biological tissue. International Journal for Numerical Methods in Biomedical Engineering, 29(3):332–350, 2013.

[43] Marco Buck. Overlapping Domain Decomposition Preconditioners for Multi-Phase Elastic Composites. PhD thesis, Technische Universität Kaiserslautern, 2013.

[44] Marco Buck, Oleg Iliev, and Heiko Andrä. Multiscale finite elements for linear elasticity: Oscillatory boundary conditions. In J Erhal, M J Gander, L Halpern, G Pichot, T Sassi, and O Widlund, editors, Domain Decomposition Methods in Science and Engineering XXI, volume 98 of Lecture Notes in Computational Science and Engineering, pages 237–245. Springer, 2014.

[45] Erik Burman and Miguel Ángel Fernández. Continuous interior penalty finite element method for the time-dependent Navier–Stokes equations: space discretization and convergence. Numerische Mathematik, 107(1):39–77, 2007.

[46] Sunčica Čanić, Craig J. Hartley, Doreen Rosenstrauch, Josip Tambača, Giovanna Guidoboni, and Andro Mikelić. Blood flow in compliant arteries: An effective viscoelastic reduced model, numerics, and experimental validation. Annals of Biomedical Engineering, 34(4):575–592, 2006.

[47] Paola Causin, Jean-Frédéric Gerbeau, and Fabio Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. Comput. Methods Appl. Mech. Engrg., 194(42-44):4506–4527, 2005.

[48] R Chandramouli. Textbook of Physiology. Jaypee Brothers Medical Publishers, 3rd edition, 2010.

[49] Eric T. Chung and Hyea Hyun Kim. A deluxe FETI-DP algorithm for a hybrid staggered discontinuous Galerkin method for H(curl)-elliptic problems. Internat. J. Numer. Methods Engrg., 98(1):1–23, 2014.

[50] Philippe G. Ciarlet. Mathematical elasticity. volume I. , Three-dimensional elasticity, volume I of Studies in mathematics and its applications. North-Holland, Amsterdam, New York, 1988.

[51] C.M. Colciago, Simone Deparis, and Alfio Quarteroni. Comparisons between reduced order models and full 3d models for fluidstructure interaction problems in haemodynamics. Journal of Computational and Applied Mathematics, 265:120–138, 2014.

[52] P. Colli Franzone, Luca F. Pavarino, and Simone Scacchi. Parallel multilevel solvers for the cardiac electro-mechanical coupling. Appl. Numer. Math., 95:140–153, 2015.

[53] Georges-Henri Cottet, Emmanuel Maitre, and Thomas Milcent. Eulerian formulation and level set models for incompressible fluid-structure interaction. M2AN Math. Model. Numer. Anal., 42:471–492, 2008.

[54] Roy R. Craig Jr. and Mervyn C. C. Bampton. Coupling of substructures for dynamic analyses. AIAA Journal, 6(7):1313–1319, 1968. doi: 10.2514/3.4741.

[55] Jean-Michel Cros. A preconditioner for the Schur complement domain decomposition method. In O Widlund I. Herrera D. Keyes and R Yates, editors, Domain Decomposition Methods in Science and Engineering, pages 373–380. National Autonomous University of Mexico (UNAM), Mexico City, Mexico, ISBN 970-32-0859-2, 2003. Proc. 14th Int. Conf. Domain Decomposition Methods; http://www.ddm.org/DD14.

[56] Paolo Crosetto, Simone Deparis, Gilles Fourestey, and Alfio Quarteroni. Parallel algorithms for fluid-structure interaction problems in haemodynamics. SIAM J. Sci. Comput., 33(4):1598–1622, 2011.

[57] Paolo Crosetto, Philippe Reymond, and Simone Deparis. Fluidstructure interaction simulation of aortic blood flow. Computers & Fluids, 43(1):46–57, 2011.

[58] Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software, 30(2):165–195, 2004.

[59] Timothy A. Davis and Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. SIAM J. Matrix Anal. Appl., 18(1):140–158, 1997.

[60] Joris Degroote, Klaus-Jürgen Bathe, and Jan Vierendeels. Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction. Comput. Struct., 87(11-12):793–801, 2009. Fifth MIT Conference on Computational Fluid and Solid Mechanics.

[61] Joris Degroote and Jan Vierendeels. Multi-level quasi-newton coupling algorithms for the partitioned simulation of fluidstructure interaction. Comput. Methods Appl. Mech. Engrg., 225228(0):14–27, 2012.

[62] Simone Deparis. Numerical analysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation. PhD thesis, SB, Lausanne, 2004.

[63] Simone Deparis, Marco Discacciati, Gilles Fourestey, and Alfio Quarteroni. Fluid-structure algorithms based on Steklov-Poincaré operators. Comput. Methods Appl. Mech. Engrg., 195(41-43):5797–5812, 2006.

[64] Simone Deparis, Marco Discacciati, and Alfio Quarteroni. A domain decomposition framework for fluid-structure interaction problems. In Clinton Groth and David W. Zingg, editors, Computational Fluid Dynamics 2004, pages 41–58. Springer Berlin Heidelberg, 2006.

[65] Simone Deparis, Davide Forti, Gwenol Grandperrin, and Alfio Quarteroni. FaCSI: A block parallel preconditioner for fluid-structure interaction in hemodynamics. Technical report, MATHICSE, 2015.

[66] Simone Deparis, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, and Oliver Rheinbach. A Comparison of Preconditioners for the Steklov-Poincaré Formulation of the Fluid-Structure Coupling in Hemodynamics. PAMM, 15(1):93–94, 2015.

[67] Simone Deparis, Davide Forti, and Alfio Quarteroni. A rescaled localized radial basis function interpolation on non-Cartesian and nonconforming grids. SIAM J. Sci. Comput., 36(6):A2745–A2762, 2014.

[68] Simone Deparis, Gwenol Grandperrin, and Alfio Quarteroni. Parallel preconditioners for the unsteady Navier-Stokes equations and applications to hemodynamics simulations. Comput. & Fluids, 92:253–273, 2014.

[69] Clark R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. SIAM J. Sci. Comput., 25(1):246–258, 2003.

[70] Clark R. Dohrmann, Axel Klawonn, and Olof B. Widlund. Domain decomposition for less regular subdomains: overlapping Schwarz in two dimensions. SIAM J. Numer. Anal., 46(4):2153–2168, 2008.

[71] Clark R. Dohrmann, Axel Klawonn, and Olof B. Widlund. A family of energy minimizing coarse spaces for overlapping schwarz preconditioners. In Domain decomposition methods in science and engineering XVII, volume 60 of Lect. Notes Comput. Sci. Eng., pages 247–254. Springer, Berlin, 2008.

[72] Clark R. Dohrmann and Olof B. Widlund. Hybrid domain decomposition algorithms for compressible and almost incompressible elasticity. Internat. J. Numer. Methods Engrg., 82(2):157–183, 2010.

[73] Clark R. Dohrmann and Olof B. Widlund. Some recent tools and a bddc algorithm for 3d problems in h(curl). Lecture Notes in Computational Science and Engineering, 91:15–25, 2013.

[74] Maksymilian Dryja, Barry F. Smith, and Olof B. Widlund. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. SIAM J. Numer. Anal., 31(6):1662–1694, 1994.

[75] Thomas Dunne. Adaptive Finite Element Approximation of Fluid-Structure Interaction Based on Eulerian and Arbitrary Lagrangian-Eulerian Variational Formulations. PhD thesis, Ruprechts-Karls Universität Heidelberg, 2007.

[76] Weinan E, Bjorn Engquist, Xiantao Li, Weiqing Ren, Eric Vanden-Eijnden, E Weinan, Bjorn Engquist, Xiantao Li, Weiqing Ren, Weinan E, Bjorn Engquist, Xiantao Li, Weiqing Ren, and Eric Vanden-Eijnden. The

heterogeneous multiscale method: A review. Commun. Comput. Phys., 2(3):367–450, 2007.

[77] Yalchin Efendiev and Thomas Y. Hou. Multiscale finite element methods, volume 4 of Surveys and Tutorials in the Applied Mathematical Sciences. Springer, New York, 2009. Theory and applications.

[78] Howard Elman, Victoria E. Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. Block preconditioners based on approximate commutators. SIAM J. Sci. Comput., 27(5):1651–1668, 2006.

[79] Howard Elman, Victoria E. Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. J. Comput. Phys., 227(3):1790–1808, 2008.

[80] Howard Elman, Victoria E. Howle, John Shadid, David Silvester, and Ray Tuminaro. Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations. SIAM J. Sci. Comput., 30(1):290–311, 2007.

[81] Charbel Farhat, Julien Cortial, Climène Dastillung, and Henri Bavestrello. Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. Internat. J. Numer. Methods Engrg., 67(5):697–724, 2006.

[82] Charbel Farhat, Michael Lesoinne, and Kendall Pierson. A scalable dual-primal domain decomposition method. Numer. Linear Algebra Appl., 7(7-8):687–714, 2000. Preconditioning techniques for large sparse matrix problems in industrial applications (Minneapolis, MN, 1999).

[83] Charbel Farhat, Michel Lesoinne, Patrick LeTallec, Kendall Pierson, and Daniel Rixen. FETI-DP: a dual-primal unified FETI method. I. A faster alternative to the two-level FETI method. Internat. J. Numer. Methods Engrg., 50(August 1999):1523–1544, 2001.

[84] Simon Fausten, Daniel Balzani, and Jörg Schröder. Modeling the physiological behavior of arterial walls - comparative study regarding the viscoelastic response. Proceedings of Applied Mathematics and Mechanics, 14:95–96, 2014.

[85] Miguel Ángel Fernández, Luca Formaggia, Jean-Frédéric Gerbeau, and Alfio Quarteroni. The derivation of the equations for fluids. In Cardiovascular Mathematics, pages 77–121. Springer, 2009.

[86] Miguel Ángel Fernández, Jean-Frédéric Gerbeau, Antoine Gloria, and Marina Vidrascu. Domain decomposition based Newton methods for fluid-structure interaction problems. In CANUM 2006—Congrès National d'Analyse Numérique, volume 22 of ESAIM Proc., pages 67–82. EDP Sci., Les Ulis, 2008.

[87] Miguel Ángel Fernández and M Moubachir. An exact block-Newton algorithm for solving fluid-structure interaction problems. C. R. Math. Acad. Sci. Paris, 336(8):681–686, 2003.

[88] Miguel Ángel Fernández and Marwan Moubachir. A newton method using exact jacobian for solving fluid-structure coupling. Computers & Structures, 83(2-3):127–142, 2005.

[89] Andreas Fischle. A Parallel Newton-Krylov-FETI-DP Solver Based on FEAP. PhD thesis, Fakultät für Mathematik, Universität Duisburg-Essen, 2014.

[90] Luca Formaggia, Miguel Ángel Fernández, A Gauthier, Jean-Frédéric Gerbeau, Christophe Prud'homme, and Alessandro Veneziani. The LifeV Project. Web. `http://www.lifev.org`.

[91] Luca Formaggia, Alfio Quarteroni, and Alessandro Veneziani, editors. Cardiovascular mathematics, volume 1 of MS&A. Modeling, Simulation and Applications. Springer-Verlag Italia, Milan, 2009. Modeling and simulation of the circulatory system.

[92] Gilles Fourestey and Simone Deparis. LifeV user manual. pages 0–32, 2012. `http://www.lifev.org`.

[93] S Friedhoff, R D Falgout, T V Kolev, Scott P MacLachlan, and Jacob B Schroder. A multigrid-in-time algorithm for solving evolution equations in parallel. In Presented at: Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, Mar 17 - Mar 22, 2013, 2013.

[94] Juan Galvis and Yalchin Efendiev. Domain decomposition preconditioners for multiscale flows in high-contrast media. Multiscale Model. Simul., 8(4):1461–1483, 2010.

[95] Juan Galvis and Yalchin Efendiev. Domain decomposition preconditioners for multiscale flows in high contrast media: reduced dimension coarse spaces. Multiscale Model. Simul., 8(5):1621–1644, 2010.

[96] Michael W. Gee, Christiane Förster, and Wolfgang A. Wall. A computational strategy for prestressing patient-specific biomechanical problems under finite deformation. Int. J. Numer. Methods Biomed. Eng., 26:52–72, 2010.

[97] Michael W. Gee, Ulrich Küttler, and Wolfgang A. Wall. Truly monolithic algebraic multigrid for fluid-structure interaction. Internat. J. Numer. Methods Engrg., 26:52–72, 2010.

[98] Michael W. Gee, Chris M. Siefert, Jonathan J. Hu, Ray S. Tuminaro, and Marzio G. Sala. ML 5.0 Smoothed Aggregation User's Guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[99] Jean-Frédéric Gerbeau and Marina Vidrascu. A quasi-Newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows. M2AN Math. Model. Numer. Anal., 37(4):631–647, 2003.

[100] Sabrina Gippert, Axel Klawonn, and Oliver Rheinbach. Analysis of FETI-DP and BDDC for linear elasticity in 3D with almost incompressible components and varying coefficients inside subdomains. SIAM J. Numer. Anal., 50(5):2208–2236, 2012.

[101] Pierre Gosselet, Daniel Rixen, François-Xavier Roux, and Nicole Spillane. Simultaneous FETI and block FETI: Robust domain decomposition with multiple search directions. International Journal for Numerical Methods in Engineering, 104(10):905–927, 2015.

[102] Ibrahima Guèye, S El Arem, Fédéric Feyel, François-Xavier Roux, and Georges Cailletaud. A new parallel sparse direct solver: Presentation and numerical experiments in large-scale structural mechanics parallel computing. Internat. J. Numer. Meth. Engrg, 88(4):370–384, 2011.

[103] Alexander Heinlein, Ulrich L. Hetmaniuk, Axel Klawonn, and Oliver Rheinbach. The approximate component mode synthesis special finite element method in two dimensions: parallel implementation and numerical results. J. Comput. Appl. Math., 289:116–133, 2015. Sixth International Conference on Advanced Computational Methods in Engineering (ACOMEN 2014).

[104] Alexander Heinlein, Axel Klawonn, Jascha Knepper, and Oliver Rheinbach. Coarse spaces for overlapping Schwarz methods based on the ACMS space. 2016. In preparation.

[105] Alexander Heinlein, Axel Klawonn, and Oliver Rheinbach. Parallel overlapping Schwarz with an energy-minimizing coarse space. 2015. Submitted to the Proceedings of the 23rd International Conference on Domain Decomposition Methods, Springer Lect. Notes Comput. Sci. Eng.; `http://tu-freiberg.de/fakult1/forschung/preprints`.

[106] Alexander Heinlein, Axel Klawonn, and Oliver Rheinbach. Parallel two-level overlapping Schwarz methods in fluid-structure interaction. 2015. Accepted to Springer Lect. Notes Sci. Comput.; Proceedings of ENUMATH 2015; TUBAF Preprint 15/2015: `http://tu-freiberg.de/fakult1/forschung/preprints`.

[107] Alexander Heinlein, Axel Klawonn, and Oliver Rheinbach. A parallel implementation of a two-level overlapping schwarz method with energy-minimizing coarse space based on trilinos. 2016. In preparation.

[108] Amy Henderson and Jim Ahrens. The Paraview guide : a parallel visualization application. Kitware, Inc., New York, 2004.

[109] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the trilinos project. ACM Trans. Math. Softw., 31(3):397–423, 2005.

[110] Ulrich L. Hetmaniuk and Axel Klawonn. Error estimates for a two-dimensional special finite element method based on component mode synthesis. Electron. Trans. Numer. Anal., 41:109–132, 2014.

[111] Ulrich L. Hetmaniuk and Richard B. Lehoucq. A special finite element method based on component mode synthesis. M2AN Math. Model. Numer. Anal., 44(3):401–420, 2010.

[112] Gerhard A. Holzapfel and Thomas Christian Gasser. A viscoelastic model for fiber-reinforced composites at finite strains: Continuum basis, computational aspects and applications. Computer Methods in Applied Mechanics and Engineering, 190(34):4379–4403, 2001.

[113] Gerhard A. Holzapfel, Thomas Christian Gasser, and Ray W. Ogden. A new constitutive framework for arterial wall mechanics and a comparative study of material models. Journal of elasticity and the physical science of solids, 61(1):1–48, 2000.

[114] Thomas Y. Hou and Xiao-Hui Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. J. Comput. Phys., 134(1):169–189, 1997.

[115] Jaroslav Hron and Stefan Turek. A monolithic fem/multigrid solver for ALE formulation of fluid structure interaction with application in biomechanics. In H Bungartz and M Schäfer, editors, Fluid-Structure Interaction - Modelling, Simulation, Optimization, volume 53 of Lecture Notes in Computational Science and Engineering, pages 146–170. Springer, 2006. ISBN 3-540-34595-7.

[116] Björn Hübner, Elmar Walhorn, and Dieter Dinkler. A monolithic approach to fluidstructure interaction using spacetime finite elements. Computer Methods in Applied Mechanics and Engineering, 193(23-26):2087–2104, 2004.

[117] Walter C. Hurty. Vibrations of structural systems by component mode synthesis. Journal of the Engineering Mechanics Division, 86(4):51–70, 1960.

[118] Walter C. Hurty. Dynamic analysis of structural systems using component modes. AIAA journal, 3(4):678–685, 1965.

[119] Mikhail Itskov and Alexander E. Ehret. A universal model for the elastic, inelastic and active behaviour of soft biological tissues. GAMM-Mitteilungen, 32(2):221–236, 2009.

[120] Bärbel Janssen and Thomas Wick. Block preconditioning with Schur complements for monolithic fluid-structure interactions. In Proceedings of V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010, Lisbon, Portugal,, 2010.

[121] Pierre Jolivet. Domain decomposition methods. Application to high-performance computing. PhD thesis, Université de Grenoble, 2014.

[122] Pierre Jolivet, Frédéric Hecht, Frédéric Nataf, and Christophe Prud'homme. Scalable domain decomposition preconditioners for heterogeneous elliptic problems. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13, pages 80:1–80:11, New York, NY, USA, 2013. ACM.

[123] George Karypis, Kirk Schloegel, and Vipin Kumar. ParMETIS - Parallel graph partitioning and sparse matrix ordering. Version 3.2. Technical report, University of Minnesota, Department of Computer Science and Engineering, 2011.

[124] Axel Klawonn, Martin Lanser, Patrick Radtke, and Oliver Rheinbach. On an adaptive coarse space and on nonlinear domain decomposition. In Jocelyne Erhel, Martin J. Gander, Laurence Halpern, Géraldine Pichot, Taoufik Sassi, and Olof B Widlund, editors, Domain Decomposition Methods in Science and Engineering XXI, volume 98 of Lect. Notes Comput. Sci. Eng., pages 71–83. Springer-Verlag, 2014. Proceedings of the 21st International Conference on Domain Decomposition Methods, Rennes, France, June 25-29, 2012.

[125] Axel Klawonn, Martin Lanser, and Oliver Rheinbach. Nonlinear FETI-DP and BDDC methods. SIAM J. Sci. Comput., 36(2):A737–A765, 2014.

[126] Axel Klawonn, Martin Lanser, and Oliver Rheinbach. FE^{2}TI: Computational scale bridging for dual-phase steels. 2015. Accepted to ParCo 2015. TUBAF Preprint: 2015-12, http://tu-freiberg.de/fakult1/forschung/preprints.

[127] Axel Klawonn, Martin Lanser, and Oliver Rheinbach. Toward Extremely Scalable Nonlinear Domain Decomposition Methods for Elliptic Partial Differential Equations. SIAM J. Sci. Comput., 37(6):C667—-C696, 2015.

[128] Axel Klawonn, Martin Lanser, Oliver Rheinbach, Holger Stengel, and Gerhard Wellein. Hybrid MPI/OpenMP parallelization in FETI-DP methods. In Miriam Mehl, Manfred Bischoff, and Michael Schäfer, editors, Recent Trends in Computational Engineering - CE2014, volume 105 of Lecture Notes in Computational Science and Engineering, pages 67–84. Springer International Publishing, 2015.

[129] Axel Klawonn and Luca F. Pavarino. Overlapping Schwarz methods for mixed linear elasticity and Stokes problems. Comput. Methods Appl. Mech. Engrg., 165(1-4):233–245, 1998.

[130] Axel Klawonn, Patrick Radtke, and Oliver Rheinbach. FETI-DP with different scalings for adaptive coarse spaces. PAMM, 14(1):835–836, 2014.

[131] Axel Klawonn, Patrick Radtke, and Oliver Rheinbach. A comparison of adaptive coarse spaces for iterative substructuring methods in two dimensions. Submitted for publication to ETNA, 2015.

[132] Axel Klawonn, Patrick Radtke, and Oliver Rheinbach. FETI-DP Methods with an Adaptive Coarse Space. SIAM J. Numer. Anal., 53(1):297–320, 2015.

[133] Axel Klawonn and Oliver Rheinbach. Inexact FETI-DP methods. Internat. J. Numer. Methods Engrg., 69(2):284–307, 2007.

[134] Axel Klawonn and Oliver Rheinbach. Robust FETI-DP methods for heterogeneous three dimensional elasticity problems. Comput. Methods Appl. Mech. Engrg., 196(8):1400–1414, 2007.

[135] Axel Klawonn and Oliver Rheinbach. A hybrid approach to 3-level FETI. PAMM, 8(1):10841–10843, 2008.

[136] Axel Klawonn and Oliver Rheinbach. Highly scalable parallel domain decomposition methods with an application to biomechanics. ZAMM Z. Angew. Math. Mech., 90(1):5–32, 2010.

[137] Axel Klawonn and Oliver Rheinbach. Deflation, projector preconditioning, and balancing in iterative substructuring methods: connections and new results. SIAM J. Sci. Comput., 34(1):A459–A484, 2012.

[138] Axel Klawonn and Olof B. Widlund. Dual-primal FETI methods for linear elasticity. Comm. Pure Appl. Math., 59(11):1523–1572, 2006.

[139] Axel Klawonn, Olof B. Widlund, and Maksymilian Dryja. Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. SIAM J. Numerical Analysis, 40(1):159–179, 2002.

[140] Jascha Knepper. Master's thesis. 2016. In preparation.

[141] A. Kolmogoroff. Uber Die Beste Annaherung Von Funktionen Einer Gegebenen Funktionenklasse. Annals of Mathematics, 37(1):107–110, 1936.

[142] Fande Kong and Xiao-Chuan Cai. A scalable Schwarz method for 3D linear elasticity problems on domains with complex geometry, 2013. Poster presented at The International Conference for High Performance Computing, Networking, Storage and Analysis (SC13), Denver, Colorado, USA, 2013. `http://sc13.supercomputing.org/sites/default/files/PostersArchive/post147.html`.

[143] Ulrich Küttler, Michael W. Gee, Christiane Förster, Andrew Comerford, and Wolfgang A. Wall. Coupling strategies for biomedical fluidstructure

interaction problems. International Journal for Numerical Methods in Biomedical Engineering, 26(3-4):305–321, 2010.

[144] Ulrich Langer and Huidong Yang. Domain decomposition solvers for some fluid-structure interaction problems. PAMM, 12(1):375–376, 2012.

[145] Ulrich Langer and Huidong Yang. Partitioned solution algorithms for fluid-structure interaction problems with hyperelastic models. J. Comput. Appl. Math., 276:47–61, 2015.

[146] Jing Li and Olof B. Widlund. FETI-DP, BDDC, and block Cholesky methods. Internat. J. Numer. Methods Engrg., 66(2):250–271, 2006.

[147] Florian Lindner, Miriam Mehl, Klaudius Scheufele, and Benjamin Uekermann. A comparison of various quasi-newton schemes for partitioned fluid-structure interaction. In Coupled Problems. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, ECCOMAS, 2015.

[148] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. A parareal in time discretization of PDEs. Comptes Rendus de l'Académie des Sciences - Series I - Mathematics, 332:661–668, 2001.

[149] Jan Mandel and Clark R. Dohrmann. Convergence of a balancing domain decomposition by constraints and energy minimization. Numer. Linear Algebra Appl., 10(7):639–659, 2003. Dedicated to the 70th birthday of Ivo Marek.

[150] Jan Mandel, Clark R Dohrmann, and Radek Tezaur. An algebraic theory for primal and dual substructuring methods by constraints. Appl. Numer. Math., 54(2):167–193, 2005.

[151] Jan Mandel and Bedřich Sousedík. Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods. Comput. Methods Appl. Mech. Engrg., 196(8):1389–1399, 2007.

[152] Jan Mandel, Bedřich Sousedík, and Clark R. Dohrmann. Multispace and multilevel BDDC. Computing, 83(2-3):55–85, 2008.

[153] Jan Mandel, Bedřich Sousedík, and Jakub Šístek. Adaptive BDDC in three dimensions. Math. Comput. Simulation, 82(10):1812–1831, 2012.

[154] Matthias Mayr, Thomas Klöppel, Wolfgang A. Wall, and Michael W. Gee. A temporal consistent monolithic approach to fluid-structure interaction enabling single field predictors. SIAM J. Sci. Comput., 37(1):B30—-B59, 2015.

[155] D. Mohrman and L. J. Heller. Cardiovascular Physiology. McGraw Hill Professional, 8th edition, 2013.

[156] M. Munteanu, Luca F. Pavarino, and Simone Scacchi. A scalable Newton-Krylov-Schwarz method for the bidomain reaction-diffusion system. SIAM J. Sci. Comput., 31(5):3861–3883, 2009.

[157] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems. SIAM J. Sci. Comput., 21(6):1969–1972, 2000.

[158] Fabio Nobile and Christian Vergara. An effective fluid-structure interaction formulation for vascular dynamics by generalized Robin conditions. SIAM J. Sci. Comput., 30(2):731–763, 2008.

[159] James Nolen, George Papanicolaou, and Olivier Pironneau. A framework for adaptive multiscale methods for elliptic problems. Multiscale Model. Simul., 7(1):171–196, 2008.

[160] Suhas V. Patankar and Dudley Brian Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. International Journal of Heat and Mass Transfer, 15(10):1787–1806, 1972.

[161] M. Pernice and M. D. Tocci. A multigrid-preconditioned Newton-Krylov method for the incompressible Navier-Stokes equations. SIAM J. Sci. Comput., 23(2):398–418, 2001. Copper Mountain Conference (2000).

[162] Charles S. Peskin. The immersed boundary method. Acta Numer., 11:479–517, 2002.

[163] Allan Pinkus. n-widths in approximation theory, volume 7 of Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]. Springer-Verlag, Berlin, Berlin, Heidelberg, New York, 1985.

[164] Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani. Factorization methods for the numerical approximation of Navier-Stokes equations. Comput. Methods Appl. Mech. Engrg., 188(1-3):505–526, 2000.

[165] Patrick Radtke. Adaptive Coarse Spaces for FETI-DP and BDDC Methods. PhD thesis, 2015.

[166] Rolf Rannacher and Thomas Richter. An adaptive finite element method for fluid-structure interaction problems based on a fully eulerian formulation. In Hans-Joachim Bungartz, Miriam Mehl, and Michael Schäfer, editors, Fluid Structure Interaction II, volume 73 of Lecture Notes in Computational Science and Engineering, pages 159–191. Springer Berlin Heidelberg, 2010.

[167] Mudassar Razzaq, Hogenrich Damanik, Jaroslav Hron, Abderrahim Ouazzi, and Stefan Turek. FEM multigrid techniques for fluid-structure interaction with application to hemodynamics. Appl. Numer. Math., 62(9):1156–1170, 2012.

[168] Mudassar Razzaq, Jaroslav Hron, and Stefan Turek. Numerical simulation of laminar incompressible fluid-structure interaction for elastic material with point constraints. In Advances in mathematical fluid mechanics, pages 451–472. Springer, Berlin, Berlin, 2010.

[169] Oliver Rheinbach. Parallel iterative substructuring in structural mechanics. Arch. Comput. Methods Eng., 16(4):425–463, 2009.

[170] Thomas Richter. A fully eulerian formulation for fluid-structure-interaction problems with large deformations and free structure movement. In Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010, Lisbon, Portugal, 14-17 June 2010, 2010.

[171] Yousef Saad. Iterative Methods for Sparse Linear Systems. Engineering-Pro collection. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[172] Phillip A. Sackinger, Peter Randall Schunk, and Rekha R. Rao. A Newton-Raphson pseudo-solid domain mapping technique for free and moving boundary problems: a finite element implementation. J. Comput. Phys., 125(1):83–103, 1996.

[173] Marzio G. Sala and Michael A. Heroux. Robust algebraic preconditioners with IFPACK 3.0. Technical Report SAND-0662, Sandia National Laboratories, 2005.

[174] Carlo Sansour. On the physical assumptions underlying the volumetric-isochoric split and the case of anisotropy. European Journal of Mechanics - A/Solids, 27(1):28–39, 2008.

[175] Simone Scacchi. A hybrid multilevel Schwarz method for the bidomain model. Comput. Methods Appl. Mech. Engrg., 197(45-48):4051–4061, 2008.

[176] Simone Scacchi. A multilevel hybrid Newton-Krylov-Schwarz method for the Bidomain model of electrocardiology. Computer Methods in Applied Mechanics and Engineering, 200(58):717–725, 2011.

[177] Boris Schling. The Boost C++ Libraries. XML Press, 2011.

[178] Jörg Schröder and Patrizio Neff. Invariant formulation of hyperelastic transverse isotropy based on polyconvex free energy functions. Internat. J. Solids Structures, 40(2):401–445, 2003.

[179] Jörg Schröder, Patrizio Neff, and Daniel Balzani. A variational approach for materially stable anisotropic hyperelasticity. Internat. J. Solids Structures, 42(15):4352–4371, 2005.

[180] Hermann Amandus Schwarz. Gesammelte mathematische Abhandlungen. Number 2. Springer Berlin, 15 edition, 1890.

[181] Chris M. Siefert. (Developer of the Trilinos packages `ML`, `MueLu`, `Ifpack`, `Ifpack2` and `EpetraExt`). Private communication May 2015, 2015.

[182] Juan Carlos Simo. On a fully three-dimensional finite-strain viscoelastic damage model: Formulation and computational aspects. Comput. Methods Appl. Mech. Engrg., 60(2):153–173, 1987.

[183] Juan Carlos Simo. Numerical analysis and simulation of plasticity. In Philippe G. Ciarlet and Jacques-Louis Lions, editors, Handbook of numerical analysis, volume VI. Elsevier Science, 1998.

[184] Kathrin Smetana. A new certification framework for the port reduced static condensation reduced basis element method. Computer Methods in Applied Mechanics and Engineering, 283:352–383, 2015.

[185] Kathrin Smetana and Anthony T. Patera. Optimal local approximation spaces for component-based static condensation procedures. Technical report, 2015.

[186] Barry F. Smith, Petter E. Bjørstad, and William D. Gropp. Domain decomposition. Cambridge University Press, Cambridge, 1996. Parallel multilevel methods for elliptic partial differential equations.

[187] Bedřich Sousedík, Jakub Šístek, and Jan Mandel. Adaptive-multilevel BDDC and its parallel implementation. Computing, 95(12):1087–1119, 2013.

[188] Nicole Spillane, Victorita Dolean, Patrice Hauret, Frédéric Nataf, Clemens Pechstein, and Robert Scheichl. A robust two-level domain decomposition preconditioner for systems of PDEs. C. R. Math. Acad. Sci. Paris, 349(23-24):1255–1259, 2011.

[189] Nicole Spillane, Victorita Dolean, Patrice Hauret, Frédéric Nataf, Clemens Pechstein, and Robert Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. Numer. Math., 126(4):741–770, 2014.

[190] Nicole Spillane and Daniel J. Rixen. Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms. Internat. J. Numer. Methods Engrg., 95(11):953–990, 2013.

[191] Michael Stephan and Jutta Docter. JUQUEEN: IBM Blue Gene/Q Supercomputer System at the Jülich Supercomputing Centre. Journal of large-scale research facilities, 1:A1, 2015.

[192] Masato Tanaka, Masaki Fujikawa, Daniel Balzani, and Jörg Schröder. Robust numerical calculation of tangent moduli at finite strains based on complex-step derivative approximation and its application to localization analysis. Comput. Methods Appl. Mech. Engrg., 269:454–470, 2014.

[193] Robert L Taylor. FEAP - A Finite Element Analysis Program, Version 8.2. `http://www.ce.berkeley.edu/projects/feap/`.

[194] Robert L. Taylor. FEAP - A Finite Element Analysis Program, Version 8.2, User Manual. `http://www.ce.berkeley.edu/projects/feap`, 2008.

[195] Tayfun E. Tezduyar, Sunil Sathe, and Keith Stein. Solution techniques for the fully discretized equations in computation of fluid-structure interactions with the space-time formulations. Comput. Methods Appl. Mech. Engrg., 195(41-43):5743–5753, 2006.

[196] The HDF Group. Hierarchical data format version 5. `http://www.hdfgroup.org/HDF5`.

[197] Andrea Toselli and Olof Widlund. Domain decomposition methods—algorithms and theory, volume 34 of Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 2005.

[198] Paolo Tricerri, Luca Dedè, Simone Deparis, Alfio Quarteroni, Anne M. Robertson, and Adélia Sequeira. Fluid-structure interaction simulations of cerebral arteries modeled by isotropic and anisotropic constitutive laws. Comput. Mech., 55(3):479–498, 2015.

[199] Xuemin Tu. Three-level BDDC in three dimensions. SIAM J. Sci. Comput., 29(4):1759–1780 (electronic), 2007.

[200] Stefan Turek, Jaroslav Hron, Mudassar Razzaq, Hilmar Wobker, and Michael Schäfer. Numerical benchmarking of fluid-structure interaction: a comparison of different discretization and solution approaches. In Hans-Joachim Bungartz, Miriam Mehl, and Michael Schäfer, editors, Fluid structure interaction. II, volume 73 of Lect. Notes Comput. Sci. Eng., pages 413–424. Springer, Heidelberg, 2010.

[201] Mehfooz ur Rehman, Kees Vuik, and Guus Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. Internat. J. Numer. Methods Fluids, 57(12):1731–1751, 2008.

[202] Wolfgang A. Wall, Axel Gerstenberger, Ulrich Küttler, and Ursula M. Mayer. An XFEM based fixed-grid approach for 3D fluid-structure interaction. In Fluid structure interaction. II, volume 73 of Lect. Notes Comput. Sci. Eng., pages 327–349. Springer, Heidelberg, 2010.

[203] Hongwu Wang, Jack Chessa, Wing Kam Liu, and Ted Belytschko. The immersed/fictitious element method for fluidstructure interaction: Volumetric consistency, compressibility and thin members. Internat. J. Numer. Meth. Engrg., 74(1):32–55, 2008.

[204] Sansuke M. Watanabe. ADAN: Um Modelo Anatomicamente Detalhado da Rede Arterial Humana para Hemodinâmica Computacional. PhD thesis, 2013.

[205] WHO. Cardiovascular diseases (CVDs) Fact sheet N°317. Updated January 2015. `http://www.who.int/mediacentre/factsheets/fs317/en/`. [Retrieved January 26, 2016].

[206] Yuqi Wu and Xiao-Chuan Cai. A parallel two-level method for simulating blood flows in branching arteries with the resistive boundary condition. Computers & Fluids, 45(1):92–102, 2011. 22nd International Conference on Parallel Computational Fluid Dynamics (ParCFD 2010)ParCFD.

[207] Yuqi Wu and Xiao-Chuan Cai. A fully implicit domain decomposition based ALE framework for three-dimensional fluid-structure interaction with application in blood flow computation. J. Comput. Phys., 258:524–537, 2014.

[208] Huidong Yang, Walter Zulehner, and Michael Kuhn. A newton based fluidstructure interaction solver with algebraic multigrid methods on hybrid meshes. In Yunqing Huang, Ralf Kornhuber, Olof Widlund, and Jinchao Xu, editors, Domain Decomposition Methods in Science and Engineering XIX, volume 78 of Lecture Notes in Computational Science and Engineering, pages 285–292. Springer Berlin Heidelberg, 2011.

[209] Olgierd C. Zienkiewicz, Robert L. Taylor, and David D. Fox. The Finite Element Method for Solid and Structural Mechanics. Elsevier/Butterworth Heinemann, Amsterdam, seventh edition, 2014.

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie - abgesehen von unten angegebenen Teilpublikationen - noch nicht veröffentlicht worden ist, sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde.

Die Bestimmungen der Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Axel Klawonn betreut worden.

Teilpublikationen

- Daniel Balzani, Simone Deparis, Simon Fausten, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, Oliver Rheinbach, and Jörg Schröder. Aspects of Arterial Wall Simulations: Nonlinear Anisotropic Material Models and Fluid Structure Interaction, Proceedings of the WCCM XI, 2014.
- Alexander Heinlein, Ulrich Hetmaniuk, Axel Klawonn, and Oliver Rheinbach. The approximate component mode synthesis special finite element method in two dimensions: parallel implementation and numerical results, J. Comput. Appl. Math., 289:116–133, 2015, Sixth International Conference on Advanced Computational Methods in Engineering (ACOMEN 2014).
- Simone Deparis, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, and Oliver Rheinbach. A Comparison of Preconditioners for the Steklov-Poincare Formulation of the Fluid-Structure Coupling in Hemodynamics. PAMM, 15(1):93–94, 2015.
- Daniel Balzani, Simone Deparis, Simon Fausten, Davide Forti, Alexander Heinlein, Axel Klawonn, Alfio Quarteroni, Oliver Rheinbach, and Jörg Schrder. Numerical Modeling of Fluid-Structure Interaction in Arteries with Anisotropic Polyconvex Hyperelastic and Anisotropic Viscoelastic Material Models at Finite Strains. IJNMBE, pages 1–41, 2015.
- Alexander Heinlein, Axel Klawonn, and Oliver Rheinbach. Parallel Two-Level Overlapping Schwarz Methods in Fluid-Structure Interaction. 2016. Accepted to Springer Lect. Notes Sci. Comput.; Proceedings of ENUMATH 2015.
- Alexander Heinlein, Axel Klawonn, and Oliver Rheinbach. Parallel Overlapping Schwarz with an Energy-Minimizing Coarse Space. 2016. Submitted to the Proceedings of the 23rd International Conference on Domain Decomposition Methods, Springer Lect. Notes Comput. Sci. Eng.
- Alexander Heinlein, Axel Klawonn, and Oliver Rheinbach. A Parallel Implementation of a Two-Level Overlapping Schwarz Method with Energy-Minimizing Coarse Space based on Trilinos. 2016. In preparation.
- Alexander Heinlein, Axel Klawonn, and Jascha Knepper, and Oliver Rheinbach. Coarse spaces for overlapping Schwarz methods based on the ACMS space. 2016. In preparation.

Köln, den 06.06.2016 (Alexander Heinlein)