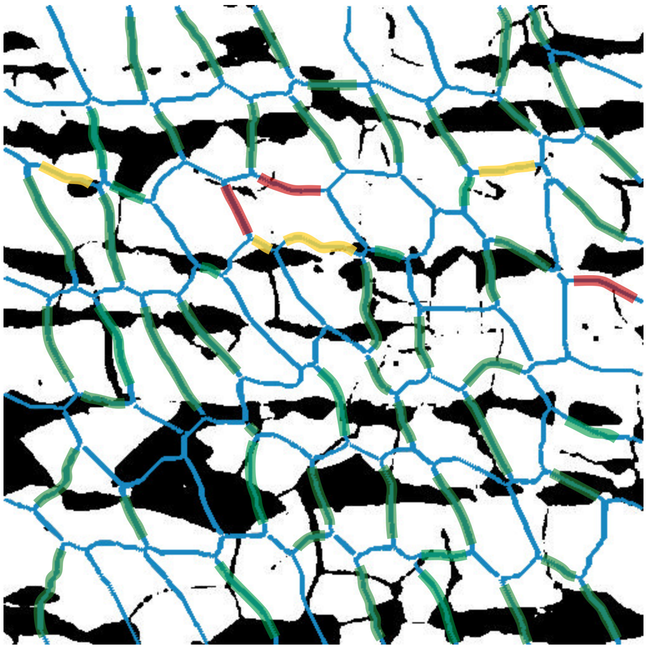
Efficient and Robust FETI-DP and BDDC Methods – Approximate Coarse Spaces and Deep Learning-Based Adaptive Coarse Spaces

Janine Weber



Efficient and Robust FETI-DP and BDDC Methods – Approximate Coarse Spaces and Deep Learning-Based Adaptive Coarse Spaces

INAUGURAL-DISSERTATION

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

vorgelegt von
Janine Weber
aus Frechen

Köln, 2022

Berichterstatter:	Prof. Dr. Axel Klawonn (Universität zu Köln)
	Prof. Dr. Oliver Rheinbach (TU Bergakademie Freiberg)
	Prof. Dr. Luca F. Pavarino (Università di Pavia)

Tag der mündlichen Prüfung: 17. Januar 2022

Abstract

For the approximate solution of partial differential equations (PDEs), which model many physical processes in nature, often numerical algorithms are used. Usually, the numerical solution relies on a discretization of the considered PDE, e.g., by finite elements, and leads to large systems of equations which have to be solved. Domain decomposition methods are robust and parallel scalable, preconditioned iterative algorithms for the solution of these large linear systems arising in the discretization of elliptic partial differential equations by finite elements. Domain decomposition methods rely on the subdivision of the computational domain into a number of smaller nonoverlapping or overlapping subdomains for which the local solutions can be computed completely in parallel on different cores of a parallel computer. In two-level domain decomposition methods, an additional coarse problem is solved in each iteration of the iterative solution process, to ensure a global transport of information between the different subdomains.

The FETI-DP and BDDC methods are highly scalable nonoverlapping domain decomposition methods which obtain their scalability and robustness from the definition of an appropriate coarse space. However, in a parallel implementation, the exact solution of the corresponding coarse problem can eventually become a bottleneck, given a large size of the coarse space which is often the case for three-dimensional model problems. Thus, we are interested in efficient coarse spaces which are preferably small and can be computed with low computational effort. One common approach to design more efficient coarse spaces is to replace the exact solution of the coarse problem by an approximate solution.

On the other hand, classic coarse spaces which exclusively use geometric information of the domain decomposition, usually experience a deteriorating convergence rate for second-order elliptic PDEs with coefficient distributions with a large contrast. In this case, robust, i.e., adaptive coarse spaces are necessary which enhance the coarse space with specific, problem-dependent constraints usually resulting from the solution of certain local eigenvalue problems.

In this thesis, we introduce and compare different efficient and robust FETI-DP and BDDC coarse spaces for two- and three-dimensional model problems. First, we present three approximate, i.e., efficient BDDC coarse spaces which are implemented in the same parallel software framework. Furthermore, we introduce a heuristic coarse space for both FETI-DP and BDDC which can be interpreted as a generalization of classic coarse spaces as well as a low-dimensional approximation of a specific adaptive coarse space. We will observe that this heuristic coarse space shows robust results for different real-world problems. Finally, we introduce a hybrid approach which exclusively implements adaptive constraints on certain equivalence classes of the domain decomposition which are classified as critical by a neural network in a preprocessing step. We show numerical results for both, two- and three-dimensional test problems. We can observe very promising results where a large number of the eigenvalue problems can be avoided to be computed while at the same time obtaining a robust convergence behavior for highly complex coefficient distributions.

Zusammenfassung

Für die approximative Lösung von partiellen Differentialgleichungen (PDGLen), welche viele physikalische Prozesse in der Natur modellieren, werden häufig numerische Verfahren verwendet. Im Allgemeinen basiert die numerische Lösung auf einer Diskretisierung der betrachteten PDGL, z.B. mit Hilfe Finiter Elemente, und führt auf große Gleichungssysteme, die gelöst werden müssen. Gebietszerlegungsverfahren sind robuste und parallel skalierbare, vorkonditionierte iterative Algorithmen für die Lösung solcher großer linearer Systeme, die aus der Diskretisierung von elliptischen partiellen Differentialgleichungen mittels Finiter Elemente entstehen. Gebietszerlegungsverfahren basieren auf der Idee, ein Gebiet in eine feste Anzahl an kleineren, nicht überlappenden oder überlappenden Teilgebieten zu unterteilen, für die die lokalen Lösungen komplett parallel auf verschiedenen Kernen eines parallelen Computers berechnet werden können. In Zwei-Level Gebietszerlegungsverfahren wird zusätzlich in jeder Iteration des iterativen Lösungsprozesses auch immer ein globales, grobes Problem gelöst, welches den globalen Informationsaustausch zwischen den einzelnen Teilgebieten sicherstellt.

Die FETI-DP und BDDC Methoden sind hoch skalierbare, nicht überlappende Gebietszerlegungsverfahren, die ihre Skalierbarkeit und Robustheit durch die Definition eines geeigneten Grobgitterraumes erhalten. Allerdings kann die exakte Lösung des zugehörigen groben Problems aufgrund der hohen Dimension des Grobgitterraumes in einer parallelen Implementierung zu einer Art Flaschenhals werden, was insbesondere bei dreidimensionalen Modellproblemen oft der Fall ist. Daher sind wir an effizienten Grobgitterräumen interessiert, die möglichst klein sind und mit geringem Rechenaufwand berechnet werden können. Ein gebräuchlicher Ansatz, um effizientere Grobgitterräume zu konstruieren, ist es, die exakte Lösung des groben Problems durch eine approximative Lösung zu ersetzen.

Andererseits zeigen klassische Grobgitterräume, die lediglich geometrische Informationen der Gebietszerlegung nutzen, für elliptische PDGL zweiter Ordnung mit Koeffizientenverteilungen mit hohen Kontrasten häufig eine verschlechterte Konvergenzrate. In diesem Fall sind robuste, d.h. adaptive Grobgitterräume notwendig, welche den Grobgitterraum mit speziellen, problemabhängigen Bedingungen aus der Lösung von bestimmten lokalen Eigenwertproblemen verbessern.

In dieser Arbeit führen wir verschiedene effiziente und robuste FETI-DP und BDDC Grobgitterräume für zwei- und dreidimensionale Modellprobleme ein und vergleichen diese miteinander. Zuerst präsentieren wir drei approximative, d.h. effiziente BDDC Grobgitterräume, die in derselben parallelen Software implementiert sind. Anschließend führen wir einen heuristischen Grobgitterraum für FETI-DP und BDDC ein, der als eine Verallgemeinerung von klassischen Grobgitterräumen sowie als eine niedrig-dimensionale Approximation eines speziellen adaptiven Grobgitterraumes interpretiert werden kann.

Wir werden sehen, dass dieser heuristische Grobgitterraum für verschiedene realistische Probleme robuste Ergebnisse liefert. Schließlich werden wir einen hybriden Ansatz vorstellen, der nur auf ausgewählten Äquivalenzklassen der Gebietszerlegung adaptive Bedingungen implementiert, die in einem vorherigen Schritt durch ein neuronales Netz als notwendig klassifiziert wurden. Wir zeigen numerische Ergebnisse sowohl für zwei- als auch für dreidimensionale Testprobleme. Wir können sehr vielversprechende Ergebnisse beobachten, bei denen wir eine große Anzahl an Eigenwertproblemen einsparen können, während wir gleichzeitig ein robustes Konvergenzverhalten für hochkomplexe Koeffizientenverteilungen erhalten.

Acknowledgements

First of all, I would like to thank my advisor Axel Klawonn for giving me the great opportunity to work on a range of highly interesting topics within his group during the last four years. I am also very grateful for the support and encouragement he has given me, especially before starting and at the beginning of my research, as well as for all the time he spent in discussions with me. I would also like to thank Oliver Rheinbach and Luca F. Pavarino for reviewing this thesis.

I would like to express special thanks to Alexander Heinlein and Martin Lanser. I am very thankful for your support and guidance during the last four years, for all your excellent implementation tips and many enthusiastic discussions.

Of course, I also want to thank all my other current and former colleagues, namely Viktor Grimm, Christian Hochmuth, Jascha Knepper, Martin Kühn, Sabine Musielack-Erle, and Matthias Uran. Thank you for a congenial working atmosphere as well as for numerous fruitful mathematical discussions on blackboards, whiteboards, and - during times of Zoom and Co. - on iPads and shared screens.

Furthermore, I also owe a special thanks to Sabrina Gippert for patiently sharing her experiences regarding her time as a doctorate with me before starting my own research.

I gratefully acknowledge the computing time granted by the Center for Computational Sciences and Simulation (CCSS) of the Universität of Duisburg-Essen and provided on the supercomputer magnitUDE (DFG grants INST 20876/209-1 FUGG, INST 20876/243-1 FUGG) at the Zentrum für Informations- und Mediendienste (ZIM). I would also like to thank the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS and its predecessor JUQUEEN at Jülich Supercomputing Centre (JSC). Additionally, the access to the computational facilities of the Center for Data and Simulation Science (CDS) at the University of Cologne is also gratefully acknowledged.

Moreover, I would like to thank my family and my friends for the constant support during the last years, either by patiently listening to me talking about mathematical topics or by providing some necessary distractions from math.

I also owe special thanks to my boyfriend Christoph. With all your patience and calmness, you have constantly been my oasis of peace during the last years, especially during less successful times. I am also grateful for many moments of laughters and adventures which we have shared, without talking or thinking about mathematics.

Finally, I have to express my deepest gratitude to my parents. Without your overwhelming love, support and constant encouragement to follow my goals, I would never have been at this point in my life. Danke für alles!

Contents

List of Tables

List of Figures

1 Introduction

Many physical processes in natural science, engineering, and medical research can be modeled by partial differential equations, often with prescribed boundary values on the boundary of the respective computational domain. In particular, in many engineering applications as, e.g., solid and structural mechanics, elliptic partial differential equations play an important role. Common examples for the application of partial differential equations are, for instance, the deformation of an elastic body under the action of external and internal forces or the prediction of flow within porous media; see, e.g., [22, 31, 36, 49, 84, 147, 156]. For many of these problems, the existence and uniqueness of a solution can be proven under certain assumptions on the boundary and the underlying domain. However, it is often impossible to derive a classic solution analytically; see, e.g., [49]. Here, numerical algorithms can be used to compute approximate solutions to a given problem.

In order to numerically compute an approximate solution of a given partial differential equation (PDE), the respective PDE is usually discretized. A popular technique in this context is the Finite Element Method (FEM), which discretizes the corresponding variational problem of the PDE in an appropriate, finite-dimensional finite element space; see, e.g., [22, 49]. This leads to a linear or nonlinear system of equations, whose solution corresponds to an approximate solution of the considered PDE. The precision of the computed solution is mainly determined by the mesh resolution. Thus, the precision is limited by the available computational resources and the condition of the considered problem. In particular, in FEMs, the accuracy of the numerical solution directly depends on the dimension of the respective finite element space. This results in very large systems of equations which need to be solved and which are, usually, also very ill-conditioned; see, e.g., [147, 165]. As a consequence, direct solution methods based on sophisticated variants of the Gaussian elimination algorithm are, especially in three spatial dimensions, often not feasible for the solution of the respective systems, due to a high demand of memory and limited computational resources. Even though direct solvers can, in principle, also be parallelized for the application on parallel computers, their potential with regard to parallel scalability is limited to a certain size of the considered system for three-dimensional problems. On the other hand, iterative methods as, e.g., Krylov subspace methods, which gradually approximate the solution, usually require a high number of iterations given the poor condition of the respective systems of equations and the resulting slow convergence behavior. Thus, adequate preconditioners are needed to accelerate the convergence and to obtain a robust iterative solver. In particular, given the increasingly parallel architecture of modern computers and supercomputers, we are interested in robust preconditioners which perform efficiently on such parallel environments.

For the iterative solution of linear systems of equations resulting from FEMs, usually, preconditioned Krylov subspace methods as the preconditioned conjugate gradient (PCG) method [64,81,153] or the generalized minimal residual (GMRES) method [64,154] are used. Here, one efficient approach to design an appropriate preconditioner are the class of domain decomposition methods; cf., e.g., [147, 165]. Domain decomposition methods (DDMs) are highly scalable, iterative methods which have been shown to be numerically stable for many practically relevant model problems and which are designed for the application on parallel computers. They rely on a divide-and-conquer strategy and decompose the original global problem into a number of smaller subproblems. Mathematically, this corresponds to a subdivision of the domain into a number of smaller subdomains, where the local problems can be solved in parallel on different processors. Depending on the specific method, the different local subdomains can be either overlapping or nonoverlapping. In order to obtain a continuous solution across the different subdomains, also communication between the different processors is necessary, i.e., a global coupling between the different subdomains has to be ensured. Among the historically first and best-known domain decomposition methods are the one- or two-level (overlapping) Schwarz methods; see, e.g., [157,165] and the references therein. One-level DDMs, where information is only exchanged between neighboring subdomains, result in a rate of convergence which deteriorates with a growing number of subdomains when used for the approximate solution of elliptic systems of PDEs; see, e.g., [165]. A remedy is obtained in two-level DDMs by the setup and the solution of an additional, small global problem which needs to be solved in each iteration of the iterative solver. In particular, in two-level DDMs, the additional global problem ensures the fast global transport of information between the different subdomains as well as the scalability and robustness of the iterative method. In the context of two-level DDMs, we refer to this global problem as *coarse problem* and to the related solution space as *coarse space*.

In the following, we focus on two specific nonoverlapping domain decomposition methods, i.e., the FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) [50, 51,116,165] and the BDDC (Balancing Domain Decomposition by Constraints) [33,38, 125,128,129] method. FETI-DP and BDDC have been shown to be numerically scalable for a range of different problems [116,123,124,128,133,142] and have been tested extensively on up to half a million cores of a parallel computer; see, e.g., [3,4,99–102,112,173]. Both methods obtain their parallel potential and scalability from the setup and the solution of an appropriate coarse problem. However, in a parallel implementation, the exact solution of the coarse problem can also become a bottleneck, i.e., a limiting factor with respect to the expected time to solution. Generally speaking, we are interested in a coarse problem, or a related coarse space, respectively, which fulfills the following two properties. First, we want to design a coarse space which is robust in the sense that it results in a good rate of convergence for highly heterogeneous and arbitrary coefficient distributions of the considered model problem. In general, the convergence rate of FETI-DP and BDDC methods is determined by the eigenvalues of the preconditioned system. For second-order elliptic PDEs, coefficient discontinuities with a large contrast can lead to a deterioration of the convergence rate. We will observe that this is usually the case for classic coarse spaces, which exclusively use geometric information of the domain decomposition. Second, we aim to design an efficient coarse space in the sense that it is preferably small and computationally cheap. We will see that, to a certain extent, both preferences can be contradictory and that the design of a coarse space which fulfills both desired properties is usually a trade-off between different strategies.

In this thesis, we present and numerically compare a range of different coarse spaces for the FETI-DP and BDDC method in two and three spatial dimensions. Let us note that it has been shown in [125, 129] that the FETI-DP and the BDDC method are dual to each other and share essentially the same spectra. Thus, many theoretical results are equally valid for both methods. In general, the condition number in FETI-DP and BDDC strongly depends on the choice of the primal variables, i.e., the coarse space and the considered PDE. Originally, coarse spaces which exclusively use geometric information of the domain decomposition were used. We refer to these coarse spaces as *classic coarse spaces*. In their most elementary form, exclusively primal constraints associated with the vertices at the cross points shared by two or more subdomains of the domain decomposition are chosen to set up the coarse problem. For scalar elliptic problems in two spatial dimensions and with constant coefficients on each subdomain, this results in a polylogarithmic condition number bound, where the constant is independent of the coefficient contrast. For more complex coefficient functions as well as in three spatial dimensions, additional primal constraints associated with averages along edges or faces of the domain decomposition are necessary to retain the robustness of the solver. In [111], the authors introduced weighted averages along faces between two neighboring subdomains, which have been numerically shown to be robust also for coefficient heterogeneities not aligned with the domain decomposition interface. However, for completely arbitrary or highly complex coefficient distributions, also weighted average constraints are not sufficient anymore. In particular, this results in a constant within the condition number estimate which depends on the contrast of the coefficient function. Thus, the potential with respect to robustness of classic coarse spaces is clearly limited. Let us note that coefficient functions with sharp jumps along and across the domain decomposition interface can occur, for instance, when modeling composite materials in solid or structural mechanics, as, e.g., the microstructure of a dual-phase steel. On top of that, classic coarse spaces can obtain a rather high dimension, given a coarse problem where weighted averages on all edges and faces of a three-dimensional domain decomposition are implemented. As mentioned above, the exact solution of the respective coarse problem can become a limiting factor in a parallel implementation with respect to scalability.

One approach to alleviate the latter problem is the usage of approximate coarse spaces for FETI-DP and BDDC methods. In principle, these methods replace certain components of the preconditioner, usually the solution of the coarse problem, by an approximate solution. During the recent two decades, different approximate variants of the BDDC and the FETI-DP method became popular and were numerically tested for the solution of different linear and nonlinear PDEs [3,4,39,41,99,100,102,110,112,126,149,150,167,168]. Especially in the context of BDDC methods, *multilevel methods* in different variations have been used; see, e.g., [4, 131, 161, 167, 168]. Here, exact BDDC is applied recursively to the original coarse problem. This leads to a smaller coarse problem which has to be solved exactly on the coarsest level and can thus delay the observed bottleneck within a parallel implementation. In this sense, approximate coarse spaces can be regarded as more efficient compared to the classic approaches, since the solution of the coarse problem can, in principle, be computed with less computational effort. Even though approximate coarse spaces can significantly increase the parallel potential of the respective domain decomposition method, as a drawback, they usually also result in a higher condition number estimate. On top of that, for most approximate coarse spaces, the convergence can also deteriorate for highly complex coefficient distributions with sharp jumps, resulting in a reduced robustness.

As a remedy, adaptive, i.e., problem-dependent coarse spaces have been proposed to design robust FETI-DP and BDDC methods. Typically, despite using geometric information, these coarse spaces use specific constraints which are computed from certain eigenvectors of local generalized eigenvalue problems. In [130,159], adaptive coarse spaces for FETI-DP and BDDC methods in two spatial dimensions were proposed, at this time without providing a corresponding theoretical bound. In [53, 54], the authors proposed eigenvalue problems on complete subdomains, which replaced a Poincaré inequality to set up adaptive coarse spaces for additive Schwarz methods. Later, in [132,161], the authors implemented the adaptive coarse space presented in [130, 159] in a parallel framework and tested it with BDDC in three spatial dimensions. In [97], the authors introduced an adaptive coarse space for FETI-DP and BDDC in two dimensions which also replaced a Poincaré inequality and an extension theorem. The complete theory for the resulting coarse space was provided in [107, 148]. In [108, 148], this adaptive coarse space was compared to those of [130,159], and [37] as well as a variant thereof introduced in [106], with respect to their robustness and performance for different two-dimensional test problems. Moreover, in [108, 148], a theoretical condition number estimate for the adaptive FETI-DP coarse space of [130, 159] was provided for two-dimensional model problems. In particular, the constant in the respective condition number estimate is independent of the contrast of the coefficient function, which ensures a robust convergence behavior. In [89], an adaptive coarse space for BDDC in two spatial dimensions was introduced.

For three-dimensional problems, it was shown in [92,119], that the coarse space of [130, 159] can result in high condition numbers and iteration counts for highly heterogeneous coefficient functions when only adaptive constraints on faces are implemented. Hence, the authors of [92, 119] proposed to additionally enrich the coarse space by adaptive constraints computed for edges between neighboring subdomains that do not share a face. Under this assumption, it is also possible to derive a theoretical condition number bound which exclusively depends on geometrical constants and which is independent of the coefficient contrast; cf. [92,119]. Thus, the resulting adaptive coarse space is robust also for arbitrary coefficient distributions.

For the BDDC method in three spatial dimensions, different authors proposed related adaptive coarse spaces in [12,29,141]. Furthermore, in [88], an adaptive coarse space has been considered both for FETI-DP and BDDC in three dimensions.

Let us note that the above enumeration of adaptive coarse spaces is not an exhaustive list of all existing methods. Especially for the overlapping Schwarz domain decomposition method, a range of different adaptive approaches exist which have not been mentioned here, e.g., [55,69,70]. In this thesis, however, we focus on nonoverlapping domain decomposition methods and, in particular, the adaptive FETI-DP coarse space as introduced in [130, 132, 159].

As mentioned above, most adaptive coarse spaces rely on the solution of local generalized eigenvalue problems to enhance the coarse space with selected eigenvectors. Even though this ensures the robustness of the iterative solver, the setup and the solution of the respective eigenvalue problems can become a limiting factor in the expected time to solution, especially in a parallel implementation. In [92, 93, 108, 119], different strategies to reduce the number of necessary eigenvalue problems based on the residual after one step of the adaptive FETI-DP or BDDC method have been considered, as well as heuristic approaches based on the identification of coefficient jumps on the considered parts of the interface. In [72, 75], an alternative approach which uses deep learning to design an efficient and robust FETI-DP coarse space has been proposed. This approach uses principles from the newly developing field of scientific machine learning [7] which combines concepts from machine learning with numerical algorithms for the approximate solution of PDEs. The publications [72, 75] are based on Chapter 8 of this thesis.

The remainder of this thesis is organized as follows. In the next chapter, we introduce two different model problems which we use as test problems to numerically compare the different approximate and adaptive coarse spaces presented in this thesis. Moreover, we briefly describe the Galerkin method and comment on the concepts of numerical and parallel scalability. In Chapter 3, we outline the two nonoverlapping domain decomposition methods FETI-DP and BDDC. In addition to a short presentation of classic coarse spaces, we also describe the adaptive FETI-DP coarse space as introduced in [130, 159] (for two dimensions) and [92, 119] (for three dimensions). In Chapter 4, we present common techniques to implement coarse space enrichments for both FETI-DP and BDDC. Followingly, in Chapter 5, we introduce and compare three different approximate BDDC coarse spaces in a common framework. Here, we especially focus on a three-level BDDC preconditioner, which uses the concept of BDDC recursively and is based on a third level of the domain decomposition into subregions. In Chapter 6, we present a heuristic coarse space for FETI-DP and BDDC, to which we refer to as *frugal coarse space* due to its relatively low computational effort. This coarse space can be interpreted as a low-dimensional approximation of the adaptive coarse space presented in Chapter 3. We numerically test this frugal coarse space for a range of different model problems, both for FETI-DP using MATLAB [134] as well as for BDDC using our parallel implementation based on PETSc [8, 9] and MPI [65, 158]. Chapter 7 provides a short overview of supervised machine learning techniques and deep learning. Finally, in Chapter 8, we present a hybrid approach that enhances the FETI-DP coarse space with adaptive constraints only on edges or faces, which are classified as critical by a neural network in a preprocessing step. On all other edges or faces, respectively, we do not enforce additional constraints and omit the respective eigenvalue problem. In the first part of Chapter 8, we describe the specific concept of our proposed method and explain the generation of training and validation data for the neural network both for two- and three-dimensional problems. In the second part of Chapter 8, we test the resulting coarse space for different realistic test problems, using both regular domain decompositions as well as irregular obtained by METIS [87]. Eventually, we summarize the main observations of this thesis and provide an outlook on possible and planned future work in Chapter 9.

2 Model problems and finite elements

In the present chapter, we introduce some basic concepts and some notation which will be used throughout this thesis. At first, we present two types of partial differential equations (PDEs) which serve as model problems for the numerical experiments discussed in this thesis. In particular, we will introduce stationary linear diffusion problems in Section 2.1 as well as compressible linear elasticity problems in Section 2.2. Both problems will be considered for two- and three-dimensional domains $\Omega \subset \mathbb{R}^d, d = 2, 3$, and in both cases, we will focus on highly heterogeneous problems with large discontinuities in the material stiffness or the diffusion coefficient, respectively. For more details and a theoretical description of the model problems, especially in the context of finite elements, please refer to, e.g., [22,24,31,147,165]. In Section 2.3, we will briefly summarize the Galerkin method for the discretization of the variational formulation of the partial differential equations presented in Sections 2.1 and 2.2. The discretization using the Galerkin method then leads to a linear system of equations which can - in principle - be solved with a direct or an iterative method. However, iterative methods are usually less robust than direct methods which is why often iterative Krylov subspace methods with robust preconditioners are used. In this context, we will comment on linear system solvers for the resulting linear system of equations in Section 2.4. In particular, we introduce the concepts of numerical and parallel scalability, which are highly relevant in the context of domain decomposition methods.

Throughout this chapter, let $\Omega \subset \mathbb{R}^d, d = 2, 3$, be a bounded polygonal or polyhedral domain and let $\partial\Omega_D \subset \partial\Omega$ be a closed subset of nonvanishing measure where we impose Dirichlet boundary conditions. On the remaining part of the boundary $\partial\Omega_N := \partial\Omega \backslash \partial\Omega_D$, we impose Neumann boundary conditions.

For the description of the variational formulation of our model problems, we further define the Sobolev space

$$H_0^1(\Omega, \partial\Omega_D)^k := \{v \in H^1(\Omega)^k : v = 0 \text{ on } \partial\Omega_D\}$$

of weakly differentiable functions on Ω. Let us note that we have $k = 1$ for the scalar diffusion equation for both $d = 2, 3$, whereas for the linear elasticity problem, we have $k = d$ and vector valued functions.

2.1 Stationary diffusion equation

As a first model problem for the numerical experiments presented in the remainder of this thesis, we consider a scalar elliptic boundary value problem which is also known as *stationary diffusion equation*. In particular, for a sufficiently smooth coefficient function

$\rho : \Omega \to \mathbb{R}$ and appropriate functions $f : \Omega \to \mathbb{R}$ and $g : \partial\Omega_N \to \mathbb{R}$, we have the boundary value problem

$$\begin{aligned} -\nabla \cdot (\rho \nabla u) &= f \text{ in } \Omega \\ u &= 0 \text{ on } \partial\Omega_D \\ \rho \nabla u \cdot n &= g \text{ on } \partial\Omega_N \end{aligned} \tag{2.1}$$

where n denotes the outer unit normal on $\partial\Omega_N$.

Let us note that throughout this thesis, we only consider a homogeneous flow $g = 0$. Thus, for a piecewise constant parameter distribution $\rho \in L^\infty(\Omega)$ with $\rho \geq \rho_{\min} > 0$ and $f \in L^2(\Omega)$, we obtain the weak formulation: Find $u \in V = H_0^1(\Omega, \partial\Omega_D)$ such that

$$a(u, v) = F(v) \quad \forall v \in V = H_0^1(\Omega, \partial\Omega_D),$$

where

$$a(u, v) := \int_\Omega \rho \nabla u \cdot \nabla v \, dx \quad \text{and} \quad F(v) := \int_\Omega f v \, dx. \tag{2.2}$$

For a detailed investigation of the existence and uniqueness of the solution of this model problem, based on Lax-Milgram's theorem, we refer to, e.g., [22, 165].

2.2 Compressible linear elasticity

As a second type of model problem, we consider compressible linearized or simply *linear elasticity* problems. Here, the domain $\Omega \subset \mathbb{R}^d, d = 2, 3$, can be interpreted as a body of elastic material which is deformed under the action of internal and external forces. In particular, we denote by $f : \Omega \to \mathbb{R}^d$ a given (internal) body force and by $g : \partial\Omega_N \to \mathbb{R}^d$ a given (external) surface force. Then, the difference between the starting or reference configuration and the deformed configuration of the body is denoted as the *displacement* and can be expressed by a displacement function $u : \Omega \to \mathbb{R}^d$. In a linear elasticity model, the displacement u is the solution of the following boundary value problem

$$\begin{aligned} -\operatorname{div}(\sigma(u)) &= f \text{ in } \Omega \\ \sigma(u) \cdot n &= g \text{ on } \partial\Omega_N \\ u &= 0 \text{ on } \partial\Omega_D, \end{aligned} \tag{2.3}$$

where n denotes the outer unit normal on $\partial\Omega_N$. Here,

$$\sigma(u) := \lambda \operatorname{tr}(\varepsilon(u)) \, I + 2\mu\varepsilon(u)$$

denotes the stress tensor and λ and μ are the material dependent Lamé constants. Furthermore, in (2.3), the linearized strain tensor $\varepsilon(v)$ is defined by the symmetric gradient

$$\varepsilon(v) := \frac{1}{2}(\nabla v + \nabla v^T) \quad \text{and} \quad \varepsilon_{ij}(v) := \frac{1}{2}\Big(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\Big), \ 1 \leq i, j \leq d. \tag{2.4}$$

Let us note that the Lamé constants of a material can easily be calculated from Young's modulus $E > 0$ and Poisson's ratio $\nu \in (0, \frac{1}{2})$ by

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)}.$$

In this thesis, we will consider only compressible linear elasticity problems which means that the Poisson ratio ν is bounded away from 1/2. For $\nu \to 1/2$, i.e., almost incompressible linear elasticity, the value of λ tends to infinity and locking effects occur when using a discretization with standard finite elements, which leads to a slow convergence. This will not be further discussed in this thesis.

In the following, we assume the material parameters E and ν to be piecewise constant and bounded on Ω. For $f \in L^2(\Omega)$ and $g \in L^2(\partial\Omega)$, we then obtain the variational formulation of compressible linear elasticity: Find $u \in V = H_0^1(\Omega, \partial\Omega_D)^d$ such that

$$a(u,v) = F(v) \quad \forall v \in V = H_0^1(\Omega, \partial\Omega_D)^d,$$

where

$$\begin{aligned} a(u,v) &:= \int_\Omega 2\mu\varepsilon(u) : \varepsilon(v) dx + \int_\Omega \lambda \text{div}(u)\text{div}(v) dx \\ \text{and} \quad F(v) &:= \int_\Omega f \cdot v dx + \int_{\partial\Omega_N} g \cdot v ds. \end{aligned} \tag{2.5}$$

Here, we use the notation

$$\varepsilon(u) : \varepsilon(v) := \sum_{i,j=1}^{d} \varepsilon_{ij}(u)\varepsilon_{ij}(v)$$

for the product of the linearized strain tensors.

For more details on the derivation of the variational formulation and theoretical statements regarding the existence and uniqueness of solutions of linear elasticity problems, see, e.g., [22, 31, 165].

2.3 The Galerkin method and finite elements

As we have seen in Sections 2.1 and 2.2, we can derive a variational formulation for both the stationary diffusion problem as well as the linear elasticity problem which is of the general form: Find $u \in V$ such that

$$a(u,v) = F(v) \quad \forall v \in V \tag{2.6}$$

for a given bilinear form $a(\cdot,\cdot)$, a linear functional $F(\cdot)$, and an appropriate Sobolev space V.

The main idea of the classic Galerkin approach is now to replace the infinite dimensional Sobolev space V by a finite dimensional subspace $V^h \subset V$. This can be seen as a discretization of the infinite dimensional space V. Common examples for the finite dimensional subspace V^h are, e.g., the space of conforming piecewise linear or quadratic finite element (FE) functions. We denote by $\varphi_i, i = 1, \dots, n$ the basis of V^h and note that $n < \infty$. Then, by replacing $v \in V$ by $v_h \in V^h$ in (2.6), we obtain

$$a(u_h, v_h) = F(v_h) \quad \forall v_h \in V^h. \tag{2.7}$$

We can then express $u_h \in V^h$ as a linear combination of the basis functions $\varphi_i, i = 1, \dots, n$ which, using $u_h = \sum_{i=1}^n u_i \varphi_i$, yields

$$\sum_{i=1}^{n} u_i a(\varphi_i, \varphi_j) = F(\varphi_j) \quad \forall j = 1, \dots, n. \tag{2.8}$$

The discretized variational formulation in (2.8) can equivalently be written as

$$Au = b, \tag{2.9}$$

where $A = (a_{ij})_{i,j}$ with $a_{ij} = a(\varphi_i, \varphi_j)$, $u = (u_i)_i$, and $b = (b_i)_i$ with $b_i = F(\varphi_i)$. Thus, in order to find the discretized solution u_h of (2.7) we have to solve the system of linear equations (2.9) for u.

Let us note that the specific choice of the finite element space V^h strongly depends on the considered model problem. For instance, for the numerical experiments in this thesis, we use conforming $\mathcal{P}_1$ finite elements for the numerical solution of stationary diffusion problems, whereas we use conforming $\mathcal{P}_1$ or $\mathcal{P}_2$ finite elements for compressible linear elasticity problems.

2.4 Solvers for systems of linear equations

As we have seen in Section 2.3, the discretization of the variational formulation in (2.6) using the Galerkin approach and finite elements leads to a linear system of equations. The solution of this linear system of equations basically corresponds to an approximate solution of the considered PDE. The precision of the computed solution using the Finite Element Method (FEM) is mainly determined, i.e., limited by the available computational resources and the condition of the considered problem. In principle, the solution of the linear system of equations (2.9) can be computed either by direct or iterative methods. However, in FEMs, the accuracy of the numerical solution directly depends on the dimension of the finite element space V^h. This usually results in very large systems of equations which need to be solved. Additionally, these systems are usually very ill-conditioned; see, e.g., [147, 165]. As a consequence, direct solution methods for the resulting system have a complexity which polynomially depends on the number of unknowns and thus a high demand of memory. Therefore, especially in three spatial dimensions, it is often not feasible to solve the respective system with a direct method and iterative methods are necessary.

For the iterative solution of linear systems of equations resulting from FEMs, usually, preconditioned Krylov subspace methods as the preconditioned conjugate gradient (PCG) method [64,81,153] or the generalized minimal residual (GMRES) method [64,154] are used. For both methods, the convergence of the linear solver applied to elliptic systems of PDEs strongly depends on the condition number of the considered system. Thus, to obtain a fast and robust solver, appropriate preconditioners are needed.

As also described in the introduction, an efficient approach to precondition the obtained linear system in (2.9) is the application of domain decomposition methods (DDMs). These methods are based on a divide-and-conquer strategy and use a subdivison of the domain into a number of smaller subdomains. Thus, DDMs are by construction designed for a parallel implementation, since the smaller subdomain problems can be solved in parallel on different processors. A detailed mathematical description of two specific DDMs will be given in Chapter 3. Even though DDMs are tailored for a parallel execution, also communication between the different processors is necessary to obtain a solution which is continuous across the different subdomains. In general, using a one-level DDM for elliptic systems of PDEs, in which information is only exchanged between neighboring subdomains, results in a rate of convergence which deteriorates with a growing number of subdomains; see, e.g., [165]. A remedy is obtained in two-level DDMs by the set-up and solution of an additional global problem which needs to be solved in each iteration of the iterative solver. In particular, in two-level DDMs, the additional global problem ensures the global transport of information between the different subdomains.

The global or coarse problem within DDMs further ensures the scalability and robustness of the iterative method. Generally speaking, the concept of scalability provides a measure whether a given algorithm performs well on a parallel machine. We distinguish between two types of scalability, i.e., numerical and parallel scalability. In general, we say that an algorithm is *scalable* if the computational effort to obain a solution of the same accuracy is proportional to the problem size. In the context of DDMs, we denote an algorithm as *numerically scalable* if its rate of convergence does not deteriorate with a growing number of subdomains. We further say that a DDM is *weakly parallel scalable* if the time to solution remains constant while varying the problem size and the number of parallel processors proportional to each other. Finally, a DDM is *strongly parallel scalable* if a problem with constant size is solved in half the time when doubling the number of parallel processors. Let us note that, in general, it is harder to achieve strong parallel scalability of an algorithm than weak parallel scalability. In this thesis, we will consider two specific DDMs, the FETI-DP and the BDDC method which are numerically scalable and can achieve weak and strong scalability in a parallel implementation; see Chapter 3 for more details and the respective references.

3 FETI-DP and BDDC – classic and adaptive coarse spaces

In this chapter, we give an algorithmic description of the FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) [50, 51, 116, 165] and the BDDC (Balancing Domain Decomposition by Constraints) method; see, e.g., [33, 38, 125, 128, 129]. Both methods are nonoverlapping domain decomposition methods [157, 165] and are very closely related to each other. Thus, we will first explain the main idea of nonoverlapping domain decomposition methods and introduce some basic notation in Section 3.1. In Section 3.2, we will then give a detailed description of the classic FETI-DP method and present the corresponding condition number bound using a classic FETI-DP coarse space which is usually based on the selection of primal vertices and weighted averages along edges or faces of the domain decomposition. Analogously, we will review the classic BDDC method and the resulting condition number estimate in Section 3.3. In Section 3.4, we will give a brief mathematical description of classic weighted average constraints for three-dimensional problems. However, we will see that the classic condition number bounds for both FETI-DP and BDDC only guarantee robustness of the iterative solver under fairly restrictive assumptions on the underlying coefficient or material distribution, respectively, of the considered model problem. A remedy can be obtained by computing adaptive coarse spaces [12, 17, 29, 44, 48, 53–55, 69, 70, 89, 92–94, 107, 108, 130, 132, 141, 143, 162, 163]. In this thesis, we will focus on a very specific adaptive coarse space for the FETI-DP method, which relies on a local jump operator and is based on the work in [92–94, 107, 108, 119, 130, 132, 148]. We will give an algorithmic description of this very specific adaptive coarse space in Section 3.5 and again present the corresponding condition number bound. Finally, in Section 3.6, we will formulate some concluding remarks with respect to central differences between classic and adaptive FETI-DP and BDDC coarse spaces as well as with respect to the related expected computational effort.

Parts of this chapter have already been published in modified or unmodified form in [72, 73, 75].

3.1 Nonoverlapping domain decomposition

As already mentioned, both the FETI-DP and the BDDC method are nonoverlapping domain decompositon methods. Generally speaking, domain decomposition methods (see, e.g., [157, 165]) are iterative methods which are used for the parallel solution of linear systems arising from a finite element discretization of partial differential equations; see also Section 2.4. They rely on a divide-and-conquer strategy and are based on a geometric decomposition of the domain into a finite number of nonoverlapping subdomains, i.e., a finite number of local problems. For both FETI-DP and BDDC, these local problems are coupled in a relatively low number of degrees of freedom. To ensure a fast convergence towards a continuous global solution for the entire domain, a *coarse problem* or global problem is solved in each iteration of the iterative solver which ensures a global transport of information among the subdomains. Additionally, the coarse problem, i.e., the definition of an appropriate *coarse space*, guarantees the scalability in the number of iterations of the iterative linear system solver. For completeness, let us mention that also the previously developed methods FETI (also FETI-1) and Balancing Neumann-Neumann (also Balancing Domain Decomposition) [21, 46, 52, 121, 127] have been fundamental for the development of FETI-DP and BDDC.

Let us now describe the preliminaries and notations for our domain decomposition methods to introduce the FETI-DP and BDDC algorithms. Parts of this chapter have already been published in modified or unmodified form in [72, 73, 75].

For a given domain $\Omega \subset \mathbb{R}^d, d = 2, 3$, we assume a decomposition into $N \in \mathbb{N}$ nonoverlapping subdomains $\Omega_i, i = 1, \ldots, N$, such that $\overline{\Omega} = \bigcup_{i=1}^N \overline{\Omega}_i$. We further assume that each of the subdomains Ω_i is the union of finite elements with matching finite element nodes on the interface

$$\Gamma := \left(\bigcup_{i=1}^N \partial\Omega_i \right) \setminus \partial\Omega_D.$$

In our case, each subdomain is the union of shape regular elements of diameter $\mathcal{O}(h)$. The diameter of a subdomain Ω_i is denoted by H_i or, generically, by $H = \max_i(H_i)$. Additionally, we denote by W_i the local finite element space associated with a subdomain Ω_i. In case of a two-dimensional domain $\Omega \subset \mathbb{R}^2$, the finite element nodes on the interface are either vertex nodes, belonging to the boundary of more than two subdomains, or edge nodes, belonging to the boundary of exactly two subdomains; see also Fig. 3.1 for an exemplary domain decomposition in two dimensions. Let us note that vertex nodes are often also referred to as corners or corner nodes. For the case of a three-dimensional domain $\Omega \subset \mathbb{R}^3$, edge nodes also belong to the boundary of more than two subdomains, and the interface further consists of face nodes, belonging to the boundary of exactly two subdomains; see, e.g., [109, Def. 2.1 and Def. 2.2] and [116, Def. 3.1]. For the remainder of this thesis, we will denote by $\mathcal{E}_{ij}$ an edge and by $\mathcal{F}_{ij}$ a face, respectively, between the subdomains Ω_i and Ω_j. All finite element nodes inside a subdomain Ω_i are denoted as interior nodes. For a given domain decomposition, we obtain local finite element problems

$$K^{(i)}\, u^{(i)} = f^{(i)}$$

with $K^{(i)} : W_i \to W_i$ and $f^{(i)} \in W_i$ by restricting the considered differential equation (see Sections 2.1 and 2.2) to Ω_i for each subdomain and discretizing its variational formulation in the finite element space W_i; see also Section 2.3. Let us remark that the local problems associated with the subdomains can be solved completely in parallel and independently of each other. However, the matrices $K^{(i)}$ are, in general, not invertible for subdomains which have no contact to the Dirichlet boundary $\partial\Omega_D$. We define the product space $W := \prod_{i=1}^N W_i$ and denote by $\widehat{W} \subset W$ the subspace of functions in W that are continuous on the interface Γ. For a detailed description of FETI-DP and BDDC in the following sections, we partition the finite element variables $u^{(i)} \in W_i$ into interior variables $u_I^{(i)}$, and, on the interface, into dual variables $u_\Delta^{(i)}$ and primal variables $u_\Pi^{(i)}$. We denote the respective degrees of freedom by the indizes I, Δ, and Π. For the remainder of this thesis, we always choose (at least) all degrees of freedom belonging to vertices as primal variables. Thus, the dual variables always belong to edges in two dimensions and to edges and faces in three dimensions, respectively. Let us note that also other choices are possible; see, e.g., [116]. Finally, we introduce the space $\widetilde{W}$, consisting of functions $w \in W$ that are continuous in the primal variables. We thus have the relation $\widehat{W} \subset \widetilde{W} \subset W$.

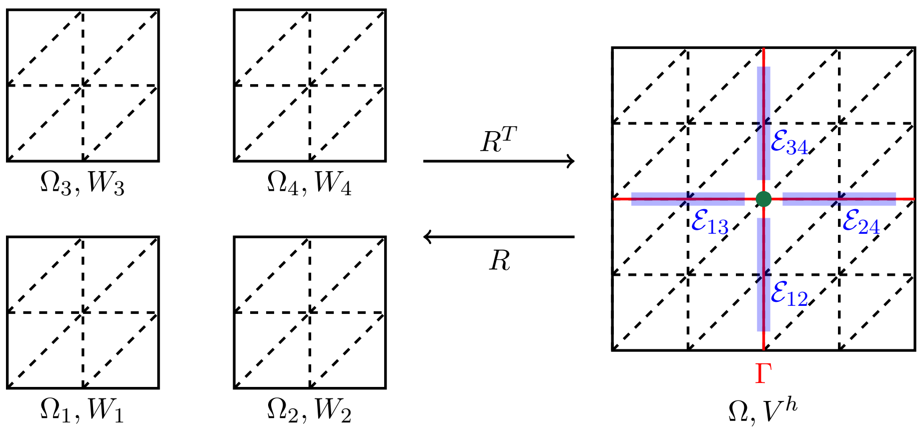


Figure 3.1: **Left:** Decomposition of a discretized domain $\Omega \subset \mathbb{R}^2$ into four subdomains Ω_i, $i = 1, ..., 4$. **Right:** Discretized domain Ω, corresponding finite element space V^h, interface Γ (marked in red), four edges $\mathcal{E}_{ij}$ (marked in blue), and one vertex (marked in green). The operator R^T acts as a finite element assembly operator on the interface. Figure in modified form in [104].

3.2 Classic FETI-DP

3.2.1 The FETI-DP preconditioner

As mentioned before, the FETI-DP method was first introduced in [50,51]. As a first step in the FETI-DP (and also the BDDC [38,128]) algorithm, we compute the local stiffness matrices $K^{(i)}$ and the local right-hand sides $f^{(i)}$ for each subdomain $\Omega_i, i = 1, \ldots, N$. The local problems are completely decoupled and, as already mentioned, the matrices $K^{(i)}$ are, in general, not invertible for subdomains without contact to the Dirichlet boundary $\partial\Omega_D$. These subdomains are also called *floating* subdomains. As a consequence, the local solution on floating subdomains is, in general, not unique and can be different from the global solution u of the partial differential equation, restricted to the respective subdomain, i.e., $u_{|\overline{\Omega}_i}$. In particular, the local stiffness matrices of floating subdomains have a non-trivial null space. For stationary diffusion problems, this non-trivial null space

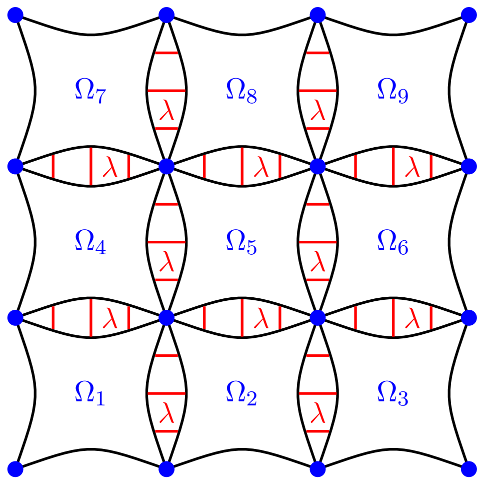


Figure 3.2: Decomposition of a square subdomain $\Omega \subset \mathbb{R}^2$ into nine subdomains $\Omega_i, i = 1, \ldots, 9$. The global assembly in primal vertices is marked with blue circles. On the remaining interface variables, i.e., the dual variables, continuity is iteratively enforced with Lagrange multipliers λ, marked in red.

consists of the constant functions whereas for linear elasticity problems, the respective null space consists of the rigid body modes, i.e., translations and rotations of the entire subdomain. In both cases, the null space needs to be controlled by the domain decomposition algorithm in order to obtain a continuous global solution. Both the FETI-DP and the BDDC algorithm deal with this difficulty by sub-assembling the decoupled system in selected primal variables Π.

For an algorithmic description of the FETI-DP method, let us first introduce the simple restriction operators $R_i : V^h \rightarrow W_i,\ i = 1, ..., N$, the block vectors $u^T := \left(u^{(1)T}, ..., u^{(N)T}\right)$ and $f^T := \left(f^{(1)T}, ..., f^{(N)T}\right)$, and the block matrices $R^T := \left(R_1^T, ..., R_N^T\right)$ and $K = \operatorname{diag}\left(K^{(1)}, ..., K^{(N)}\right)$. We then obtain the fully assembled system

$$K_g = R^T K R \tag{3.1}$$

and the fully assembled right-hand side

$$f_g = R^T f. \tag{3.2}$$

The block matrix K is not invertible as long as a single subdomain has no contact to the Dirichlet boundary. Thus, the system

$$Ku = f$$

has no unique solution, i.e., an unknown vector u might be discontinuous on the interface; cf. also the explanations at the beginning of this section. Let us now describe how the continuity of $u \in W := W_1 \times ... \times W_N$ on the interface is enforced using FETI-DP. Here, we use a presentation of the FETI-DP method which is very similar to the compact notation in [108] and which is also based on the description in [72, 73, 75].

We assume the following partitioning of the local stiffness matrices $K^{(i)}$, the local load vectors $f^{(i)}$, and the local solutions $u^{(i)}$ using the subdivision of the degrees of freedom as introduced in Section 3.1 into interior, primal and dual variables:

$$K^{(i)} = \begin{bmatrix} K_{II}^{(i)} & K_{\Delta I}^{(i)T} & K_{\Pi I}^{(i)T} \\ K_{\Delta I}^{(i)} & K_{\Delta\Delta}^{(i)} & K_{\Pi\Delta}^{(i)T} \\ K_{\Pi I}^{(i)} & K_{\Pi\Delta}^{(i)} & K_{\Pi\Pi}^{(i)} \end{bmatrix}, \; u^{(i)} = \begin{bmatrix} u_I^{(i)} \\ u_\Delta^{(i)} \\ u_\Pi^{(i)} \end{bmatrix}, \text{ and } f^{(i)} = \begin{bmatrix} f_I^{(i)} \\ f_\Delta^{(i)} \\ f_\Pi^{(i)} \end{bmatrix}.$$

It is often convenient to further introduce the union of interior and dual degrees of freedom as an additional set of degrees of freedom denoted by the index B. This leads to a more compact notation and we can define the following matrices and vectors

$$K_{BB}^{(i)} = \begin{bmatrix} K_{II}^{(i)} & K_{\Delta I}^{(i)T} \\ K_{\Delta I}^{(i)} & K_{\Delta\Delta}^{(i)} \end{bmatrix}, \; K_{\Pi B}^{(i)} = \begin{bmatrix} K_{\Pi I}^{(i)} & K_{\Pi\Delta}^{(i)} \end{bmatrix}, \text{ and } f_B^{(i)} = \begin{bmatrix} f_I^{(i)T} & f_\Delta^{(i)T} \end{bmatrix}^T.$$

We then introduce the block diagonal matrices

$$\begin{aligned} K_{BB} &= \operatorname{diag}_{i=1}^N K_{BB}^{(i)}, \\ K_{II} &= \operatorname{diag}_{i=1}^N K_{II}^{(i)}, \\ K_{\Delta\Delta} &= \operatorname{diag}_{i=1}^N K_{\Delta\Delta}^{(i)}, \\ \text{and } K_{\Pi\Pi} &= \operatorname{diag}_{i=1}^N K_{\Pi\Pi}^{(i)}. \end{aligned} \tag{3.3}$$

Analogously, we obtain the block vector $u_B = [u_B^{(1)T}, \ldots, u_B^{(N)T}]^T$ and the block right-hand side $f_B = \left[f_B^{(1)T}, \ldots, f_B^{(N)T}\right]^T$ which can be partitioned accordingly. As discussed before, the solution of the decoupled block diagonal system is, in general, not unique. For the FETI-DP algorithm, continuity in the primal variables Π is enforced by a finite element assembly process, while continuity in the dual variables Δ is enforced iteratively by Lagrangian multipliers λ. To describe the primal assembly process, we introduce the assembly operators $R_\Pi^{(i)^T}, i = 1, \ldots, N$, which consist of values in $\{0, 1\}$. This yields the primally assembled matrices

$$\widetilde{K}_{\Pi\Pi} = \sum_{i=1}^N R_\Pi^{(i)T} K_{\Pi\Pi}^{(i)} R_\Pi^{(i)} \text{ and } \widetilde{K}_{\Pi B} = \left[R_\Pi^{(1)T} K_{\Pi B}^{(1)}, \ldots, R_\Pi^{(N)T} K_{\Pi B}^{(N)}\right], \tag{3.4}$$

as well as the corresponding right-hand side

$$\widetilde{f} = \left[f_B^T, \left(\textstyle\sum_{i=1}^N R_\Pi^{(i)T} f_\Pi^{(i)}\right)^T\right]^T.$$

A graphical representation of a typical FETI-DP decomposition can be found in Fig. 3.2 where the assembly in the primal vertices is exemplarily visualized by the blue circles.

In order to additionally enforce continuity in the dual degrees of freedom, we introduce a jump operator $B_B = [B_B^{(1)}, \ldots, B_B^{(N)}]$ with $B_B^{(i)}$ having zero entries for the interior

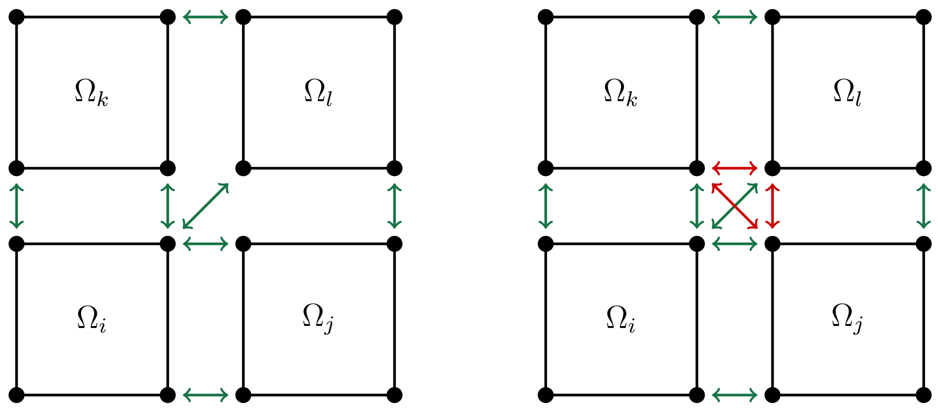


Figure 3.3: Visualization of nonredundant and redundant choice of Lagrange multipliers for FETI-DP. Example of four subdomains sharing a vertex. **Left:** Nonredundant choice of Lagrange multipliers, visualized by the green arrows. **Right:** Fully redundant choice of Langrange multipliers, visualized by the red arrows which are implemented in addition to the green arrows.

degrees of freedom and entries out of $\{-1, 1\}$ for the dual degrees of freedom. The entries for the dual degrees of freedom are chosen such that $B_B u_B = 0$ if and only if u_B is continuous on the interface. In particular, a row in $B_B u_B = 0$ enforces equality of two variables associated with the same physical point but two different subdomains. Thus, a typical row in B_B contains only a single 1 and a single -1. This continuity condition is enforced by the Lagrange multipliers λ, which act between two degrees of freedom each. In Fig. 3.2, the Lagrange multipliers are exemplarily indicated by the red lines.

Let us briefly comment on the observation that the definition of the jump matrix B, i.e., the choice of the Lagrange multipliers λ is not uniquely defined. First, the orientation of the Langrange multipliers is not uniquely defined, i.e., the rows of the matrix B can be multiplied by -1 without changing the continuity constraint nor the solution. Second, the Langrange multipliers can be chosen either nonredundantly or redundantly. For example, for a vertex in two dimensions (under the temporary assumption that this vertex is a dual variable) which is shared by four subdomains, a minimum of three and a maximum of six Langrange multipliers can be chosen without changing the solution; see Fig. 3.3 for a visualization of the nonredundant and the fully redundant choice. For a more detailed description of the possible choices of the Lagrange multipliers also for three dimensions, we further refer to [119]. In our implementations, we always use the fully redundant implementation of the Lagrange multipliers.

The FETI-DP master system is then given by

$$\begin{pmatrix} K_{BB} & \widetilde{K}_{\Pi B}^T & B_B^T \\ \widetilde{K}_{\Pi B} & \widetilde{K}_{\Pi\Pi} & O \\ B_B & O & O \end{pmatrix} \begin{pmatrix} u_B \\ \tilde{u}_\Pi \\ \lambda \end{pmatrix} = \begin{pmatrix} f_B \\ \tilde{f}_\Pi \\ 0 \end{pmatrix}. \tag{3.5}$$

To solve (3.5), the variables u_B and $\tilde{u}_\Pi$ are eliminated, resulting in a linear system in the Lagrange multipliers λ. By a block Gaussian elimination, we thus obtain the (unpreconditioned) standard FETI-DP system

$$F\lambda = d, \tag{3.6}$$

with

$$\begin{aligned} F &= B_B K_{BB}^{-1} B_B^T + B_B K_{BB}^{-1} \widetilde{K}_{\Pi B}^T \widetilde{S}_{\Pi\Pi}^{-1} \widetilde{K}_{\Pi B} K_{BB}^{-1} B_B^T \text{ and} \\ d &= B_B K_{BB}^{-1} f_B + B_B K_{BB}^{-1} \widetilde{K}_{\Pi B}^T \widetilde{S}_{\Pi\Pi}^{-1} \left(\left(\sum_{i=1}^{N} R_\Pi^{(i)T} f_\Pi^{(i)} \right) - \widetilde{K}_{\Pi B} K_{BB}^{-1} f_B \right). \end{aligned} \tag{3.7}$$

Here, the Schur complement $\widetilde{S}_{\Pi\Pi}$ for the primal variables is defined as

$$\widetilde{S}_{\Pi\Pi} = \widetilde{K}_{\Pi\Pi} - \widetilde{K}_{\Pi B} K_{BB}^{-1} \widetilde{K}_{\Pi B}^T. \tag{3.8}$$

As we can observe from (3.7), the application of F can be split into two additive parts. Due to its block structure, the first part requires only local operations and can be executed completely in parallel. The second part, however, requires the solution of a coupled coarse problem in form of the application of $\widetilde{S}_{\Pi\Pi}^{-1}$. Here, $\widetilde{S}_{\Pi\Pi}$ represents the (global) coarse space. Let us recall that, in general, all matrices including the primal variables Π are partially assembled in these primal variables and thus global. However, their size strongly depends on the choice of the primal variables, i.e., the global coarse space. This is especially valid for the global matrix $\widetilde{S}_{\Pi\Pi}$.

The considered system of equations (3.6) is then solved by a Krylov subspace method, such as the PCG or GMRES method; see also Section 2.4. Thus, the FETI-DP method is the iterative solution of the preconditioned system

$$M^{-1} F \lambda = M^{-1} d. \tag{3.9}$$

In this thesis, we always use the PCG method and the standard Dirichlet preconditioner $M_D^{-1} =: M^{-1}$ given by

$$M_D^{-1} = B_{B,D} \begin{bmatrix} 0 & I_\Delta \end{bmatrix}^T \left(K_{\Delta\Delta} - K_{\Delta I} K_{II}^{-1} K_{\Delta I}^T \right) \begin{bmatrix} 0 & I_\Delta \end{bmatrix} B_{B,D}^T = B_D \widetilde{S} B_D^T;$$

see, e.g., [50, 51]. Here, I_Δ is the identity matrix on the dual degrees of freedom. The matrices $B_{B,D}$ and B_D are scaled variants of the jump operators B_B and B, respectively; cf. also the following explanations below. Thus, the Dirichlet preconditioner M_D^{-1} is basically a weighted sum of local Schur complements

$$S_{\Delta\Delta}^{(i)} = K_{\Delta\Delta}^{(i)} - K_{\Delta I}^{(i)} (K_{II}^{(i)})^{-1} K_{\Delta I}^{(i)T}$$

which can be computed and applied in parallel. Let us remark that also several other choices for the preconditioner M^{-1} are possible, such as, e.g., the lumped preconditioner given by

$$M_L^{-1} := \sum_{i=1}^{N} B_{\Delta,D}^{(i)} K_{\Delta,\Delta}^{(i)} B_{\Delta,D}^{(i)T},$$

which only considers the stiffness matrices on the interface.

Finally, let us comment on the choice of the scaling procedure for the scaled matrices $B_{B,D}$ and B_D and the choice of the primal variables. In our implementations, we will - unless not explicitly mentioned otherwise - consider the ρ-scaling approach; see, e.g., [111, 119,148,165] for a mathematical description of this approach in two and three dimensions. In this case, the scaling matrices $D^{(i)} : \operatorname{range}(B) \to \operatorname{range}(B), i = 1, \dots, N$, depend on the coefficient function and are diagonal, and we can write

$$B_D = [D^{(1)}B^{(1)}, \dots, D^{(N)}B^{(N)}]. \tag{3.10}$$

Note that also non-diagonal scaling matrices exist, e.g., resulting from deluxe scaling; see [13, 42, 43, 108] and the references therein. In this case, the scaling matrices are obtained from local Schur complements and thus computationally more expensive but often more robust; see also [119].

Despite the choice of the scaling matrices and proper weights for the preconditioner induced by the PDE coefficients, the choice of the primal variables Π fundamentally determines the convergence properties of the FETI-DP method. As already mentioned, throughout this thesis, we will always choose all vertices as primal variables. However, it is well-known in the literature, that, especially in three dimensions, using exclusively primal vertex constraints is not sufficient to obtain a robust convergence behavior and further coarse space enhancements are necessary; see also the references in Section 3.2.2.

3.2.2 Condition number bound

In general, the convergence behavior and the condition number estimate for the FETI-DP method depend strongly on the chosen primal constraints Π and the induced scaling matrices for the preconditioner of the FETI-DP system. In order to estimate the spectral condition number $\kappa(M_D^{-1}F)$ of the preconditioned FETI-DP system, we have to estimate the largest eigenvalue $\lambda_{\max}(M_D^{-1}F)$ and the smallest eigenvalue $\lambda_{\min}(M_D^{-1}F)$ from below and above. In particular, it is sufficient to show that the Rayleigh quotient

$$\frac{\langle M_D^{-1}F\lambda, \lambda\rangle_F}{\langle \lambda, \lambda\rangle_F}$$

is bounded from below and above to obtain estimates for $\lambda_{\min}(M_D^{-1}F)$ and $\lambda_{\max}(M_D^{-1}F)$ by simple algebraic arguments. Here, we define $\langle M_D^{-1}F\lambda, \lambda\rangle_F := \langle M_D^{-1}F\lambda, F\lambda\rangle$. Without recapitulating all the technical details, let us recall that this condition number estimate of FETI-DP is strongly connected to an estimate of the operator

$$P_D := B_D^T B. \tag{3.11}$$

Then, for arbitrary λ and $w := \widetilde{S}^{-1}B^T\lambda \in \widetilde{W}$, the following relation is valid:

$$\frac{\langle M_D^{-1}F\lambda, \lambda\rangle_F}{\langle \lambda, \lambda\rangle_F} = \frac{\langle B_D\widetilde{S}B_D^T B\widetilde{S}^{-1}B^T\lambda, B\widetilde{S}^{-1}B^T\lambda\rangle}{\langle \widetilde{S}^{-1}B^T\lambda, \widetilde{S}^{-1}B^T\lambda\rangle_{\widetilde{S}}} = \frac{\langle P_Dw, P_Dw\rangle_{\widetilde{S}}}{\langle w, w\rangle_{\widetilde{S}}} = \frac{|P_Dw|_{\widetilde{S}}^2}{|w|_{\widetilde{S}}^2}. \tag{3.12}$$

It can be shown that the lower bound of the Rayleigh quotient in (3.12) is given by one and thus we are interested in deriving an upper bound of the type

$$|P_D w|_{\widetilde{S}} \leq C |w|_{\widetilde{S}} \ \forall w \in \widetilde{W} \tag{3.13}$$

for the operator P_D as defined in (3.11); see also [115, 117] for a first usage of the P_D-operator in this context.

At the end of this section on classic FETI-DP, let us now briefly recall the condition number bounds for the FETI-DP method as described in Section 3.2.1 for the stationary diffusion problem and compressible linear elasticity. Please note that this is not an exhaustive list of results since more results exists under very specific assumptions on the coefficient functions and for different model problems.

In two dimensions, the preconditioned FETI-DP method with a standard vertex coarse space satisfies the polylogarithmic condition number bound

$$\kappa(M_D^{-1} F) \leq \widetilde{C} \left(1 + \log\left(\frac{H}{h}\right)\right)^2 \tag{3.14}$$

with the constant $\widetilde{C}$ independent of H, h, and jumps in the PDE coefficients; see [114, 116, 117]. However, this condition number bound does only hold under certain assumptions on the coefficient function or the material distribution, e.g., for constant or slowly varying coefficients within each subdomain; see, e.g., [133, 165].

In three dimensions, the preconditioned FETI-DP method with a standard vertex coarse space performs less well [50] and the condition number bound (3.14) cannot be retained; see [117]. Therefore, enforcing additional coarse constraints based on averages over edges or faces was proposed by several authors; see, e.g., [51, 116, 117]. Then, the condition number bound in (3.14) also holds in three dimensions for heterogeneous coefficients that are constant within each subdomain or slowly varying coefficients; see, e.g., [116, 117]. For a detailed and technical proof of the cited condition number estimates based on an upper and a lower bound of the Rayleigh quotient of the preconditioned FETI-DP system and Poincaré inequalities, we refer to, e.g., [114, 133] for two-dimensional model problems and [116, 117] for three dimensions.

To conclude this section, let us note that in [111, Sect. 7], weighted edge averages for coefficient jumps not aligned with the interface were studied numerically for the FETI-DP algorithm. In this thesis, in Chapter 6, we propose a different approach to enhance the coarse space, using generalized weighted edge or face averages, which is strongly motivated by the adaptive coarse space introduced in Section 3.5 as well as by the heuristic approach presented in [111]. As we will observe in Sections 6.4 and 6.5, these generalized weighted averages are able to retain a robust convergence behavior for many heterogeneous coefficient functions. However, for completely arbitrary coefficient distributions with high contrasts, usually, adaptively computed, i.e., problem-dependent coarse spaces are necessary to guarantee a robust convergence behavior. In contrast to the heuristically motivated approach presented in Chapter 6, a theoretical condition number bound can be derived for such adaptive FETI-DP and BDDC coarse spaces; see Section 3.5 where a very specific adaptive FETI-DP coarse space is considered.

3.3 Classic BDDC

3.3.1 The BDDC preconditioner

As mentioned before, the FETI-DP and the BDDC method are closely related to each other. Therefore, we can reuse some of the notation introduced in Section 3.2 from the FETI-DP method for the description of BDDC. The BDDC method has been proposed and studied by different authors; see [33, 38, 125, 128, 129].

For the description of the BDDC algorithm, we use the same sub-partitioning of the degrees of freedom into the index sets I, Γ, Π and Δ as already introduced in Section 3.2. Here, we present the original BDDC formulation for the Schur complement system; see [38, 128]. Equivalently, it is also possible to formulate the BDDC preconditioner as a preconditioner for the fully assembled system $K_g u = f_g$; see, e.g., [126]. Please note that the BDDC method is dual to the FETI-DP method and therefore, the condition number bounds for both methods are closely related; see [125, 129]. In particular, it was shown in [125, 129] that FETI-DP and BDDC share essentially the same spectra except for eigenvalues that are equal to zero and one; see also Section 3.3.2.

In contrast to the FETI-DP method, we now use a slightly different ordering of the variables to describe the BDDC method. In particular, for this section, we introduce the block diagonal matrices

$$\begin{aligned} K_{\Pi\Pi} &= \operatorname{diag}\left(K_{\Pi\Pi}^{(1)}, ..., K_{\Pi\Pi}^{(N)}\right), \\ K_{\Pi I} &= \operatorname{diag}\left(K_{\Pi I}^{(1)}, ..., K_{\Pi I}^{(N)}\right), \\ \text{and } K_{\Pi\Delta} &= \operatorname{diag}\left(K_{\Pi\Delta}^{(1)}, ..., K_{\Pi\Delta}^{(N)}\right) \end{aligned} \tag{3.15}$$

as well as the corresponding right-hand sides

$$\begin{aligned} f_I^T &:= \left(f_I^{(1)T}, ..., f_I^{(N)T}\right), \\ f_\Delta^T &:= \left(f_\Delta^{(1)T}, ..., f_\Delta^{(N)T}\right), \\ \text{and } f_\Pi^T &:= \left(f_\Pi^{(1)T}, ..., f_\Pi^{(N)T}\right). \end{aligned} \tag{3.16}$$

The matrices $K_{II}, K_{I\Delta}, K_{\Delta I}$, and $K_{\Delta\Delta}$ are defined analogously to Section 3.2. Thus, the global block matrix K_{BDDC} for the BDDC algorithm can be written as

$$K_{\text{BDDC}} = \begin{bmatrix} K_{II} & K_{I\Delta} & K_{I\Pi} \\ K_{\Delta I} & K_{\Delta\Delta} & K_{\Delta\Pi} \\ K_{\Pi I} & K_{\Pi\Delta} & K_{\Pi\Pi} \end{bmatrix}.$$

Throughout this section, we will use the subindex '*BDDC*' to distinguish the matrices in this section from the global matrices used in FETI-DP (see Section 3.2). Please note that K_{BDDC} is assembled only inside the subdomains and not across the interface. In fact, K_{BDDC} can be obtained from the block matrix K defined in Section 3.2 by row and column permutations. The global elimination of the inner variables u_I in K_{BDDC} yields the unassembled Schur complement matrix

$$S_{\text{BDDC}} = \begin{bmatrix} S_{\Delta\Delta} & S_{\Delta\Pi} \\ S_{\Pi\Delta} & S_{\Pi\Pi} \end{bmatrix} = \begin{bmatrix} K_{\Delta\Delta} & K_{\Delta\Pi} \\ K_{\Pi\Delta} & K_{\Pi\Pi} \end{bmatrix} - \begin{bmatrix} K_{\Delta I} \\ K_{\Pi I} \end{bmatrix} K_{II}^{-1} \begin{bmatrix} K_{I\Delta} & K_{I\Pi} \end{bmatrix}$$

as well as the corresponding right-hand side

$$g_{\text{BDDC}} = \begin{bmatrix} g_{\Delta} \\ g_{\Pi} \end{bmatrix} = \begin{bmatrix} f_{\Delta} - K_{\Delta I} K_{II}^{-1} f_I \\ f_{\Pi} - K_{\Pi I} K_{II}^{-1} f_I \end{bmatrix}.$$

In the BDDC algorithm, we use a dual assembly operator $R_{\Delta}^T = \left(R_{\Delta}^{(1)T}, ..., R_{\Delta}^{(N)T} \right)$ instead of the Boolean jump operator from FETI-DP to enforce continuity in the dual variables. The unpreconditioned BDDC system then corresponds to the global Schur complement system $S_g u_g = g_g$ with the assembled global Schur complement given by

$$S_g = \begin{bmatrix} R_{\Delta}^T & 0 \\ 0 & R_{\Pi}^T \end{bmatrix} \begin{bmatrix} S_{\Delta\Delta} & S_{\Delta\Pi} \\ S_{\Pi\Delta} & S_{\Pi\Pi} \end{bmatrix} \begin{bmatrix} R_{\Delta} & 0 \\ 0 & R_{\Pi} \end{bmatrix} \tag{3.17}$$

and

$$g_g = \begin{bmatrix} R_{\Delta}^T & 0 \\ 0 & R_{\Pi}^T \end{bmatrix} g_{\text{BDDC}},$$

where $R_{\Pi}^T = \left(R_{\Pi}^{(1)T}, ..., R_{\Pi}^{(N)T} \right)$; see also Section 3.2.1 for a definition of the primal assembly operators $R_{\Pi}^{(i)T}, i = 1, \ldots, N$.

As for the FETI-DP algorithm, we use an appropriate preconditioner to accelerate the convergence of the iterative solver. Throughout this thesis, we again use the PCG algorithm and the Dirichlet preconditioner given by

$$M_{D,\text{BDDC}}^{-1} = \begin{bmatrix} R_{\Delta,D}^T & 0 \\ 0 & I_{\Pi} \end{bmatrix} \widetilde{S}_{\text{BDDC}}^{-1} \begin{bmatrix} R_{\Delta,D} & 0 \\ 0 & I_{\Pi} \end{bmatrix}. \tag{3.18}$$

Here, $\widetilde{S}_{\text{BDDC}}$ denotes the primally assembled Schur complement matrix defined by

$$\widetilde{S}_{\text{BDDC}} = \begin{bmatrix} I_{\Delta} & 0 \\ 0 & R_{\Pi}^T \end{bmatrix} \begin{bmatrix} S_{\Delta\Delta} & S_{\Delta\Pi} \\ S_{\Pi\Delta} & S_{\Pi\Pi} \end{bmatrix} \begin{bmatrix} I_{\Delta} & 0 \\ 0 & R_{\Pi} \end{bmatrix}$$

and $R_{\Delta,D}$ is a scaled variant of the dual assembly operator R_{Δ}. Generally, the same scalings as in the FETI-DP algorithm can be used in the BDDC preconditioner. However, instead of scaling the jump operator associated with the dual degrees of freedom as in (3.10), here, we scale the assembly operator R_{Δ}^T which connects the dual degrees of freedom along the interface. In particular, for the ρ-scaling, an entry in a row of R_{Δ}^T associated with a node x in subdomain Ω_j will be scaled by the respective scaling factor associated with Ω_j. This is different to FETI-DP, where the respective entry in $B^{(j)}$ will be scaled by the respective scaling factor associated with Ω_i. For a more detailed and mathematical description of the ρ-scaling for BDDC and other scaling variants, we refer to [148] and [119] as well as the references therein.

Since the FETI-DP and the BDDC method are dual and therefore very closely related to each other, in general, the same consideration for the selection of primal constraints are valid for BDDC as for FETI-DP. In particular, our aim is again to construct a BDDC coarse problem which is robust for a wide range of coefficient problems but, at the same time, preferably computationally cheap and small. As for the FETI-DP method, also for BDDC, we will select all vertices as primal variables Π. For more complex model problems as presented later in this section, the primal coarse space will be further modified by using an approximate coarse space, cf. Chapter 5, or by using an enhanced coarse space based on frugal constraints; cf. Chapter 6.

3.3.2 Condition number bound

As mentioned before, it was shown in [129] that the BDDC and the FETI-DP methods have, except for some eigenvalues equal to zero and one, the same spectra (see also [25, 125] for an alternative proof). Thus, the condition number estimates given in Section 3.2.2 for FETI-DP are also valid for the BDDC algorithm. Let us recall from Section 3.2.2 that the proof for the condition number estimate for FETI-DP relies on an estimate for the jump operator P_D as defined in (3.11) as well as on related estimates for the Rayleigh quotient of the preconditioned FETI-DP system. For the BDDC method, however, a more natural approach is to consider the closely related operator

$$E_D := I - P_D. \tag{3.19}$$

It can be shown that the task of estimating the eigenvalues of the preconditioned BDDC system $M_{D,\mathrm{BDDC}}^{-1} S_g$ can be reduced to the problem of estimating the eigenvalues of E_D; see [63, 115, 125, 129]. Thus, as well as by using the relation in (3.19), the operator P_D is again essential for deriving a condition number bound of the preconditioned BDDC method.

3.4 Classic weighted average constraints in three dimensions

As we have just observed for FETI-DP and BDDC, in three dimensions, using exclusively primal vertex constraints is not sufficient to obtain a robust condition number estimate and additional coarse space enhancements are necessary. In this section, we briefly describe classic coarse constraints based on weighted averages along edges and faces of the domain decomposition as introduced in [111]. To avoid a proliferation of notation, we will focus on the three-dimensional case. However, the presented ideas can equally be adapted to two dimensions, where we only consider weighted averages along edges.

For classic coarse spaces in three dimensions, we introduce weighted averages

$$\frac{\sum\limits_{x_i \in \mathcal{X}_{ij}} \hat{r}_j(x_i) u(x_i)}{\sum\limits_{x_i \in \mathcal{X}_{ij}} \hat{r}_j(x_i)^2}, \quad j = 1, ..., l \tag{3.20}$$

for equivalence classes $\mathcal{X}_{ij}$ on parts of the interface, e.g., edges $\mathcal{E}_{ij}$ or faces $\mathcal{F}_{ij}$, and points x on $\mathcal{X}_{ij}$. Here, we have $l = 1$ for the scalar diffusion case and $l = 3$ or $l = 6$ in the case of linear elasticity. Let us remark that in the latter case only five of the six constraints might be linearly independent on straight edges; see [111, 116]. For the stationary diffusion case, we consider the weights defined by the coefficient ρ via

$$\hat{\rho}(x) = \max_{y \in \omega(x)} \rho(y).$$

For the case of linear elasticity, the weights are defined by the maximum Young modulus, i.e., we choose

$$\hat{E}(x) = \max_{y \in \omega(x)} E(y).$$

We further define pointwise

$$\hat{r}_1(x) = \hat{\rho}(x)$$

in the scalar case and

$$\hat{r}_j(x) = \hat{E}(x) r_j(x), \quad j = 1, ..., 6$$

in case of linear elasticity, where r_1, r_2, and r_3 are the three translations and r_4, r_5, and r_6 the three rotations, respectively. More precisely, we have $r_i := \mathrm{e}_i, i = 1, 2, 3$, where $\mathrm{e}_i, i = 1, 2, 3$, are the three standard unit vectors, and the three linear (approximations to) rotations

$$r_4 := \frac{1}{H} \begin{bmatrix} x_2 - \widehat{x}_2 \\ -x_1 + \widehat{x}_1 \\ 0 \end{bmatrix}, \quad r_5 := \frac{1}{H} \begin{bmatrix} -x_3 + \widehat{x}_3 \\ 0 \\ x_1 - \widehat{x}_1 \end{bmatrix}, \quad r_6 := \frac{1}{H} \begin{bmatrix} 0 \\ x_3 - \widehat{x}_3 \\ -x_2 + \widehat{x}_2 \end{bmatrix},$$

where $\widehat{x} \in \widehat{\Omega}$ is the center of the linear rotations; see, e.g., [111, Sect. 2].

Let us note that the weighted average in (3.20) is reduced to the standard average along edges or faces as introduced in, e.g., [51, 111, 116, 117], in the case of coefficient jumps aligned with the interface. On top of that, the weighted average in (3.20) can be helpful in cases where the discontinuities do not align with the interface [111]. However, also the robustness of these weighted classic averages is limited; cf. [111]. Finally, let us remark that in [111], only weighted translations, i.e., $\hat{r}_k, k = 1, ..., 3$, have been used and thus the coarse space described in this subsection is in fact an extension of the robust coarse space presented in [111].

3.5 FETI-DP with an adaptive coarse space based on a local jump operator

As described in the introduction of this thesis as well as in Sections 3.2.2 and 3.3.2, during the past decades, a relatively wide range of nonadaptive coarse spaces has been developed for the FETI-DP and the BDDC method for different model problems and specific heterogeneous coefficient or material distributions [38, 51, 57, 58, 109, 111, 116, 117, 125, 146, 165]. However, as we have also seen in Section 3.2.2, for completely arbitrary coefficient distributions, the constant $\widetilde{C}$ in the condition number estimate (3.14) usually depends on the contrast of the coefficient function and thus the classic methods might not converge. At the beginning of this section, we will present two simple two-dimensional examples where the classic FETI-DP coarse space with classic averages along the edges of the domain decomposition interface shows a poor convergence behavior. For both examples, we will show comparative results for the frugal coarse space which will be introduced in Chapter 6 and a specific adaptive FETI-DP coarse space [130, 132].

Parts of this chapter have already been published in modified or unmodified form in [73].

Let us consider two small motivating examples of stationary diffusion on the unit square which is partitioned into $N = 4 \times 4$ regular subdomains. We enforce homogeneous Dirichlet boundary conditions on the left side of the unit square. On the remaining part of the boundary, we impose homogeneous Neumann boundary conditions. For this model problem, we consider two different heterogeneous coefficient distributions. As a first example, we consider shifted boxes of a high coefficient crossing an edge as shown in Fig. 3.4 (left). As the results in Table 3.1 show, a standard FETI-DP method with a classic nonadaptive coarse space (middle column), where all vertices and all edge averages are made primal, experiences a bad convergence behavior, since the condition number estimate increases in proportion to the coefficient contrast. For this specific example, a *frugal* coarse space which uses generalized weighted averages along each edge of the domain decomposition (right column) is able to retain a robust convergence behavior. As we can observe from Table 3.1, using the frugal coarse space, we obtain a condition number estimate which is independent of the coefficient contrast. Additionally, we observe satisfactory iteration numbers. In particular, in this case, the performance of the frugal coarse space is comparable to the adaptive FETI-DP coarse space presented in this section; see Table 3.1 (left column). A detailed description of frugal coarse spaces for stationary diffusion and linear elasticity problems will be given in Chapter 6. In a next step, we extend the example given in Fig. 3.4 (left) to the more complex case in Fig. 3.4 (right), where we have two straight horizontal channels of a high coefficient crossing each subdomain. In this case, the frugal coarse space deteriorates in convergence as well; see Table 3.2 (right column). In particular, we obtain a condition number estimate which also depends on the contrast of the coefficient function, even though the iteration number stays modest. A similar behavior is also observed for the classic FETI-DP coarse space; see Table 3.2 (middle column).

A remedy for the observed behavior can be obtained by *adaptive* coarse spaces, which have been proposed by several authors for both overlapping and nonoverlapping domain decomposition methods [12, 17, 29, 44, 48, 53–55, 69, 70, 89, 92, 93, 93, 107, 108, 130, 132, 141, 143, 162, 163]. These coarse spaces are adapted with respect to the specific considered model problem and are thus problem-dependent and robust for arbitrary heterogeneities. We can observe from Table 3.2 that the adaptive FETI-DP coarse space shows a robust rate of convergence also for the heterogeneous example in Fig. 3.4 (right). The basic idea of most of these adaptive methods is to use additional coarse modes or primal constraints obtained by solving localized eigenvalue problems on edges, local interfaces, or subdomains to enhance the coarse space prior to the first iteration of the iterative

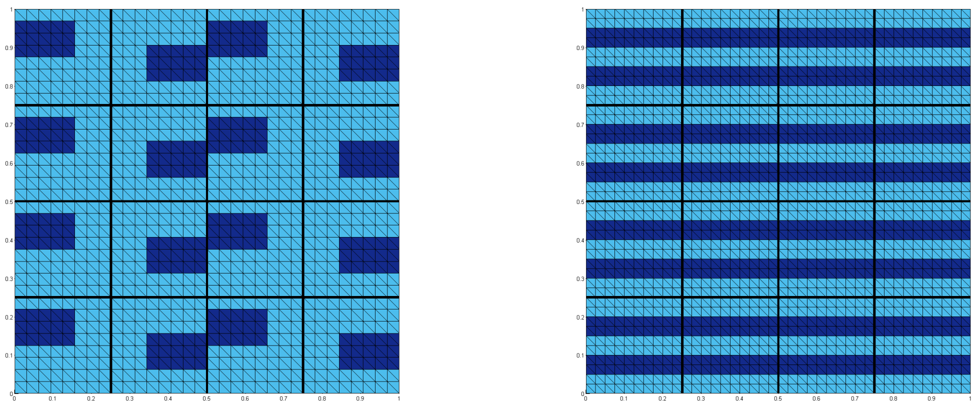


Figure 3.4: Two exemplary heterogeneous coefficient distributions for a stationary diffusion problem on the unit square decomposed into 4×4 subdomains. In both cases, dark blue corresponds to the high coefficient ($\rho = 1e6$) and light blue corresponds to the low coefficient ($\rho = 1$). **Left:** Coefficient distribution with shifted boxes associated with a high coefficient. Visualization for $H/h = 8$. **Right:** Coefficient distribution with two straight channels crossing each subdomain associated with a high coefficient. Visualization for $H/h = 10$.

solver. Moreover, adaptive coarse spaces are built in a local fashion and exploit the parallel structure of the underlying domain decomposition. In this thesis, we will focus on a very specific adaptive, i.e., problem-dependent coarse space which has successfully been applied to FETI-DP and BDDC for various heterogeneous model problems [92, 107, 108, 119, 130, 132, 148]. In Section 3.5.1, we will provide a motivation for the specific construction of this adaptive coarse space and describe the corresponding local eigenvalue problems as well the computation of the respective adaptive coarse constraints in more detail. In Section 3.5.2, we will cite the corresponding condition number estimate which does not depend on the jump of the discontinuous material parameters.

To conclude this section, let us note that, in general, different approaches to implement coarse space enrichments for FETI-DP and BDDC exist. Common approaches are a deflation or balancing approach [108, 113] and a transformation-of-basis approach [112, 116]. In this thesis, the deflation and the balancing approach is only applied to the FETI-DP method since using deflation for the BDDC method is not equivalent to the BDDC using a transformation of basis; see [113]. Thus, we use a generalized transformation-of-basis approach to enhance additional coarse constraints for the BDDC method; see [94]. Details on the implementation for both the balancing approach and the approach using a transformation of basis will be presented in Chapter 4.

3.5.1 Computing adaptive constraints based on local generalized eigenvalue problems

In this section, we present a very specific adaptive FETI-DP coarse space which is based on the work in [92, 107, 108, 119, 130, 132, 148]. Let us note that even though this adaptive

	adaptive			classic weighted avg.			frugal		
H/h	# c.	cond	it	# c.	cond	it	# c.	cond	it
stationary diffusion									
8	4	3.60	12	24	61 559.3	20	4	3.60	12
16	4	3.95	13	24	99 656.1	24	4	3.96	13
32	4	5.02	15	24	1.1775e05	26	4	5.04	15

Table 3.1: Dimensions of the coarse space (# c.), condition numbers (cond) and iteration numbers (it) for different FETI-DP coarse spaces for a stationary diffusion problem on the unit square with 4×4 subdomains for the coefficient distribution as in Fig. 3.4 (left). Homogeneous Dirichlet boundary conditions on the left side of the unit square. The higher coefficient is $1e6$ and the lower coefficient is 1. Table already published in [73, Table 1.1].

	adaptive			classic weighted avg.			frugal		
H/h	# c.	cond	it	# c.	cond	it	# c.	cond	it
stationary diffusion									
10	24	1.04	2	24	9 508.5	12	12	9 508.4	9
20	24	1.15	3	24	9 607.1	12	12	9 606.1	11
33	24	1.25	3	24	9 648.2	13	12	9 648.1	12

Table 3.2: Dimensions of the coarse space, condition numbers and iteration numbers for different FETI-DP coarse spaces for a stationary diffusion problem on the unit square with 4×4 subdomains for the coefficient distribution as in Fig. 3.4 (right). Homogeneous Dirichlet boundary conditions on the left side of the unit square. The higher coefficient is $1e6$ and the lower coefficient is 1. See Table 3.1 for the column labeling.

coarse space has also successfully been applied to the BDDC method, here, we will focus on its presentation for FETI-DP. To motivate this approach, we will recapitulate some relations between the Rayleigh quotient of the preconditioned FETI-DP system and the operator P_D as introduced in Section 3.2.2.

Parts of this chapter have already been published in modified or unmodified form in [72, 73, 75]. Additionally, some passages of the motivating part of this section are also closely related to the presentation in [119].

In general, in order to guarantee the convergence and robustness of the iterative solver for the FETI-DP system, it is necessary to obtain an upper bound for the condition number of the preconditioned FETI-DP system matrix, which is problem- and coarse space-dependent. As mentioned in Sections 3.3.2 and 3.5.2, we can derive a relation between the spectral condition number estimate of the FETI-DP method and an estimate of the P_D-operator; cf. especially (3.12) and the related explanations. This yields that we can reduce the problem of computing an upper bound for the spectral condition number

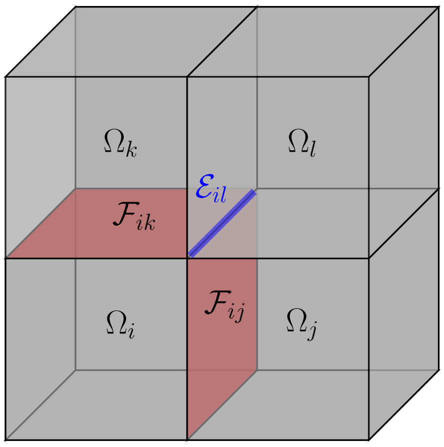


Figure 3.5: Exemplary visualization of four subdomains in three spatial dimensions sharing an edge (marked in blue) in a regular partition. The subdomain Ω_i shares a face $\mathcal{F}_{ij}$ and $\mathcal{F}_{ik}$ (marked in red) with Ω_j and Ω_k, respectively, but only an edge $\mathcal{E}_{il}$ with Ω_l.

to the problem of finding a constant $C \in \mathbb{R}$ such that

$$\frac{|P_D w|^2_{\widetilde{S}}}{|w|^2_{\widetilde{S}}} \leq C \text{ for all } w \in \widetilde{W}; \tag{3.21}$$

see also the references given in Section 3.2.2. Using this relation, a straightforward and intuitive approach to bound the constant C in (3.21) from above would be to consider the generalized eigenvalue problem of the form

$$\langle P_D v, \widetilde{S} P_D w \rangle = \mu \langle v, \widetilde{S} w \rangle \tag{3.22}$$

for all $v \in \widetilde{W} = \text{range}(\widetilde{S})$. Under the assumption that we have an a priori, i.e., non-adaptive coarse space, which ensures the invertibility of all local subdomain problems, the Schur complement $\widetilde{S}$ in (3.22) is symmetric positive definite and thus $0 \leq \mu \leq \infty$ holds for all eigenvalues of (3.22). Let us assume that the eigenvalues are ordered in a nondescending order $0 \leq \mu_1 \leq \ldots \leq \mu_n$ and let us denote the corresponding eigenvectors by $w_1, \ldots, w_n$. Then, we can select a user-defined tolerance TOL and choose all eigenvectors $w_i, i \in \{1, \ldots, m\}$, whose corresponding eigenvalues μ_i are greater than or equal to TOL to construct a specific constant for the estimation in (3.21). More formally, let the index k be given such that $\mu_k \geq TOL$ and $\mu_{k-1} < TOL$. By defining the matrix

$$U := \left(B_D \widetilde{S} P_D w_k, \ldots, B_D \widetilde{S} P_D w_n\right) \tag{3.23}$$

which depends on the eigenvectors whose eigenvalues are equal to or greater than TOL, we obtain the estimate

$$\frac{|P_D w|^2_{\widetilde{S}}}{|w|^2_{\widetilde{S}}} \leq \hat{C} \cdot TOL \text{ for all } w \in \widetilde{W}_U := \{w \in \widetilde{W} : U^T B w = 0\}, \tag{3.24}$$

where the constant $\hat{C}$ only depends on geometrical constants. This is valid since the eigenvectors can be chosen to be orthogonal with respect to the inner products defined by $\langle \cdot, P_D^T \widetilde{S} P_D \cdot \rangle$ and $\langle \cdot, \widetilde{S} \cdot \rangle$, and can be proven by arguments from standard linear algebra as well as by using (3.22).

So far, we have given a motivation why it is intuitive to consider the generalized eigenvalue problem (3.22) related to the P_D-operator in order to derive a condition number bound for the FETI-DP method. However, in the context of domain decomposition methods, the solution of (3.22) is not feasible since this equation represents a global eigenvalue problem for the entire domain Ω. Therefore, instead of solving (3.22) globally, we make use of the substructures of the nonoverlapping domain decomposition and derive local versions of the generalized eigenvalue problem (3.22). In particular, we will consider local generalized eigenvalue problems on edges and/or faces of the domain decomposition. This drastically reduces the number of subdomains which are affected by each eigenvalue problem such that the solution of the local eigenvalue problems can be parallelized (to a certain extent) quite well.

Let us now describe in detail the definition of the local eigenvalue problems as introduced in [130, 132] and provide some necessary notation. Note again that for both stationary diffusion as well as linear elasticity problems we assume the existence of an a priori, i.e., nonadaptive coarse space that ensures the invertibility of the local problems of each subdomain, e.g., we assume that all vertices are chosen as primal variables. The following description is based on [73] and [119, 148]. Since, in this thesis, we consider both two- and three-dimensional domains $\Omega \subset \mathbb{R}^d, d = 2, 3$, we formulate the following equations generically such that they are equally valid for both cases.

Let us consider $\mathcal{X}_{ij} \subset \partial\Omega_i \cap \partial\Omega_j$, e.g., $\mathcal{X}_{ij}$ could be a face $\mathcal{F}_{ij}$ or an edge $\mathcal{E}_{ij}$ between the two neighboring subdomains Ω_i and Ω_j; cf. also Fig. 3.5 for a visualization of four cubic subdomains sharing an edge. Then, for each equivalence class $\mathcal{X}_{ij}$ between two neighboring subdomains Ω_i and Ω_j, a single eigenvalue problem has to be solved. To mathematically formulate the respective local eigenvalue problem, we first introduce the local restriction of the jump matrix B to the equivalence class $\mathcal{X}_{ij}$. We define

$$B_{\mathcal{X}_{ij}} := \left(B_{\mathcal{X}_{ij}}^{(i)},\ B_{\mathcal{X}_{ij}}^{(j)} \right)$$

as the submatrix of $\left(B^{(i)}, B^{(j)}\right)$ with the rows consisting of exactly one 1 and one -1 and being zero elsewhere. Analogously, we denote by

$$B_{D,\mathcal{X}_{ij}} := \left(B_{D,\mathcal{X}_{ij}}^{(i)},\ B_{D,\mathcal{X}_{ij}}^{(j)} \right)$$

the corresponding scaled submatrix of $\left(B_D^{(i)}, B_D^{(j)}\right)$, i.e., the scaled variant of $B_{\mathcal{X}_{ij}}$. We then define

$$S_{ij} := \begin{pmatrix} S^{(i)} & 0 \\ 0 & S^{(j)} \end{pmatrix} \in \mathbb{R}^{(n_i+n_j)\times(n_i+n_j)} \tag{3.25}$$

with $S^{(i)}$ and $S^{(j)}$ being the local Schur complements of the local stiffness matrices $K^{(i)}$ and $K^{(j)}$, respectively, with respect to the interface variables and $n_l, l \in \{i, j\}$, the number of degrees of freedom on the local part of the interface. We further define

$$P_{D_{ij}} := B_{D,\mathcal{X}_{ij}}^T B_{\mathcal{X}_{ij}}$$

as a local version of the jump operator $P_D = B_D^T B$. Then, according to [92, 108, 119, 130, 132, 148], one has to solve the following generalized eigenvalue problem for each equivalence class $\mathcal{X}_{ij}$: Find $w_{ij} \in (\operatorname{Ker} S_{ij})^\perp$ such that

$$\langle P_{D_{ij}} v_{ij},\ S_{ij} P_{D_{ij}} w_{ij} \rangle = \mu_{ij} \langle v_{ij},\ S_{ij} w_{ij} \rangle \quad \forall v_{ij} \in (\operatorname{Ker} S_{ij})^\perp . \tag{3.26}$$

Let us briefly comment on the choice of the specific subspace $(\operatorname{Ker} S_{ij})^\perp$. A more straightforward localized version of the eigenvalue problem given in (3.22) is the following formulation: Find $w_{ij} \in \mathbb{R}^{(n_i+n_j)}$ such that

$$\langle P_{D_{ij}} v_{ij},\ S_{ij} P_{D_{ij}} w_{ij} \rangle = \mu_{ij} \langle v_{ij},\ S_{ij} w_{ij} \rangle \quad \forall v_{ij} \in \mathbb{R}^{(n_i+n_j)}. \tag{3.27}$$

However, this formulation suffers from the difficulty that neither the left-hand side nor the right-hand side operator is positive definite. Since the local Schur complement matrices originate from the local stiffness matrices, we know that both operators are at least symmetric positive semidefinite. In particular, the null space of the matrix S_{ij} is given by the single rigid body modes of the two subdomain interfaces; cf. also the related explanations at the beginning of Section 3.2.1. As assumed for the a priori coarse space, we couple the two subdomains Ω_i and Ω_j in the primal vertices. However, if neither of the two subdomains have direct contact to the Dirichlet boundary on an essential part of their boundary, i.e., $\partial\Omega_D \cap (\partial\Omega_i \cup \partial\Omega_j) = \emptyset$, the common rigid body modes are still in the null space of the coupled right-hand side operator; cf. also [92, 119]. Thus, it is necessary to remove the common rigid body modes in (3.27) and consider the respective eigenvalue problem in the subspace $(\operatorname{Ker} S_{ij})^\perp$ as it is done in (3.26).

For an explicit expression of the positive definite right-hand side operator on the subspace $(\operatorname{Ker} S_{ij})^\perp$, two orthogonal projection matrices Π_{ij} and $\overline{\Pi}_{ij}$ are used to control the single and common rigid body modes of the two neighboring subdomains Ω_i and Ω_j; see, e.g., [108, 119, 148]. Let us briefly introduce some additional notation to define the two projections without going into all the technical details. The following derivation of the projections is closely related to the presentations in [108, 119, 148] to which we also refer for more technical details.

For the derivation of the two projection matrices, we denote by $\widetilde{W}_{ij}$ the space of functions in the product space $W_i \times W_j$ that are continuous in the primal variables which are shared by the subdomains Ω_i and Ω_j. We further denote by Π_{ij} the l_2-orthogonal projection from $W_i \times W_j$ to $\widetilde{W}_{ij}$. Additionally, we introduce a second l_2-orthogonal projection from $W_i \times W_j$ to $\operatorname{range}(\Pi_{ij} S_{ij} \Pi_{ij} + \sigma(I - \Pi_{ij}))$ which we denote by $\overline{\Pi}_{ij}$. Here, σ is a positive constant used for stability reasons and can be chosen as, e.g., the maximum value of the diagonal entries of S_{ij}; see [130, 132]. Let us now describe how to build the two projection matrices Π_{ij} and $\overline{\Pi}_{ij}$. Note that both projection matrices are set up such that they are symmetric.

We first define the matrix $R_{ij}^{(l)T}, l = i, j$, as the local part of the assembly operator of primal variables on $\partial\Omega_i \cap \partial\Omega_j$ and as the identity on the rest of $\big(\Gamma \cap \partial\Omega_i\big) \times \big(\Gamma \cap \partial\Omega_j\big)$. We then obtain

$$R_{ij} := \begin{pmatrix} R_{ij}^{(i)} \\ R_{ij}^{(j)} \end{pmatrix}$$

and can thus define the orthogonal projection onto $\widetilde{W}_{ij}$ by

$$\Pi_{ij} := R_{ij}(R_{ij}^T R_{ij})^{-1} R_{ij}^T. \tag{3.28}$$

For the definition of $\overline{\Pi}_{ij}$ we make use of the fact that $I - \overline{\Pi}_{ij}$ is an orthogonal projection onto the rigid body modes or, in general, the null space, which is continuous on $W_i \times W_j$. Under the assumption that $\{\widetilde{r}_1, \ldots, \widetilde{r}_k\}$ is the largest set of linear independent rigid body modes that are continuous on $W_i \times W_j$, we use a modified Gram-Schmidt method to create an orthonormal basis $\{r_1, \ldots, r_k\}$ and define the projection

$$\overline{\Pi}_{ij} := I - \sum_{p=1}^{k} r_p r_p^T. \tag{3.29}$$

We can then establish and solve the following generalized eigenvalue problems of the form

$$\begin{aligned} &\overline{\Pi}_{ij}\Pi_{ij}P_{D_{ij}}^T S_{ij} P_{D_{ij}} \Pi_{ij}\overline{\Pi}_{ij} w_{ij} \\ &\qquad = \mu_{ij}(\overline{\Pi}_{ij}(\Pi_{ij} S_{ij} \Pi_{ij} + \sigma(I - \Pi_{ij}))\overline{\Pi}_{ij} + \sigma(I - \overline{\Pi}_{ij}))w_{ij}. \end{aligned} \tag{3.30}$$

One would then select all eigenvectors $w_{ij}^l, l = 1, \ldots, L$ belonging to eigenvalues μ_{ij}^l, $l = 1, \ldots, L$, which are larger than or equal to a user-defined tolerance TOL and enforce the constraints

$$w_{ij}^{lT} P_{D_{ij}}^T S_{ij} P_{D_{ij}} w_{ij} = c_{ij}^{lT} B_{\mathcal{X}_{ij}} w_{ij} = 0, \tag{3.31}$$

for given constraint vectors

$$c_{ij}^l := B_{D,\mathcal{X}_{ij}} S_{ij} P_{D_{ij}} w_{ij}^l, \; l = 1, \ldots, L, \tag{3.32}$$

e.g., with a projector preconditioning or a transformation-of-basis approach. To provide an illustrative motivation for the specific form of (3.30) let us summarize again that the purpose of the two introduced projection matrices is to obtain a right-hand side of the eigenvalue problem in (3.30) which is symmetric positive definite; cf. [130]. For this purpose, the projection Π_{ij} removes the rigid body modes or, more generally, the null space of each of the single subdomains Ω_i and Ω_j while $I - \overline{\Pi}_{ij}$ is an orthogonal projection onto the space of rigid body modes that are continuous on $W_i \times W_j$ and move Ω_i and Ω_j as a connected entity; see also [119, Sect. 5.2].

Finally, let us comment on some special aspects of the eigenvalue problem in (3.30) for the three-dimensional case. Usually, in case of a three-dimensional model problem, the eigenvalue problems in adaptive FETI-DP are defined for closed faces, i.e., we have

$\mathcal{X}_{ij} = \overline{\mathcal{F}}_{ij}$ for the eigenvalue problem of a face which is shared by the two subdomains Ω_i and Ω_j. As proposed in [132] and [119], in our implementation, we split the computed face constraint vectors into a part related to the open face $\mathcal{F}_{ij}$ as well as into several edge parts which lay on the boundary of the respective face. Thus, in case of a regular domain decomposition into cubes, we would obtain one face constraint and four edge constraints from a single eigenvector defined on a closed face. For more details, we refer to, e.g., [119, Sect. 5.2.1]. Additionally, in three dimensions, we also have to control the jump $w^{(i)} - w^{(l)}$ across an edge $\mathcal{E}_{il}$ for two subdomains Ω_i and Ω_l that only share an edge but not a face; see also Fig. 3.5. This means that, for three-dimensional problems, we also have to solve a number of edge eigenvalue problems in addition to the eigenvalue problems on faces. In particular, we have to solve an additional edge eigenvalue problem for all edges with multiplicity greater than three; cf. also [119, Sect. 5.2.1]. This variant ensures a sound theoretical condition number bound for the obtained adaptive FETI-DP coarse space; see [92]. According to numerical experiments presented in [149] and [119], however, the respective eigenvalue problems are usually related to only a small number of edges or a slightly higher number of short edges. Therefore, the additional effort for the setup and the solution of these edge eigenvalue problems is rather small. This will become important again in Chapter 8 of this thesis, where we propose to train neural networks to automatically identify edges or faces, where the computation of adaptive constraints is necessary.

3.5.2 Condition number bound

Analogously to classic FETI-DP and BDDC methods, the spectral condition number for the adaptive FETI-DP method presented in Section 3.5.1 is also closely related to an estimate of the P_D-operator; see also (3.12) for this relation.

In this section, we present the respective condition number bounds in two and three spatial dimensions for the adaptive FETI-DP coarse space as described in Section 3.5.1. For the technical details and a detailed proof of the cited condition number bounds we refer to the literature. For both a two- and a three-dimensional domain Ω, enhancing the FETI-DP and BDDC coarse space with the adaptive coarse constraints in (3.31), we are able to derive a condition number bound which is of the form

$$\kappa(\widetilde{M}^{-1}F) \leq \widetilde{C} \cdot TOL \tag{3.33}$$

with $\widetilde{C}$ independent of H and h; see [92,108,130,132]. In particular, the constant $\widetilde{C}$ does only depend on geometric constants of the domain decomposition, i.e., on the maximum number of edges of a subdomain in two dimensions or on the maximum number of faces of a subdomain and the maximum multiplicity of an edge in three dimensions, but is independent of the contrast of the coefficient. This is a very desirable property since it ensures a robust convergence of the algorithm even for highly heterogeneous coefficient or material distributions.

For the sake of completeness, let us briefly present the concrete constants for the two- and the three-dimensional case. In two dimensions, we can derive the following estimate of the P_D-operator for the FETI-DP algorithm with adaptive constraints as defined in Section 3.5.1:

$$|P_D w|^2_{\widetilde{S}} \leq N^2_{\mathcal{E}} \cdot TOL \; |w|^2_{\widetilde{S}} \text{ for all } w \in \widetilde{W}_U, \tag{3.34}$$

where $N_{\mathcal{E}}$ denotes the maximum number of edges of a subdomain. Using the relation in (3.12), this yields the respective condition number bound

$$\kappa(\widetilde{M}^{-1}F) \leq N^2_{\mathcal{E}} \cdot TOL \tag{3.35}$$

with $\widetilde{M}^{-1}$ being either the projector or the balancing preconditioner; see also Chapter 4 for more details on balancing and deflation preconditioners. For a detailed proof of the cited condition number bound in two dimensions, we refer to [148, Theorem 3.3.1].

In three dimensions, the argumentation is - in principle - similar although the proof of the estimate for the operator P_D is more complex. In this case, we can derive the estimate

$$|P_D w|^2_{\widetilde{S}} \leq 4 \max\{N_{\mathcal{F}}, N_{\mathcal{E}} M_{\mathcal{E}}\}^2 \cdot TOL \; |w|^2_{\widetilde{S}} \text{ for all } w \in \widetilde{W}_U, \tag{3.36}$$

where $N_{\mathcal{F}}$ denotes the maximum number of faces of a subdomain, $N_{\mathcal{E}}$ the maximum number of edges of a subdomain, and $M_{\mathcal{E}}$ the maximum multiplicity of an edge; see also [119, Lemma 5.5]. Using (3.36) and again the relation in (3.12), this yields the condition number bound

$$\kappa(\widetilde{M}^{-1}F) \leq 4 \max\{N_{\mathcal{F}}, N_{\mathcal{E}} M_{\mathcal{E}}\}^2 \cdot TOL \tag{3.37}$$

with $\widetilde{M}^{-1}$ being again the balancing or the deflation preconditioner; see also [119, Theorem 5.7] for a detailed proof.

3.6 Summarizing remarks with respect to robustness and computational effort

At the end of this chapter on classic and adaptive coarse spaces, let us summarize some of the main observations. For both, the FETI-DP and the BDDC method, the design of an appropriate coarse space is at the heart of the respective algorithm. On the one hand, the coarse space ensures the fast global transport of information between the different subdomains and thus the fast convergence towards a continuous global solution. At the same time, in a parallel implementation, the solution of the coarse problem can become a scaling bottleneck. Thus, in principle, we are interested in a coarse space which is preferably small. On the other hand, we want our DD algorithm to be robust, i.e., to show a stable convergence behavior also for highly heterogeneous coefficient functions. In theory, this is ensured by an upper bound of the condition number, serving as a worst case estimate, which is independent of the coefficient distribution.

In classic FETI-DP and BDDC methods, the coarse space usually consists of primal vertices and/or classic averages along edges or faces of the domain decomposition. Additionally, for linear elasticity problems, first order moments play an important role. As a result, classic coarse spaces are usually relatively small. However, they are only robust under some restrictive assumptions on the coefficient function. As we have seen in Sections 3.2.2 and 3.3.2, for completely arbitrary coefficient distributions, the constants in the condition number estimates depend on the contrast of the coefficient function for classic FETI-DP and BDDC methods. Consequently, for a high contrast or high jumps in the coefficient function, the convergence behavior of classic coarse spaces will deteriorate and we obtain condition numbers in the dimension of the coefficient contrast. In Section 3.5, we have introduced an adaptive, i.e., problem-dependent coarse space which uses certain eigenmodes to enhance the classic coarse space. As a result, we can derive a condition number bound which exclusively depends on some geometrical constants of the domain decomposition but not on the specific coefficient distribution. This results in a robust convergence behavior also for high contrasts in the coefficient distributions; cf. also the motivating example in Table 3.2. Thus, the presented adaptive algorithm fulfills our desire to construct a robust coarse space.

However, as a drawback, the adaptive FETI-DP coarse space presented in Section 3.5 requires the setup and the solution of certain local eigenvalue problems on edges and/or faces of the DD. It should be stated that, in a parallel FETI-DP implementation, the solution of these eigenvalue problems can be distributed to the compute cores, due to the local structure of the eigenvalue problems. Nonetheless, usually more than a single eigenvalue problem has to be solved on a single compute core. Especially for three-dimensional computational domains, the subsequent assembly and solution of several eigenvalue problems can take up the larger part of the total time to solution; see also [96]. In particular, the assembly of the local eigenvalue problems includes the computation of local Schur complements, see (3.25), which often takes up a significant part of the total computing time. Additionally, the communication of Schur complements, which is in general necessary, can put a high pressure on the network of the parallel computer. Taking all the above observations into account, it is desirable to develop strategies to reduce the number of eigenvalue problems which have to be solved, e.g., by discarding eigenvalue problems which do not result in any new constraints for a given tolerance value TOL. Alternatively, we can try to develop fundamentally different, i.e., new approaches which compute small and robust coarse spaces.

In [93, 94, 107], different heuristic approaches to reduce the number of necessary eigenvalue problems have been presented, which are - roughly speaking - based on the identification of coefficient jumps on the considered edge or face as well as on the residual after one step of the adaptive FETI-DP or BDDC method. We refer to the experiments in [93, 94, 107] for more details. In [96, 119], these concepts have been extended with regard to a parallel implementation. In particular, the authors have presented ideas to improve the load balance of the algorithm by using static and asynchronous dynamic processes to achieve a more balanced distribution of the local eigenvalue problems.

In this thesis, we present different approaches of how to design coarse spaces which are preferably robust and small, i.e., efficient. Let us note that, usually, we have to make a trade-off between both desired properties. In Chapter 5, we present an approach which mainly focuses on delaying the bottleneck which is caused by the exact solution of the coarse problem in a parallel implementation. This is achieved by introducing a third level of the domain decomposition. Then, only a coarse problem on the coarsest level has to be solved, which is usually much smaller. In Chapter 6, we introduce a frugal approach which builds a coarse space that is relatively small but at the same time robust for a wider range of different model problems. This coarse space is inspired by the adaptive FETI-DP coarse space in Section 3.5 and can be interpreted as a low-dimensional approximation of it. Finally, in Chapter 8, we develop an alternative approach using concepts from scientific machine learning. Here, neural networks are trained to automatically identify critical edges or faces where the solution of an eigenvalue problem is necessary to obtain a robust coarse space. Thus, this method aims to obtain robustness for heterogeneous coefficient distributions while, at the same time, maintaining a relatively small size of the resulting coarse space.

4 Implementation of coarse space enrichments for FETI-DP and BDDC

In this chapter, we describe different approaches to implement coarse space enrichments for both FETI-DP and BDDC domain decomposition methods. As already discussed in Chapter 3, different approaches to enrich the FETI-DP or BDDC coarse space by additional constraints exist. The construction of classic coarse spaces was already mentioned in Sections 3.2 and 3.3. Additionally, a detailed mathematical description of classic weighted averages was given in Section 3.4. In Section 3.5, we have introduced a very specific adaptive coarse space which basically relies on the solution of certain localized eigenvalue problems and uses selected eigenvectors to enhance the FETI-DP or BDDC coarse space. On top of that, we introduce a heuristic coarse space which we name frugal coarse space and which is numerically tested in Chapter 6. This coarse space is strongly motivated by the adaptive coarse space presented in Section 3.5 and can be interpreted as a low-dimensional approximation of it. In all cases named above, in general, different approaches of how to implement these coarse space enrichments exist. Parts of this chapter have already been published in modified or unmodified form in [72].

One way to enforce additional coarse constraints is by using a projector preconditioning approach or a balancing approach. The technique of projector preconditioning is also known as deflation; see, e.g., [45, 139]. In the context of domain decomposition methods the approach of deflation and balancing has already been used extensively and for a wider range of different model problems and coefficient or material distributions; see, e.g., [59, 85, 113, 136]. In this thesis, the deflation and the balancing approach are only applied to the FETI-DP method since using deflation for the BDDC method is not equivalent to the BDDC method using a transformation of basis; see [113].

Thus, to enhance additional coarse constraints for the BDDC method, we use a generalized transformation-of-basis approach; see [94]. In principle, in a standard transformation-of-basis approach, the nodal basis of the finite element method is transformed such that general constraints can be enforced by a partial nodal assembly; see, e.g., [94,109,116,117, 125]. In particular, under the assumption of a constant scaling on any face and any edge, as, e.g., valid for a multiplicity-scaling, the standard transformation-of-basis approach can be proven to be equivalent to a corresponding deflation approach; see [113]. For more general, i.e., arbitrary coefficient distributions and scalings, a generalized transformation-of-basis approach has been proposed in [94, 95]; see also [119] for more technical details.

However, as a drawback, explicitly transformed local stiffness matrices can become dense when using face constraints in three dimensions. To obtain an efficient parallel implementation, we instead use the equivalent approach of local saddle point problems as suggested in [116, subsection 4.2.2] for our numerical parallel results for a frugal BDDC coarse space in Section 6.5. This technique will be presented at the end of this chapter.

4.1 Projector preconditioning and balancing

Let us now describe in more detail how to implement additional coarse constraints using projections; see, e.g., [85, 113]. This section is partly based on the already published description in [72]. Additionally, we will recall the notation for projector preconditioning and deflation as used in [148] and [119]. For more details on projection methods in the context of Krylov subspace methods, see [45, 136, 139].

The starting point of the projector preconditioning approach is to aggregate a set of additional primal constraints, as, e.g., averages, first order moments over certain edges, or constraint vectors obtained from certain eigenvectors, as columns of a rectangular matrix U; see, e.g., [85, 113]. Then, in each iteration of the PCG method, we enforce the constraint

$$U^T B u = 0. \tag{4.1}$$

Let us note again that, in analogy to the description in Section 3.5.1 and [119], in the context of deflation and balancing we refer to $c^T B w = 0$ as a *constraint* while we refer to the vector c as a *constraint vector*. Thus, we can define the matrix U in (4.1) as

$$U = (c_1, \ldots, c_m) \tag{4.2}$$

for given constraint vectors $c_i, i = 1, \ldots, m$. To enforce the respective constraints we introduce the F-orthogonal projection onto the range of U as

$$P = U(U^T F U)^{-1} U^T F$$

if F is symmetric positive definite, or in the more general form

$$P = U(U^T F U)^{+} U^T F$$

if F is symmetric positive semidefinite and $U^T F U$ is singular. Here, $(U^T F U)^+$ denotes a pseudoinverse of the matrix $U^T F U$; cf., e.g., [14]. For the resulting projection P, we have $\text{range}(P) = \text{range}(U(U^T F U)^+)$ and $\ker(P) = \ker((U^T F U)^+ U^T F)$.

We then multiply the FETI-DP system $F\lambda = d$ (see (3.6)) by $(I - P)^T$ which yields the deflated and singular but consistent system

$$(I - P)^T F \lambda = (I - P)^T d. \tag{4.3}$$

For $\lambda \in \text{range}(U)$, we also have $\lambda \in \ker((I - P)^T F)$ since

$$\begin{aligned}(I - P)^T F \lambda &= F\lambda - P^T F \lambda = FU\widehat{\lambda} - P^T F U \widehat{\lambda} \\ &= FU\widehat{\lambda} - FU(U^T F U)^+ (U^T F U)\widehat{\lambda} \\ &= 0,\end{aligned}$$

with $U\widehat{\lambda} = \lambda$. This yields

$$U^T (I - P)^T F \lambda = 0$$

and since U has full column rank, we obtain

$$\lambda \in \ker((I-P)^T F).$$

Similarly, for $\lambda \in \ker((I-P)^T F)$ and F nonsingular, we have

$$(I-P)^T F\lambda = (F - FU(U^T FU)^{-1}U^T F)\lambda = 0$$

which yields

$$\lambda = U(U^T FU)^{-1}U^T F\lambda$$

and we obtain that $\lambda \in \text{range}(U)$. We thus have the relation

$$\ker((I-P)^T F) = \text{range}(U).$$

Let λ^* be the exact solution of the original system $F\lambda = d$ and λ the solution of the preconditioned system

$$M_D^{-1}(I-P)^T F\lambda = M_D^{-1}(I-P)^T d,$$

where M_D^{-1} is the Dirichlet preconditioner. We then define

$$\overline{\lambda} = U(U^T FU)^{-1}U^T d = PF^{-1}d$$

for invertible F or

$$\overline{\lambda} = PF^{+}d$$

for any pseudoinverse F^{+}, respectively. We can then finally compute the solution of the original system in (3.6) as

$$\lambda^* = \overline{\lambda} + (I-P)\lambda \in \ker\,(I-P) \oplus \text{range}\,(I-P).$$

The matrices $P^T F\,(= FP)$ and $(I-P)^T F\,(= F(I-P))$ are symmetric and the spectrum is thus not changed by projecting the correction onto range$(I-P)$ in each iteration; cf. [113]. Therefore, we include the projection $(I-P)^T$ into the preconditioner and obtain the symmetric projector preconditioner

$$M_{PP}^{-1} = (I-P)M_D^{-1}(I-P)^T$$

which can also be denoted as deflation preconditioner. We then solve the original system applying the preconditioner M_{PP}^{-1}, i.e., we solve the system

$$M_{PP}^{-1}F\lambda = M_{PP}^{-1}d.$$

The solution λ of this system is in the subspace range$(I-P)$. Adding the correction, we compute the solution λ^* of the original system by

$$\lambda^* = \overline{\lambda} + \lambda.$$

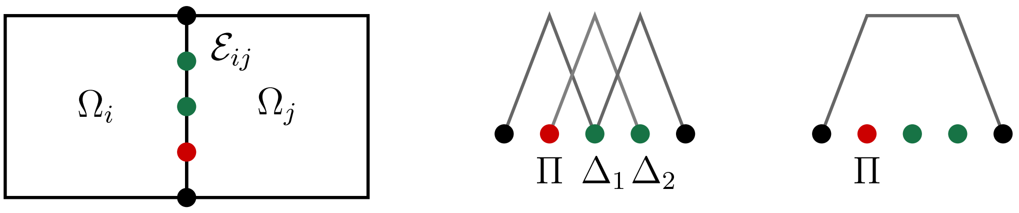


Figure 4.1: Transformation of basis for a primal edge average. **Left:** Two subdomains Ω_i and Ω_j sharing the edge $\mathcal{E}_{ij}$ which consists of two dual nodes (marked in green) and one primal edge node (marked in red). **Middle:** Nodal basis functions for all three edge nodes before the transformation of basis. **Right:** Transformed basis function for the primal edge node Π after the transformation of basis. The edge average constraint is enforced at the primal degree of freedom (marked in red).

An alternative approach is the inclusion of the computation of $\overline{\lambda}$ into the preconditioner. This results in the balancing preconditioner

$$M_{BP}^{-1} = (I - P)M_D^{-1}(I - P)^T + PF^{-1}. \tag{4.4}$$

Since $PF^{-1} = U(U^T FU)^{-1}U^T$, this preconditioner is symmetric and can be computed efficiently.

Let us note that, depending on the number of additional primal constraints, $U^T FU$ is usually of much smaller dimension than F.

In this thesis, we always use the balancing preconditioner in our numerical experiments for the FETI-DP method and directly obtain the solution without computing an additional correction. All constraints added to the a priori vertex constraints are included using the matrix U.

4.2 The standard transformation-of-basis approach

In this section, we describe how to transform the basis of FETI-DP or BDDC methods such that weighted average constraints along edges or faces can be enforced by subassembly using a standard transformation-of-basis approach; see, e.g., [109, 113, 116, 117, 125]. Let us note that for this approach, we assume a constant scaling on any face or edge, as, e.g., multiplicity scaling or ρ-scaling for certain coefficient distributions. For more general scalings and coefficient distributions, an extended approach was proposed in [94, 95, 119], which is also referred to as generalized transformation of basis.

The starting point of the transformation-of-basis approach is the observation that continuity of the solution u across the subdomain boundary in certain degrees of freedom can be enforced for the corresponding finite element basis function using partial finite element assembly. Similarly, using a transformation of basis from a nodal finite element basis to a different basis, more general constraints can be enforced using the same technique.

For a more readable and simplified mathematical description of this technique, we will demonstrate the approach with a two-dimensional example as illustrated in Fig. 4.1

(left). We consider the two neighboring subdomains Ω_i and Ω_j which share the edge $\mathcal{E}_{ij}$. The edge $\mathcal{E}_{ij}$ consists of a primal node Π and two dual nodes Δ_1 and Δ_2; see also Fig. 4.1 (middle) for a visualization of the corresponding nodal basis functions. During the Krylov iteration of the iterative solution process, the iterates u should fulfill a constraint involving the nodes on the edge $\mathcal{E}_{ij}$ which should be enforced by assembling the primal edge node Π. To further simplify the following description, we additionally assume that this constraint is the only additional constraint which is implemented using a transformation of basis, and that this constraint is a weighted edge average. To implement this edge average, we transform our local stiffness matrices $K^{(l)}, l \in \{i,j\}$, and the right-hand sides $f^{(l)}, l \in \{i,j\}$, with a transformation matrix $T^{(l)}, l \in \{i,j\}$. In particular, the transformed stiffness matrices $\overline{K}^{(l)}$, the transformed load vectors $\overline{f}^{(l)}$, and the transformed variables $\overline{u}^{(l)}$ on Ω_l can be obtained by

$$\begin{aligned} \overline{K}^{(l)} &= T^{(l)T} K^{(l)} T^{(l)}, \\ \overline{f}^{(l)} &= T^{(l)T} f^{(l)}, \\ \text{and } \overline{u}^{(l)} &= T^{(l)T} u^{(l)},\ l \in \{i,j\}. \end{aligned} \tag{4.5}$$

The transformed stiffness matrices $\overline{K}^{(l)}$ and transformed right-hand sides $\overline{f}^{(l)}$ will then replace $K^{(l)}$ and $f^{(l)}$ in the FETI-DP and the BDDC algorithm; see also [105] for more details. In general, the transformation matrices $T^{(l)}, l \in \{i,j\}$ in (4.5) are defined edge by edge. Let us consider again the example in Fig. 4.1 and let us recall the above assumption that the constraint on $\mathcal{E}_{ij}$ is the only additional constraint which is implemented using a transformation of basis. We then denote by $T_{\mathcal{E}}^{(l)}$ the restriction of $T^{(l)}$ to the edge $\mathcal{E}_{ij}$ for $l \in \{i,j\}$. We further denote by c_u the normalized constraint vector for the transformation of basis which is defined on $\partial\Omega_i \cap \mathcal{E}_{ij}$ and $\partial\Omega_j \cap \mathcal{E}_{ij}$, and which is equal on both sets. Then, the respective constraint which should be enforced by the transformation of basis writes

$$c_u^T (u_{\mathcal{E}_{ij}}^{(i)} - u_{\mathcal{E}_{ij}}^{(j)}) = 0 \ \Leftrightarrow\ c_u^T u_{\mathcal{E}_{ij}}^{(i)} = c_u^T u_{\mathcal{E}_{ij}}^{(j)} \tag{4.6}$$

with $u_{\mathcal{E}_{ij}}^{(l)} = u_{|\partial\Omega_l \cap \mathcal{E}_{ij}}, l \in \{i,j\}$. For example, $c_u = \frac{1}{n_{\mathcal{E}}}(1,\ldots,1)^T$ corresponds to a continuous edge average along the edge $\mathcal{E}_{ij}$ shared by Ω_i and Ω_j, where $n_{\mathcal{E}}$ is the length of c_u. This corresponds to the use of a nonnodal basis function which is imposed on the primal variable on the edge $\mathcal{E}_{ij}$; see Fig. 4.1 (right) for a visualization of this transformed nonnodal basis function.

In a next step, we define the square transformation matrices $T_{\mathcal{E}}^{(l)}$ by

$$T_{\mathcal{E}}^{(l)} = \left(c_u, C_u^{(l)\perp}\right),\ l \in \{i,j\}, \tag{4.7}$$

where $C_u^{(l)\perp}$ is computed such that $T_{\mathcal{E}}^{(l)}$ is an orthogonal matrix. The matrix $T_{\mathcal{E}}^{(l)}$ describes the change of basis from the new to the original nodal basis. We then define the transformation matrix $T^{(l)}$ which acts on the entire subdomain $\Omega_l, l \in \{i,j\}$, by

$$T^{(l)} = \begin{bmatrix} I_I & 0 & 0 \\ 0 & I_V & 0 \\ 0 & 0 & T_{\mathcal{E}}^{(l)} \end{bmatrix}, \tag{4.8}$$

where I_I and I_V denote the identities on interior and on vertex variables, respectively.

Finally, after the transformation of basis has been performed, assembly in the new primal variables is used to enforce the given constraint. In other words, in the transformed basis, the constraints along edges can be handled as simple nodal constraints and an assembly can be performed to construct the coarse problem. Let us note that faces in three dimensions can be handled completely analogously to the already described procedure for edges. However, computing explicit transformations on faces negatively affects the sparsity pattern of the transformed stiffness matrices. In [116], a technique was proposed to reduce this effect which avoids affecting the sparsity pattern of the matrices $\overline{K}_{BB}^{(i)}$ by the explicit transformation of basis. We will explain this approach in more detail in the following section.

4.3 Efficient implementation of the transformation of basis using local saddle point problems

In general, the simplest approach to implement a transformation of basis into FETI-DP or BDDC is to transform the stiffness matrix and the right-hand side locally, i.e., to compute $\overline{f}^{(i)} = T^{(i)T} f^{(i)}$ and $\overline{K}^{(i)} = T^{(i)T} K^{(i)} T^{(i)}$ as described in Section 4.2. As already mentioned, as long as $T^{(i)}$ is sufficiently sparse, e.g., only edge constraints and no face constraints are used, this is a fast and very efficient option. However, using richer coarse spaces and denser transformation matrices $T^{(i)}$, the performance can decrease drastically due to the higher density of $\overline{K}^{(i)}$ compared to $K^{(i)}$. Thus, the time to solution will increase as well as the memory consumption. In our parallel BDDC implementation, which is used for the experiments presented in Chapter 5 and Section 6.5, we therefore have implemented an alternative technique which relies on the solution of local saddle point problems; see [116, Sect. 4.2.2].

Let us now briefly describe this strategy. In the application of the operator F, in case of FETI-DP, the local expressions $B_B^{(i)} \left(\overline{K}_{BB}^{(i)} \right)^{-1} \overline{v}_B^{(i)}$ have to be computed for a vector $\overline{v}_B^{(i)}$. This can be done locally by applying a direct solver to $\overline{K}_{BB}^{(i)}$, $i = 1, ..., N$, on each subdomain. However, if $\overline{K}_{BB}^{(i)}$ gets too dense, the performance of the direct solver is very poor. Alternatively, local saddle point problems can be solved; see [94, 116, 119].

At first, we observe that instead of minimizing

$$\overline{u}_B^{(i)T} \overline{K}_{BB}^{(i)} \overline{u}_B^{(i)}$$

we can equivalently consider the minimization of

$$\begin{bmatrix} \overline{u}_B^{(i)T} & 0 \end{bmatrix} \begin{bmatrix} \overline{K}_{BB}^{(i)} & \overline{K}_{\Pi B}^{(i)T} \\ \overline{K}_{\Pi B}^{(i)} & \overline{K}_{\Pi\Pi}^{(i)} \end{bmatrix} \begin{bmatrix} \overline{u}_B^{(i)} \\ 0 \end{bmatrix}, \tag{4.9}$$

where the values at the primal variables Π are set to zero. Then, instead of minimizing the bilinear form in (4.9) in the transformed variables $\overline{u}_B^{(i)}$, we introduce a corresponding constraint $Q^{(i)T}u^{(i)} = 0$ for the nontransformed variables, where the local matrices $Q^{(i)T}$ enforce the selected primal constraints. Thus, every time we have to solve a system with a local block from $\overline{K}_{BB}$, we instead solve the following saddle point problem

$$\begin{bmatrix} K_{II}^{(i)} & K_{\Gamma I}^{(i)T} & 0 \\ K_{\Gamma I}^{(i)} & K_{\Gamma\Gamma}^{(i)} & Q^{(i)} \\ 0 & Q^{(i)T} & 0 \end{bmatrix} \begin{bmatrix} u_I^{(i)} \\ u_\Gamma^{(i)} \\ \mu^{(i)} \end{bmatrix} = \begin{bmatrix} v_I^{(i)} \\ v_\Gamma^{(i)} \\ 0 \end{bmatrix}, \tag{4.10}$$

with additional local Lagrange multipliers $\mu^{(i)}$; cf. [116, Sect. 4.2.2] for more details. Note that the right-hand side $(v_I^{(i)T}, v_\Gamma^{(i)T})$ in (4.10) corresponds to $(\overline{v}_B^T, 0)$ transformed back to the original basis and restricted to the local subdomain; see also [119, Sect. 4.4]. Summarizing the above explanations, we replace the solution of the local problems with $\overline{K}_{BB}^{(i)}$ by the solution of much sparser saddlepoint systems in our parallel BDDC implementation. Additionally, it can also be quite expensive to explicitly compute the transformation matrix T and to explicitly compute Galerkin products of the type $\overline{K}^{(i)} = T^{(i)T}K^{(i)}T^{(i)}$. Therefore, all applications of the transformed matrices $\overline{K}_{\Gamma I}^{(i)}$, $\overline{K}_{\Pi B}^{(i)}$, and $\overline{K}_{\Pi\Pi}^{(i)}$ have to be replaced by equivalent formulations. Here, equivalent means that the solution of a system in the transformed solution space can be simply obtained by a multiplication of T^T with the respective solution in the untransformed space, and vice versa by a multiplication with T. Thus, the specific implementation does not affect the convergence of the iterative method but only the time to solution. More details on the implementation as well as a mathematical proof of the equivalence of the reformulation are given in [98].

To conclude this section, let us note that similar techniques of using local saddle point problems to enforce additional constraints have also been proposed in [61, 129, 130] by different authors. Moreover, in [51], global Lagrange multipliers are used in combination with saddle point problems, which is slightly different from our approach.

5 BDDC with a three-level preconditioner as an approximate coarse space

As we have seen in Chapter 3, both the FETI-DP and the BDDC method are divide-and-conquer algorithms and obtain their robustness and parallel scalability from an appropriate coarse space, i.e., a second level. Even though the coarse space in both methods is of crucial importance for the parallel scalability and ensures the approximation of a solution, which is continuous across the subdomains, it can also become a bottleneck in a parallel implementation. This means that the exact solution of the FETI-DP or BDDC coarse problem with a sparse direct solver can become a limiting factor for the parallel scalability of the respective method on large scales. As a consequence, during the last decade, different approximate variants of the BDDC and the FETI-DP method became popular, which were numerically tested for the solution of various linear and nonlinear partial differential equations [3, 4, 39, 41, 99, 100, 102, 110, 112, 126, 149, 150, 167, 168]. In principle, these methods differ from their respective exact methods by using an approximate solution of certain components of the preconditioner, most noticeably the coarse problem. While, as a drawback, computing only an approximative solution of the coarse problem can reduce the numerical robustness, it usually increases the scalability of the considered method significantly. One widely used approach of approximate coarse problems in the context of BDDC are *multilevel methods*; see, e.g., [4,131,161,167,168]. In this case, exact BDDC is applied recursively to the coarse problem leading to a new coarse problem of much smaller dimension at the highest level. Similarly, in other approximate BDDC variants, cycles of algebraic multigrid (AMG) are applied to the coarse problem; see, e.g., [39, 101, 126].

Here, we will focus on a very specific approximate BDDC coarse space which is based on the work in [166–168]. The proposed three-level BDDC preconditioner relies - as the name already suggests - on the introduction of a third level and approximates the original BDDC coarse space by the solution of a coarser problem on the highest level. To numerically test this three-level BDDC method and compare it to other approximate BDDC coarse spaces, we integrate it into our parallel software framework which has already been used in [101]. In particular, in [101], the authors have compared, in a common framework, several linear and nonlinear BDDC variants using AMG-based approximations of the coarse space, based on the BDDC formulation of [126] for linear problems. At the basis of this framework is a highly scalable PETSc-based [8–10] BDDC implementation, which applies BoomerAMG [80] for all AMG solves. For the results presented in this chapter, we have extended this implementation and also included the aforementioned three-level BDDC method in our framework as well as in our software package. Additionally, we have also implemented a vertex-based BDDC coarse space [41] in the same software to provide a broader range of comparison.

The outline of this chapter is as follows. At first, in Section 5.1, we will present an alternative of the original formulation of the BDDC preconditioner for the Schur complement system as introduced in Section 3.3.1. In particular, we will present an equivalent formulation of the BDDC method as a preconditioner for the fully assembled system $K_g u = f_g$, see, e.g., [126], which lays at the heart of our parallel software. In Section 5.2, we will then describe three different approximate BDDC preconditioners in a common framework, using the description of the exact BDDC preconditioner in Section 5.1 as a common basis. Here, we will especially focus on the three-level BDDC preconditioner which is similar to the proposed approach in [167, 168] but also formulated as a preconditioner for the fully assembled matrix K_g. Additionally, we will also briefly introduce the already mentioned BDDC preconditioner using AMG as well as a vertex-based preconditioner using a Gauss-Seidel method. We will then cite or prove, respectively, the condition number bounds for all three aforementioned approximate BDDC preconditioners in Section 5.3. In Section 5.4, we will provide some details of our parallel BDDC implementation which explicitly uses the same building blocks for all approximate BDDC preconditioners to ensure a fair comparison. Finally, in Section 5.5, we present numerical results for the three-level BDDC method for linear elasticity problems in three dimensions. In particular, we show comparative results to the approximate AMG-based coarse space which performed best in [101]. For the sake of completeness, we further show first parallel results for the vertex-based BDDC preconditioner [41].

Parts of this chapter have already been published in modified or unmodified form in [103]. Let us note that preliminary results for the three-level BDDC preconditioner have also been published in the master thesis [170]; which can be seen as a preparation of the publication [103]. However, prior to the publication [103], the implemented method has been revised algorithmically as well as with respect to several parallel implementational details to further increase the parallel scalability of the considered approach. Hence, the results presented here significantly differ from the results presented in [170].

5.1 An alternative formulation of the exact BDDC preconditioner for the assembled system

In Section 3.3.1, we have already presented a description of the original formulation of the BDDC preconditioner for the Schur complement system $S_g u_g = g_g$ where S_g is the assembled global Schur complement; cf. also (3.17). Our parallel software, however, in which we have implemented the three-level BDDC preconditioner as well as other approximate BDDC coarse spaces, is based on an equivalent formulation of the BDDC method as a preconditioner for the fully assembled system $K_g u = f_g$; see, e.g., [126]. In particular, the exact BDDC preconditioner formulation from [126] is applied directly to the fully assembled matrix in (3.1). For reasons of completeness, let us now briefly present this alternative formulation of the exact BDDC preconditioner.

In principle, we can reuse most of the notation as introduced in Sections 3.2.1 and 3.3.1. First, let us recall from (3.1) and (3.2) that the fully assembled system is given by

$$K_g = R^T K R$$

with the corresponding right-hand side

$$f_g = R^T f,$$

where K and f are a block matrix or block vector, respectively, and R^T is an assembly operator with $R^T : W \to V^h$. We use the same sub-partitioning of the degrees of freedom into the index sets I, Γ, Π and Δ as already introduced in Sections 3.2.1 and 3.3.1. Additionally, let us define the global assembly operators $\breve{R}^T$ and $\widetilde{R}^T$ with $\breve{R}^T : W \to \widetilde{W}$ and $\widetilde{R}^T : \widetilde{W} \to V^h$. We then introduce the partially assembled system

$$\widetilde{K} := \breve{R}^T K \breve{R}. \tag{5.1}$$

Note that we can also obtain the globally assembled finite element matrix K_g from $\widetilde{K}$ by

$$K_g = \widetilde{R}^T \widetilde{K} \widetilde{R}. \tag{5.2}$$

Ordering the interior variables first and the interface variables last, we obtain

$$\widetilde{K} = \begin{pmatrix} K_{II} & \widetilde{K}_{\Gamma I}^T \\ \widetilde{K}_{\Gamma I} & \widetilde{K}_{\Gamma\Gamma} \end{pmatrix}. \tag{5.3}$$

The matrix K_{II} is block-diagonal and applications of K_{II}^{-1} only require local solves on the interior parts of the subdomains and are thus easily parallelizable.

As already described in Sections 3.2.1 and 3.3.1, we can add different scalings, e.g., ρ-scaling [111] or deluxe-scaling [13], to the prolongation operators and thus define the scaled operator $\widetilde{R}_D : V^h \to \widetilde{W}$. We can then formulate the BDDC preconditioner for K_g by

$$M_{D,\text{BDDC-K}}^{-1} := \left(\widetilde{R}_D^T - \mathcal{H}\, P_D\right) \widetilde{K}^{-1} \left(\widetilde{R}_D - P_D^T \mathcal{H}^T\right); \tag{5.4}$$

see [126]. Here, the operator $\mathcal{H} : \widetilde{W} \to V^h$ is the discrete harmonic extension to the interior of the subdomains given by

$$\mathcal{H} := \begin{pmatrix} 0 & -\left(K_{II}\right)^{-1} \widetilde{K}_{\Gamma I}^T \\ 0 & 0 \end{pmatrix}. \tag{5.5}$$

Let us recall from the definition in (3.11) in Section 3.2.2, that the operator $P_D : \widetilde{W} \to \widetilde{W}$ is defined by

$$P_D := B_D^T B.$$

Additionally, in the context of BDDC methods, the operator P_D is sometimes also defined as

$$P_D = I - E_D := I - \widetilde{R}\widetilde{R}_D^T; \tag{5.6}$$

see [165, Chapter 6] for more details. Let us note again that in the standard definition, the BDDC preconditioner is formulated for the reduced interface problem, i.e., as

$$M_{D,\mathrm{BDDC}}^{-1} S_g := \widetilde{R}_{D,\Gamma}^T \widetilde{S}_{\mathrm{BDDC}}^{-1} \widetilde{R}_{D,\Gamma} S_g; \tag{5.7}$$

cf. also (3.18) in Section 3.3.1. Here, the prolongation operator $\widetilde{R}_{D,\Gamma}$ is formed in the same way as $\widetilde{R}_D$ but it is restricted to the interface variables on Γ, and S_g and $\widetilde{S}_{\mathrm{BDDC}}$ are the Schur complements of the matrices K_g and $\widetilde{K}$, respectively; see also Section 3.3.1.

To further examine the relation between the BDDC preconditioner formulated for the fully assembled system K_g and the original formulation for the Schur complement system S_g, we can compare the spectra of both methods, which are central for the convergence properties of the related algorithm. In particular, it can be shown that the preconditioned system $M_{D,\mathrm{BDDC\text{-}K}}^{-1} K_g$ has, except for some eigenvalues equal to one, the same spectrum as the standard BDDC preconditioner formulated using the Schur complement; see [126, Theorem 1]. Here, we provide a related but slightly more direct proof, which has already been published by the author of this thesis and her coauthors in [103].

First, we explicitly write the BDDC preconditioner $M_{D,\mathrm{BDDC\text{-}K}}^{-1}$ for the fully assembled system as

$$\begin{aligned} M_{D,\mathrm{BDDC\text{-}K}}^{-1} &:= \left(\widetilde{R}_D^T - \mathcal{H}\, P_D\right) \widetilde{K}^{-1} \left(\widetilde{R}_D - P_D^T \mathcal{H}^T\right) \\ &= \begin{pmatrix} I & K_{II}^{-1} \widetilde{K}_{\Gamma I}^T (I - \widetilde{R}_\Gamma \widetilde{R}_{D,\Gamma}^T) \\ 0 & \widetilde{R}_{D,\Gamma}^T \end{pmatrix} \widetilde{K}^{-1} \begin{pmatrix} I & 0 \\ (I - \widetilde{R}_{D,\Gamma} \widetilde{R}_\Gamma^T) \widetilde{K}_{\Gamma I} K_{II}^{-1} & \widetilde{R}_{D,\Gamma} \end{pmatrix} \\ &= \begin{pmatrix} I & K_{II}^{-1} \widetilde{K}_{\Gamma I}^T (I - E_{D,\Gamma}) \\ 0 & \widetilde{R}_{D,\Gamma}^T \end{pmatrix} \widetilde{K}^{-1} \begin{pmatrix} I & 0 \\ (I - E_{D,\Gamma}^T) \widetilde{K}_{\Gamma I} K_{II}^{-1} & \widetilde{R}_{D,\Gamma} \end{pmatrix}. \end{aligned}$$

Using the block factorization

$$\widetilde{K}^{-1} = \begin{pmatrix} I & -K_{II}^{-1} \widetilde{K}_{\Gamma I}^T \\ 0 & I \end{pmatrix} \begin{pmatrix} K_{II}^{-1} & 0 \\ 0 & \widetilde{S}_{\mathrm{BDDC}}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\widetilde{K}_{\Gamma I} K_{II}^{-1} & I \end{pmatrix},$$

we obtain, by a direct computation, the alternative representation

$$M_{D,\mathrm{BDDC\text{-}K}}^{-1} = \begin{pmatrix} K_{II}^{-1} + K_{II}^{-1} \widetilde{K}_{\Gamma I}^T E_{D,\Gamma} \widetilde{S}_{\mathrm{BDDC}}^{-1} E_{D,\Gamma}^T \widetilde{K}_{\Gamma I} K_{II}^{-1} & -K_{II}^{-1} \widetilde{K}_{\Gamma I}^T E_{D,\Gamma} \widetilde{S}_{\mathrm{BDDC}}^{-1} \widetilde{R}_{D,\Gamma} \\ -\widetilde{R}_{D,\Gamma}^T \widetilde{S}_{\mathrm{BDDC}}^{-1} E_{D,\Gamma}^T \widetilde{K}_{\Gamma I} K_{II}^{-1} & \widetilde{R}_{D,\Gamma}^T \widetilde{S}_{\mathrm{BDDC}}^{-1} \widetilde{R}_{D,\Gamma} \end{pmatrix}.$$

Computing $M_{D,\mathrm{BDDC\text{-}K}}^{-1} K_g$ finally yields

$$M_{D,\mathrm{BDDC\text{-}K}}^{-1} K_g = \begin{pmatrix} I & U \\ 0 & M_{D,\mathrm{BDDC}}^{-1} S_g \end{pmatrix}$$

where $U := K_{II}^{-1} K_{I\Gamma} - K_{II}^{-1} K_{\Gamma I}^T \widetilde{R}_{D,\Gamma}^T \widetilde{S}_{\mathrm{BDDC}}^{-1} \widetilde{R}_{D,\Gamma} S_g$, using that $E_{D,\Gamma} = \widetilde{R}_\Gamma \widetilde{R}_{D,\Gamma}^T$, and $K_{\Gamma I} = \widetilde{R}_\Gamma^T \widetilde{K}_{\Gamma I}$. Here, $M_{D,\mathrm{BDDC}}^{-1}$ is the classic BDDC preconditioner for the Schur complement; see (5.7). The result then follows from the fact, that the set of eigenvalues of a block-triangular matrix equals the union of the sets of eigenvalues of the diagonal blocks.

5.2 Approximate BDDC preconditioners

As mentioned in the introduction of this chapter, in general, different approaches exist of how to construct an approximate BDDC preconditioner. Specifically, all approximate BDDC methods considered in this thesis are based on an approximate solution of the coarse problem of BDDC, i.e., an approximation of the original coarse space is computed. To ensure a simple and fair comparison between the different methods, all approximate preconditioners are implemented using the same software framework; see also [99, 101].

In the following, we will derive a common representation of approximate BDDC preconditioners which can be generically used for all approaches considered in this thesis. By a block factorization of the partially assembled matrix $\widetilde{K}^{-1}$, we obtain

$$\widetilde{K}^{-1} = \begin{pmatrix} K_{BB}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^{T} \\ I \end{pmatrix} \widetilde{S}_{\Pi\Pi}^{-1} \begin{pmatrix} -\widetilde{K}_{\Pi B}K_{BB}^{-1} & I \end{pmatrix}, \tag{5.8}$$

where $\widetilde{S}_{\Pi\Pi}$ is the primally assembled Schur complement; see also (3.8). In particular, the matrix $\widetilde{S}_{\Pi\Pi}$ represents the BDDC coarse operator or the coarse problem, respectively.

By replacing $\widetilde{S}_{\Pi\Pi}^{-1}$ by an approximation $\widehat{S}_{\Pi\Pi}^{-1}$ in (5.8), we obtain an approximation for $\widetilde{K}^{-1}$ by

$$\widehat{K}^{-1} = \begin{pmatrix} K_{BB}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^{T} \\ I \end{pmatrix} \widehat{S}_{\Pi\Pi}^{-1} \begin{pmatrix} -\widetilde{K}_{\Pi B}K_{BB}^{-1} & I \end{pmatrix}. \tag{5.9}$$

As a next step, replacing $\widetilde{K}^{-1}$ in (5.4) by the approximated factorization of $\widehat{K}^{-1}$, we can define a generic approximation to the BDDC preconditioner, i.e.,

$$\widehat{M}_{\text{BDDC-K}}^{-1} := \left(\widetilde{R}_D^T - \mathcal{H}P_D\right) \widehat{K}^{-1} \left(\widetilde{R}_D - P_D^T\mathcal{H}^T\right). \tag{5.10}$$

Thus, we basically obtain the approximate BDDC preconditioner $\widehat{M}_{\text{BDDC-K}}^{-1}$ by replacing the solution of the coarse problem $\widetilde{S}_{\Pi\Pi}^{-1}$ in $M_{D,\text{BDDC-K}}^{-1}$ by an approximate solution. The concrete definition of the approximate BDDC preconditioner then exclusively depends on the specific method which is used for the approximation of the coarse problem related to $\widetilde{S}_{\Pi\Pi}^{-1}$.

For the remainder of this chapter, all approximate BDDC preconditioners are marked with a hat. Moreover, to avoid a proliferation of notation, we will drop the additional subindex '-K' in $\widehat{M}_{\text{BDDC-K}}^{-1}$ for the approximate BDDC preconditioners since all presented variants will be formulated as a preconditioner for the assembled system matrix K_g; see also Section 5.1.

In the following sections, we compare three different approaches to form $\widehat{S}_{\Pi\Pi}^{-1}$:

a) using exact BDDC recursively forming a three-level BDDC method, denoted by $\widehat{M}_{\text{BDDC,3L}}^{-1}$;

b) using AMG to precondition $\widetilde{S}_{\Pi\Pi}$, denoted by $\widehat{M}_{\text{BDDC,AMG}}^{-1}$;

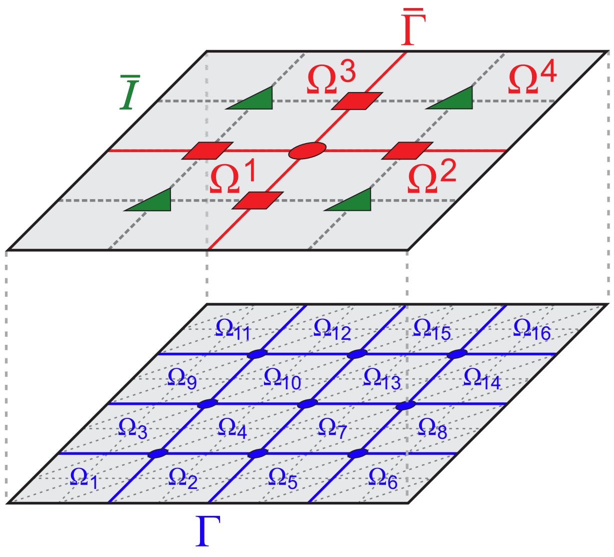

$\overline{\Gamma}$
$\overline{I}$
Ω^3
Ω^4
Ω^1
Ω^2
Ω_{11} Ω_{12} Ω_{15} Ω_{16}
Ω_9 Ω_{10} Ω_{13} Ω_{14}
Ω_3 Ω_4 Ω_7 Ω_8
Ω_1 Ω_2 Ω_5 Ω_6
Γ

Figure 5.1: Example of a three-level domain decomposition into 16 regular subdomains (bottom) and 4 regular subregions (top). We mark in blue the interface Γ between subdomains and in red the interface $\overline{\Gamma}$ between subregions. Primal nodes $\overline{\Pi}$ on the third level are visualized as red circles, while primal nodes Π on the second level are visualized as blue circles. Inner or dual nodes on the third level, i.e., $\overline{I}$ or $\overline{\Delta}$, are visualized as green triangles or red squares, respectively.

c) using an exact solution of a smaller vertex-based coarse space in combination with a Jacobi/Gauss-Seidel method, denoted by $\widehat{M}^{-1}_{\text{BDDC,VB}}$.

In Section 5.5, we will especially focus on a numerical comparison of the three-level BDDC preconditioner $\widehat{M}^{-1}_{\text{BDDC,3L}}$ and the AMG-based approach defined by $\widehat{M}^{-1}_{\text{BDDC,AMG}}$. Additionally, we will further show some parallel results for the vertex-based preconditioner $\widehat{M}^{-1}_{\text{BDDC,VB}}$ for completeness. Let us remark that $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ was denoted $\widehat{M}^{-1}_{3}$ in [101].

5.2.1 A three-level BDDC preconditioner

The main idea of the three-level BDDC method as proposed in [166–168] is to recursively apply the exact BDDC preconditioner to the Schur complement matrix $\widetilde{S}_{\Pi\Pi}$ and thus forming a multilevel approach. Related methods have also been considered in [4, 131, 160, 161]. We therefore recursively introduce a further, i.e., a third level of the domain decomposition of Ω into $\overline{N}$ nonoverlapping subregions $\Omega^1, ..., \Omega^{\overline{N}}$. Each subregion then comprises a given number of subdomains. On the subregion level, we have as many degrees of freedom as we have primal variables on the subdomain level. All primal variables Π on the subdomain level are then again partitioned into interior, primal, and dual variables, denoted by $\overline{I}, \overline{\Pi}$, and $\overline{\Delta}$, with respect to the subregions; see also Fig. 5.1

for a possible selection in two dimensions. Now, in principle, the subdomains take over the role of finite elements on the third level and the subregions the role of the subdomains. In particular, this means that the basis functions of the third level are the coarse basis functions of the second level, localized to the subregions.

Let us note that the three-level BDDC preconditioner in its original formulation in [167, 168] is a preconditioner for the Schur complement system S_g. In [167, 168], the BDDC formulation for the Schur complement system on the interface is used and applied recursively. Here, we adopt this approach and formulate a corresponding preconditioner for the fully assembled system matrix K_g. We thus use the alternative formulation of the exact BDDC preconditioner for the assembled system in Section 5.1 as a starting point and apply this approach to form the third level. For the remainder of this chapter, we mark all operators and spaces defined for the third level with bars, e.g., $\overline{I}$ represents the interior variables of the third level, while I represent the respective variables on the second level. Additionally, in Section 5.3, we derive the same condition number bound as in [167, 168] for the three-level BDDC method.

Let us now describe the application of BDDC to $\widetilde{S}_{\Pi\Pi}$ in some more details. The following presentation of the three-level BDDC method has already been published in a slighlty modified form in [103].

We first define the global space $\overline{V}^h$, which is spanned by all coarse basis functions of the second level and denote by $\overline{W}_i$, $i = 1, ..., \overline{N}$ the local spaces which are spanned by the restrictions of the coarse basis functions to the subregions Ω^i, $i = 1, ..., \overline{N}$. The product space $\overline{W}$ is now defined as $\overline{W} = \overline{W}_1 \times ... \times \overline{W}_{\overline{N}}$.

Using local Schur complements $S_{\Pi\Pi}^{(i)} = K_{\Pi\Pi}^{(i)} - K_{\Pi B}^{(i)} K_{BB}^{(i)-1} K_{\Pi B}^{(i)T}$ on the subdomains and the block matrix $S_{\Pi\Pi} = \operatorname{diag}(S_{\Pi\Pi}^{(1)}, ..., S_{\Pi\Pi}^{(N)})$, we can redefine the coarse problem as

$$\widetilde{S}_{\Pi\Pi} = \sum_{i=1}^{N} R_{\Pi}^{(i)T} S_{\Pi\Pi}^{(i)} R_{\Pi}^{(i)},$$

where $R^T = \left(R^{(1)T}, ..., R^{(N)T}\right)$ and $R^{(i)} = \operatorname{diag}\left(R_B^{(i)}, R_{\Pi}^{(i)}\right)$, $i = 1, ..., N$. Alternatively, we can perform this assembly process only on the subregions, i.e.,

$$\overline{S}_j = \sum_{i=1}^{N_j} R_{\Pi}^{(i)T} S_{\Pi\Pi}^{(i)} R_{\Pi}^{(i)}, \ \forall j = 1, ..., \overline{N}, \tag{5.11}$$

where N_j is the number of subdomains belonging to subregion Ω^j. Obviously, $\widetilde{S}_{\Pi\Pi}$ takes over the role of K_g on the third level, while $\overline{S}_j$ takes over the role of K_i. Consequently, defining a prolongation $\overline{R} : \overline{V}^h \to \overline{W}$ on the subregion level, we can also write

$$\widetilde{S}_{\Pi\Pi} = \overline{R}^T \overline{S}\, \overline{R},$$

with $\overline{S} = \operatorname{diag}(\overline{S}_1, ..., \overline{S}_{\overline{N}})$.

As a next step, let us introduce the space $\widetilde{\overline{W}} \subset \overline{W}$ of functions, which are continuous in all primal variables $\overline{\Pi}$ on the third level, and the corresponding assembly operators

$\check{\overline{R}}^T : \overline{W} \rightarrow \widetilde{\overline{W}}$ and $\widetilde{\overline{R}}^T : \widetilde{\overline{W}} \rightarrow \overline{V}^h$. Using $\check{\overline{R}}$, we can form the partially assembled system on the subregion level

$$\widetilde{\overline{S}} := \check{\overline{R}}^T \overline{S} \check{\overline{R}}. \tag{5.12}$$

Adding scalings to the prolongations as before and thus defining $\widetilde{\overline{R}}_D : \overline{V}^h \rightarrow \widetilde{\overline{W}}$, we obtain the BDDC preconditioner for the third level by

$$\overline{M}_{\text{BDDC}}^{-1} := \left(\widetilde{\overline{R}}_D^T - \overline{\mathcal{H}}\,\overline{P}_D \right) \widetilde{\overline{S}}^{-1} \left(\widetilde{\overline{R}}_D - \overline{P}_D^T \overline{\mathcal{H}}^T \right). \tag{5.13}$$

Here, the operator $\overline{\mathcal{H}} : \widetilde{\overline{W}} \rightarrow \overline{V}^h$ is the discrete harmonic extension to the interior of the subregions and writes

$$\overline{\mathcal{H}} := \left(\begin{array}{cc} 0 & -\left(\overline{S}_{\overline{II}}\right)^{-1} \widetilde{\overline{S}}_{\overline{\Gamma I}}^T \\ 0 & 0 \end{array} \right), \tag{5.14}$$

with the blocks $\overline{S}_{\overline{II}}$ and $\widetilde{\overline{S}}_{\overline{\Gamma I}}$ of the partially assembled matrix

$$\widetilde{\overline{S}} = \left(\begin{array}{cc} \overline{S}_{\overline{II}} & \widetilde{\overline{S}}_{\overline{\Gamma I}}^T \\ \widetilde{\overline{S}}_{\overline{\Gamma I}} & \widetilde{\overline{S}}_{\overline{\Gamma\Gamma}} \end{array} \right), \tag{5.15}$$

and the jump operator defined as $\overline{P}_D := I - \overline{\widetilde{R}\widetilde{R}}_D^T$.

Now, by choosing $\widehat{S}_{\Pi\Pi}^{-1} := \overline{M}_{\text{BDDC}}^{-1}$ as an approximation for $\widetilde{S}_{\Pi\Pi}^{-1}$ in (5.9), we obtain

$$\widehat{K}_{3\text{L}}^{-1} = \left(\begin{array}{cc} K_{BB}^{-1} & 0 \\ 0 & 0 \end{array} \right) + \left(\begin{array}{c} -K_{BB}^{-1} \widetilde{K}_{\Pi B}^T \\ I \end{array} \right) \overline{M}_{\text{BDDC}}^{-1} \left(\begin{array}{cc} -\widetilde{K}_{\Pi B} K_{BB}^{-1} & I \end{array} \right), \tag{5.16}$$

and can formally define the three-level BDDC preconditioner by

$$\widehat{M}_{\text{BDDC,3L}}^{-1} := \left(\widetilde{R}_D^T - \mathcal{H} P_D \right) \widehat{K}_{3\text{L}}^{-1} \left(\widetilde{R}_D - P_D^T \mathcal{H}^T \right). \tag{5.17}$$

With regard to an efficient parallel implementation, let us note that instead of inverting $\widetilde{\overline{S}}$ in (5.13) directly, we again can use a block factorization

$$\widetilde{\overline{S}}^{-1} = \left(\begin{array}{cc} \overline{S}_{\overline{BB}}^{-1} & 0 \\ 0 & 0 \end{array} \right) + \left(\begin{array}{c} -\overline{S}_{\overline{BB}}^{-1} \widetilde{\overline{S}}_{\overline{\Pi B}}^T \\ I \end{array} \right) \widetilde{\overline{T}}_{\overline{\Pi\Pi}}^{-1} \left(\begin{array}{cc} -\widetilde{\overline{S}}_{\overline{\Pi B}} \overline{S}_{\overline{BB}}^{-1} & I \end{array} \right), \tag{5.18}$$

where the primal Schur complement on the subregion level is defined by

$$\widetilde{\overline{T}}_{\overline{\Pi\Pi}} := \widetilde{\overline{S}}_{\overline{\Pi\Pi}} - \widetilde{\overline{S}}_{\overline{\Pi B}} \overline{S}_{\overline{BB}}^{-1} \widetilde{\overline{S}}_{\overline{\Pi B}}^T.$$

Thus, using a three-level BDDC preconditioner, we can reduce the problem of solving the coarse problem $\widetilde{S}_{\Pi\Pi}$ directly in (5.8) to the solution of a smaller coarse problem $\widetilde{\overline{T}}_{\overline{\Pi\Pi}}$ on the subregion level. Note that the solution of the coarse problem is usually the bottleneck in a parallel implementation. In general, the dimension of $\widetilde{\overline{T}}_{\overline{\Pi\Pi}}$ is much smaller compared to the size of $\widetilde{S}_{\Pi\Pi}$ (depending on the number of subdomains per subregion) which increases the parallel scalabilty of the method significantly.

To conclude this section let us note that, following [126, Theorem 1], the preconditioned system $\overline{M}_{\text{BDDC}}^{-1}\widetilde{S}_{\Pi\Pi}$ on the subregion level has the same eigenvalues as $\widetilde{\overline{R}}_{D,\overline{\Gamma}}^{T}\widetilde{\overline{T}}_{\overline{\Gamma\Gamma}}^{-1}\widetilde{\overline{R}}_{D,\overline{\Gamma}}\widetilde{T}_{\overline{\Gamma\Gamma}}$ except for some eigenvalues equal to one. Here, we have the Schur complement $\widetilde{T}_{\overline{\Gamma\Gamma}}$ of $\widetilde{S}_{\Pi\Pi}$ on the interface of the subregions, the primally assembled Schur complement $\widetilde{\overline{T}}_{\overline{\Gamma\Gamma}}$ of $\widetilde{\overline{S}}$ on the interface of the subregions, and the splitting $\widetilde{\overline{R}}_D = \text{diag}(I_{\overline{I}} \quad \widetilde{\overline{R}}_{D,\overline{\Gamma}})$. Therefore, we can use the condition number estimates provided in [167, 168] as in Section 5.3.

5.2.2 BDDC with an AMG-based coarse preconditioner

As a second approximate BDDC approach, we briefly describe a BDDC method which uses an AMG-based preconditioner for the solution of the coarse problem $\widetilde{S}_{\Pi\Pi}$. The resulting BDDC preconditioner has already been introduced in [101], where the authors have compared several linear and nonlinear BDDC variants using AMG-based approximations of the coarse space. Since we numerically compare the respective AMG-based preconditioner with the previously introduced three-level BDDC method in Section 5.5, we also include a brief description of the linear BDDC method with an AMG-based preconditioner for completeness.

Generally speaking, algebraic as well as geometric multigrid methods are based on defining a hierarchy of coarser grids or levels and restricting the considered linear system of PDEs to each of them. Then, by defining appropriate smoothing or relaxation schemes for each level as well as restriction and interpolation operators between the neighboring levels, a mathematical relation between the different levels is established. Finally, the considered system is solved only on the coarsest level, which is usually small and therefore computationally efficient. The solution of the coarse grid is then interpolated back to the original grid using the defined prolongation operators as well as some smoothing operators to correct the error of the interpolated solution. Such an iteration, i.e., starting at the original grid as well as terminating at the original grid with the interpolated and smoothed solution which has been computed at the coarse grid, is referred to as a *V-cycle* in the context of multigrid methods. In AMG-based methods, other than in geometric multigrid methods, the coarsening is performed without using any geometrical or mesh-related information but by exclusively considering the entries of the respective system matrix. Since a more detailed description of AMG methods is beyond the scope of this thesis, we refer to, e.g., [152, 164] for more details on AMG methods. Additionally, for a detailed description of the application of AMG for systems of PDEs, we refer to [120, Chap. 4].

For a mathematical description of the approximate BDDC method using AMG, let us denote the application of a fixed number of V-cycles of an AMG method to $\widetilde{S}_{\Pi\Pi}$ by M_{AMG}^{-1}. By choosing M_{AMG}^{-1} in (5.9) as an approximation of $\widetilde{S}_{\Pi\Pi}$, i.e., by choosing

$\widehat{S}_{\Pi\Pi}^{-1} := M_{\text{AMG}}^{-1}$, we obtain

$$\widehat{K}_{\text{AMG}}^{-1} = \begin{pmatrix} K_{BB}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^{T} \\ I \end{pmatrix} M_{\text{AMG}}^{-1} \begin{pmatrix} -\widetilde{K}_{\Pi B}K_{BB}^{-1} & I \end{pmatrix}. \tag{5.19}$$

Again, by using $\widehat{K}_{\text{AMG}}^{-1}$ as an approximation for $\widetilde{K}^{-1}$ in (5.10), we obtain the inexact reduced preconditioner $\widehat{M}_{\text{BDDC,AMG}}^{-1}$.

5.2.3 A vertex-based BDDC preconditioner

As third and last approximate BDDC method considered in this thesis, we briefly describe a vertex-based preconditioner for the coarse problem, as introduced by Dohrmann, Pierson, and Widlund [40, 41], in our framework. As already defined at the beginning of Section 5.2, we denote this approach by $\widehat{M}_{\text{BDDC,VB}}^{-1}$. Here, the preconditioner for the coarse problem can be interpreted as a standard two-level additive or multiplicative Schwarz algorithm. In particular, the direct solution of the coarse problem $\widetilde{S}_{\Pi\Pi}^{-1}$ is replaced by a preconditioner M_{VB}^{-1} based on a smaller vertex-based coarse space. This can reduce the memory and computational requirements of the method significantly.

Even though the use of a standard vertex coarse space has similar memory benefits, it was shown early in the history of FETI-DP and BDDC, that vertex nodes alone as coarse nodes do not give us competitive algorithms [51,111]; cf. also the respective explanations in Sections 3.2.2 and 3.3.2 on classic coarse spaces. As a remedy, the classic FETI-DP or BDDC coarse spaces are usually enhanced by additional (weighted) average values over certain equivalence classes, i.e., edges and/or faces. Taking this observation as a starting point, the basic idea of the coarse component of the preconditioner M_{VB}^{-1} is to approximate the averages over edges or faces using adjacent vertex values. This technique allows to delay the point, i.e., the bottleneck, when a new level has to be introduced and, in a multilevel context, may help to reduce the number of levels.

For a mathematical description of this technique, we denote the vertex-based coarse space by $\widetilde{W}_{\Psi}$ and the original coarse space by $\widetilde{W}_{\Pi}$. Then, as in [41], we define $\Psi : \widetilde{W}_{\Psi} \to \widetilde{W}_{\Pi}$ as the coarse interpolant between the coarse space based on vertices and the original coarse space based on certain equivalence classes. It is important that the coarse basis functions of $\widetilde{W}_{\Psi}$, i.e., the columns of Ψ, provide a partition of unity in the original coarse space $\widetilde{W}_{\Pi}$. This is, e.g., fulfilled for the following definition of Ψ as suggested in [41]. Let us first assume that $\widetilde{W}_{\Pi}$ consists of edge averages only. Then, each row of Ψ corresponds to a single edge constraint and has, in the case of an inner edge, two entries of $1/2$ in the two columns corresponding to the two vertices located at the endpoints of the edge. All other entries of the row are zero. In case of an edge touching the Dirichlet boundary with one endpoint, the corresponding row has a single entry of 1 in the column corresponding to the vertex located at the other end of the edge. Completely analogously, a partition of unity can also be formed for coarse spaces $\widetilde{W}_{\Pi}$ consisting of face constraints.

Again as in [41], we define $\widetilde{S}_{\Pi\Pi,r} := \Psi^T \widetilde{S}_{\Pi\Pi}\Psi$ as the reduced coarse matrix. Note that the number of rows and columns of $\widetilde{S}_{\Pi\Pi,r}$ equals the number of vertices for scalar problems. The preconditioner M_{VB}^{-1} for the coarse matrix $\widetilde{S}_{\Pi\Pi}$ is then given as

$$M_{\mathrm{VB}}^{-1} = \Psi \widetilde{S}_{\Pi\Pi,r}^{-1} \Psi^T + \mathrm{GS}(\widetilde{S}_{\Pi\Pi}), \tag{5.20}$$

where GS denotes the application of a Gauss-Seidel preconditioner. In particular, M_{VB}^{-1} can be interpreted as a Gauss-Seidel preconditioner with an additive coarse correction.

Let us note that in [41], only edge averages or only face averages are used which are each reduced to vertex-based coarse spaces as described above. In general, also the combination of vertices, edge, and face averages as coarse components can be considered and can be reduced to an exclusively vertex-based coarse space.

Eventually, we can formally define the vertex-based approximate BDDC preconditioner by choosing $\widehat{S}_{\Pi\Pi}^{-1} := M_{\mathrm{VB}}^{-1}$ as an approximation for $\widetilde{S}_{\Pi\Pi}^{-1}$ in (5.9). We then obtain the approximation $\widehat{K}_{\mathrm{VB}}^{-1}$ of $\widetilde{K}^{-1}$ as

$$\widehat{K}_{\mathrm{VB}}^{-1} = \begin{pmatrix} K_{BB}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^T \\ I \end{pmatrix} M_{\mathrm{VB}}^{-1} \begin{pmatrix} -\widetilde{K}_{\Pi B} K_{BB}^{-1} & I \end{pmatrix}, \tag{5.21}$$

and finally

$$\widehat{M}_{\mathrm{BDDC,VB}}^{-1} = \left(\widetilde{R}_D^T - \mathcal{H}\, P_D\right) \widehat{K}_{\mathrm{VB}}^{-1} \left(\widetilde{R}_D - P_D^T \mathcal{H}^T\right); \tag{5.22}$$

using the notation from (5.10); see also [41].

5.3 Condition number bounds

In this section, we will derive specific condition number bounds for all three approximate BDDC coarse spaces presented in Section 5.2. To further provide a common framework we will first prove a generic condition number bound for the approximate BDDC preconditioner $\widehat{M}_{\text{BDDC-K}}^{-1}$ as defined in (5.10) and, in a second step, explicitly examine the generic constants to obtain specific upper bounds for the three considered approximate BDDC variants. The following theory and related proofs have already been published by the author of this thesis and her coauthors in [103].

First, we need to make two assumptions, which are equivalent to Assumptions 1 and 2 in [126].

Assumption 1. *For the averaging operator $E_{D,2} := \widetilde{R}(\widetilde{R}_D^T - \mathcal{H}P_D)$ we have*

$$|E_{D,2}w|_{\widetilde{K}}^2 \le \Phi(H,h)|w|_{\widetilde{K}}^2 \quad \forall w \in \widetilde{W},$$

with $\Phi(H,h)$ being a function of the mesh size h and the subdomain diameter H.

Under Assumption 1, the condition number of the exactly preconditioned system is bounded by

$$\kappa(M_{\text{BDDC-K}}^{-1} K_g) \le \Phi(H,h); \tag{5.23}$$

see, e.g., Theorem 3 in [126].

If appropriate primal constraints, e.g., edge averages and vertex constraints, are chosen, we obtain the condition number bound with $\Phi(H,h) = C(1+\log(H/h))^2$ for our homogeneous linear elasticity test case in Section 5.5 as well as for homogeneous stationary diffusion problems; see also Section 3.3.2 for a corresponding statement for the original formulation of the exact BDDC preconditioner for the Schur complement system as well as the proof at the end of Section 5.1 for a direct relation between the two variants.

Additionally, we need a second assumption which establishes a connection between the primally assembled system matrix $\widetilde{K}$ and its approximate variant $\widehat{K}$.

Assumption 2. *There are positive constants $\tilde{c}$ and $\widetilde{C}$, which might depend on h and H, such that*

$$\tilde{c} u^T \widetilde{K} u \leq u^T \widehat{K} u \leq \widetilde{C} u^T \widetilde{K} u \quad \forall u \in \widetilde{W}.$$

Using the two assumptions above, we can now prove the following Theorem 5.1 for the preconditioned operator $\widehat{M}^{-1}_{\text{BDDC-K}} K_g$, where $\widehat{M}^{-1}_{\text{BDDC-K}}$ is the approximate BDDC preconditioner for the fully assembled system; cf. also (5.10). In the proof, we basically follow the arguments in the proof of Theorem 4 in [126], but here, we use exact discrete harmonic extension operators, i.e., an exact $E_{D,2}$. This is in contrast to Theorem 4 in [126], where inexact discrete harmonic extensions are used, which is not necessary in our case. Although large parts of the proof are identical, we recapitulate the complete line of arguments for the convenience of the reader.

Theorem 5.1. *Let Assumptions 1 and 2 hold. Then, the preconditioned system operator $\widehat{M}^{-1}_{BDDC\text{-}K} K_g$ is symmetric, positive definite with respect to the bilinear form $\langle \cdot,\cdot \rangle_{K_g}$ and we have*

$$\frac{1}{\widetilde{C}} \langle u,u \rangle_{K_g} \leq \langle \widehat{M}^{-1}_{BDDC\text{-}K} K_g u, u \rangle_{K_g} \leq \frac{\Phi(H,h)}{\tilde{c}} \langle u,u \rangle_{K_g} \; \forall u \in V^h.$$

Therefore, we obtain the condition number bound $\kappa(\widehat{M}^{-1}_{BDDC\text{-}K} K_g) \leq \frac{\widetilde{C}}{\tilde{c}} \Phi(H,h)$ for the approximate BDDC preconditioner $\widehat{M}^{-1}_{BDDC\text{-}K}$ as defined in (5.10).

Proof. Let $u \in V^h$ be given. We define

$$w = \widehat{K}^{-1} (\widetilde{R}_D - P_D^T \mathcal{H}^T) K_g u \in \widetilde{W} \tag{5.24}$$

and thus also have

$$\widehat{K} w = (\widetilde{R}_D - P_D^T \mathcal{H}^T) K_g u.$$

Using $\widetilde{R}^T \widetilde{R}_D = I$, yields $\widetilde{R}^T P_D^T = \widetilde{R}^T (I - \widetilde{R}_D \widetilde{R}^T) = 0$ and thus $\text{range}(P_D^T) \subset \text{null}(\widetilde{R}^T)$. Hence, we obtain

$$\langle u,u \rangle_{K_g} = u^T \widetilde{R}^T (\widetilde{R}_D - P_D^T \mathcal{H}^T) K_g u = u^T \widetilde{R}^T \widehat{K} w = \langle w, \widetilde{R} u \rangle_{\widehat{K}}. \tag{5.25}$$

Using the Cauchy-Schwarz inequality and Assumption 2, we can further estimate

$$\begin{aligned}\langle w, \widetilde{R}u\rangle_{\widehat{K}} \leq \langle w, w\rangle_{\widehat{K}}^{1/2}\langle \widetilde{R}u, \widetilde{R}u\rangle_{\widehat{K}}^{1/2} &\overset{Asm.\ 2}{\leq} \sqrt{\widetilde{C}}\langle w, w\rangle_{\widehat{K}}^{1/2}\langle \widetilde{R}u, \widetilde{R}u\rangle_{\widetilde{K}}^{1/2} \\ &\overset{(5.2)}{=} \sqrt{\widetilde{C}}\langle w, w\rangle_{\widehat{K}}^{1/2}\langle u, u\rangle_{K_g}^{1/2}. \end{aligned} \tag{5.26}$$

Combining equations (5.25) and (5.26), we have $\langle u, u\rangle_{K_g} \leq \widetilde{C}\langle w, w\rangle_{\widehat{K}}$. Using (5.24) and (5.10), we can prove the lower bound:

$$\begin{aligned}\frac{1}{\widetilde{C}}\langle u, u\rangle_{K_g} &\leq \langle w, w\rangle_{\widehat{K}} \\ &\overset{(5.24)}{=} u^T K_g(\widetilde{R}_D^T - \mathcal{H}P_D)\widehat{K}^{-1}\widehat{K}\widehat{K}^{-1}(\widetilde{R}_D - P_D^T\mathcal{H}^T)K_g)u \\ &= \langle u, (\widetilde{R}_D^T - \mathcal{H}P_D)\widehat{K}^{-1}(\widetilde{R}_D - P_D^T\mathcal{H}^T)K_g u\rangle_{K_g} \\ &\overset{(5.10)}{=} \langle u, \widehat{M}_{\text{BDDC-K}}^{-1}K_g u\rangle_{K_g}. \end{aligned} \tag{5.27}$$

Let us now prove the upper bound using Assumption 1, (5.24), and (5.10):

$$\begin{aligned}\langle \widehat{M}_{\text{BDDC-K}}^{-1}K_g u, \widehat{M}_{\text{BDDC-K}}^{-1}K_g u\rangle_{K_g} &= \langle(\widetilde{R}_D^T - \mathcal{H}P_D)w, (\widetilde{R}_D^T - \mathcal{H}P_D)w\rangle_{K_g} \\ &= \langle \widetilde{R}(\widetilde{R}_D^T - \mathcal{H}P_D)w, \widetilde{R}(\widetilde{R}_D^T - \mathcal{H}P_D)w\rangle_{\widetilde{K}} \\ &= \langle E_{D,2}w, E_{D,2}w\rangle_{\widetilde{K}} = |E_{D,2}w|_{\widetilde{K}}^2 \\ &\overset{Asm.1}{\leq} \Phi(H,h)|w|_{\widetilde{K}}^2. \end{aligned} \tag{5.28}$$

Together with Assumption 2, we obtain

$$\begin{aligned}\langle \widehat{M}_{\text{BDDC-K}}^{-1}K_g u, \widehat{M}_{\text{BDDC-K}}^{-1}K_g u\rangle_{K_g} &\overset{(5.28)}{\leq} \Phi(H,h)|w|_{\widetilde{K}}^2 \\ &\overset{Asm.2}{\leq} \frac{1}{\tilde{c}}\Phi(H,h)|w|_{\widehat{K}}^2 \\ &\overset{(5.27)}{=} \frac{1}{\tilde{c}}\Phi(H,h)\langle u, \widehat{M}_{\text{BDDC-K}}^{-1}K_g u\rangle_{K_g}. \end{aligned} \tag{5.29}$$

Using a Cauchy-Schwarz inequality in combination with (5.29), we finally obtain

$$\langle u, \widehat{M}_{\text{BDDC-K}}^{-1}K_g u\rangle_{K_g} \leq \frac{\Phi(H,h)}{\tilde{c}}\langle u, u\rangle_{K_g}.$$

□

Let us recall from Section 5.2 that, for the preconditioners considered here, we replace the inverse operator of the Schur complement in the primal variables $\widetilde{S}_{\Pi\Pi}^{-1}$ by an approximation $\widehat{S}_{\Pi\Pi}^{-1}$. Therefore, we have to show that Assumption 2 used in the proof of Theorem 5.1 is still relevant and holds under certain assumptions.

Assumption 3. *There are positive constants $\hat{c}$ and $\widehat{C}$, which might depend on h and H, such that*

$$\hat{c}\tilde{u}_\Pi^T \widetilde{S}_{\Pi\Pi}\tilde{u}_\Pi \leq \tilde{u}_\Pi^T \widehat{S}_{\Pi\Pi}\tilde{u}_\Pi \leq \widehat{C}\tilde{u}_\Pi^T \widetilde{S}_{\Pi\Pi}\tilde{u}_\Pi \quad \forall \tilde{u}_\Pi \in \widetilde{W}_\Pi.$$

Using Assumption 3, we can now prove the following lemma.

Lemma 5.2. *Let Assumption 3 hold and $\widehat{K}^{-1}$ be defined as in (5.9). Then, Assumption 2 holds with $\tilde{c} := \min(\hat{c}, 1)$ and $\widetilde{C} := \max(\widehat{C}, 1)$.*

Proof. We first split $\widehat{K}^{-1} = A_1 + A_2$ into its two additive parts

$$A_1 := \begin{pmatrix} K_{BB}^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

and

$$A_2 := \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^T \\ I \end{pmatrix} \widehat{S}_{\Pi\Pi}^{-1} \begin{pmatrix} -\widetilde{K}_{\Pi B}K_{BB}^{-1} & I \end{pmatrix};$$

cf. also (5.9). Computing the product $A_1\widetilde{K}$ then yields

$$A_1\widetilde{K} = \begin{pmatrix} K_{BB}^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} K_{BB} & \widetilde{K}_{\Pi B}^T \\ \widetilde{K}_{\Pi B} & \widetilde{K}_{\Pi\Pi} \end{pmatrix} = \begin{pmatrix} I & K_{BB}^{-1}\widetilde{K}_{\Pi B}^T \\ 0 & 0 \end{pmatrix}. \tag{5.30}$$

Simultaneously, by a direct computation, we obtain

$$\begin{aligned} A_2\widetilde{K} &= \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^T \\ I \end{pmatrix} \widehat{S}_{\Pi\Pi}^{-1} \begin{pmatrix} -\widetilde{K}_{\Pi B}K_{BB}^{-1} & I \end{pmatrix} \begin{pmatrix} K_{BB} & \widetilde{K}_{\Pi B}^T \\ \widetilde{K}_{\Pi B} & \widetilde{K}_{\Pi\Pi} \end{pmatrix} \\ &= \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^T \\ I \end{pmatrix} \widehat{S}_{\Pi\Pi}^{-1} \begin{pmatrix} 0 & \widetilde{S}_{\Pi\Pi} \end{pmatrix} \\ &= \begin{pmatrix} -K_{BB}^{-1}\widetilde{K}_{\Pi B}^T \\ I \end{pmatrix} \begin{pmatrix} 0 & \widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi} \end{pmatrix} \\ &= \begin{pmatrix} 0 & -K_{BB}^{-1}\widetilde{K}_{\Pi B}^T\widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi} \\ 0 & \widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi} \end{pmatrix}. \end{aligned} \tag{5.31}$$

Adding (5.30) and (5.31) yields the final result

$$\widehat{K}^{-1}\widetilde{K} = \begin{pmatrix} I & G \\ 0 & \widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi} \end{pmatrix}$$

with $G := K_{BB}^{-1}\widetilde{K}_{\Pi B}^T(I - \widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi})$.

Therefore, except for additional eigenvalues equal to one, $\widehat{K}^{-1}\widetilde{K}$ and $\widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi}$ have the same spectrum, and we have $\lambda_{\min}(\widehat{K}^{-1}\widetilde{K}) = \min\left(\lambda_{\min}(\widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi}), 1\right)$ and $\lambda_{\max}(\widehat{K}^{-1}\widetilde{K}) = \max\left(\lambda_{\max}(\widehat{S}_{\Pi\Pi}^{-1}\widetilde{S}_{\Pi\Pi}), 1\right)$. Consequently, Assumption 2 holds with $\tilde{c} := \min(\hat{c}, 1)$ and $\widetilde{C} := \max(\widehat{C}, 1)$. □

In the following subsections, we will explicitly specify the constants $\widetilde{c}$ and $\widetilde{C}$ in Theorem 5.1 as well as $\hat{c}$ and $\widehat{C}$ in Lemma 5.2, respectively, for the three considered approximate BDDC methods as introduced in Section 5.2.

5.3.1 Three-level BDDC

For the three-level BDDC preconditioner $\widehat{M}_{\text{BDDC,3L}}^{-1}$ (see Section 5.2.1) we obtain, with Lemma 4.6 in [168] in two spatial dimensions and Lemma 4.7 in [167] in three spatial dimensions,

$$\hat{c} = \frac{1}{C_{3\text{L}} \left(1 + \log(\frac{\hat{H}}{H})\right)^2}$$

and

$$\widehat{C} = 1$$

for the constants $\hat{c}$ and $\widehat{C}$ in Assumption 3. Here, $\hat{H}$ is the maximum diameter of a subregion and of course, depending on the considered problem and the dimension of the domain, sufficient primal constraints on the second level have to be chosen; see [167, 168] and the discussion at the end of this chapter. Let us note that the results in [167, 168] are only proven for scalar diffusion problems. To the best of our knowledge, an extension to linear elasticity problems has not been published so far and might still be an open problem. Using Lemma 5.2 and Theorem 5.1, we obtain, for scalar elliptic problems, the following condition number bound for the three-level BDDC preconditioner $\widehat{M}_{\text{BDDC,3L}}^{-1}$:

$$\kappa(\widehat{M}_{\text{BDDC,3L}}^{-1} K_g) \leq \frac{\widetilde{C}}{\widetilde{c}} \Phi(H,h) = C_{3\text{L}} \left(1 + \log\left(\frac{\hat{H}}{H}\right)\right)^2 \Phi(H,h); \tag{5.32}$$

see also [167, 168].

5.3.2 Approximate BDDC using AMG

For the preconditioner $\widehat{M}_{\text{BDDC,AMG}}^{-1}$ based on AMG (see Section 5.2.2), we find that the constants $\widehat{C}$ and $\hat{c}$ in Assumption 3 depend on the properties of the AMG V-cycle used. Therefore, without defining further details on the chosen AMG V-cycle, we obtain the general condition number bound for the approximate BDDC preconditioner $\widehat{M}_{\text{BDDC,AMG}}^{-1}$ based on AMG:

$$\kappa(\widehat{M}_{\text{BDDC,AMG}}^{-1} K_g) \leq \frac{\widetilde{C}}{\widetilde{c}} \Phi(H,h) = \frac{\max(\widehat{C},1)}{\min(\hat{c},1)} \Phi(H,h). \tag{5.33}$$

Our aim is, of course, to obtain good constants $\tilde{c}, \widetilde{C}$ and $\hat{c}, \widehat{C}$, respectively, to ensure a small condition number bound and thus a fast convergence of the approximate BDDC method.

In general, it is well-known that, especially for sophisticated linear elasticity problems, problem-related and customized AMG variants can improve the performance significantly. Classic AMG methods are designed for scalar elliptic PDEs and assume that the null space of the discretized operator only consists of constant vectors. However, as already explained in Section 3.2.1, this assumption does not hold for linear elasticity problems where the null space contains the rigid body modes, i.e., also linear rotations. Thus, several approaches have been considered to handle linear elasticity problems with AMG. One of these approaches, which we also use in our parallel implementation, is the global matrix approach [6]. The main idea of this method is to interpolate near null space vectors exactly. This can help to improve the scalability and performance of AMG methods for elasticity problems significantly; see also [6]. More details on the global matrix approach to obtain good constants in (5.33) will be given in Section 5.3.4.

5.3.3 Vertex-based BDDC

For the vertex-based BDDC preconditioner $\widehat{M}^{-1}_{\text{BDDC,VB}}$ (see Section 5.2.3) we obtain, with Theorem 3 in [41] for edge-based or face-based coarse spaces and quasi-monotone face-connected paths,

$$\widehat{c} \geq \frac{1}{C_1}, \;\max(\widehat{C},1) \leq C_C$$

and

$$\Phi(H,h) = C\left(1+\log\left(\frac{H}{h}\right)\right)^2;$$

see [41, Theorem 3]. Here, C_C is obtained by a coloring argument and therefore usually fulfills $C_C \geq 1$. The constant C_1 depends on geometric constants, e.g., the maximum number of subdomains connected by an edge (see [41, Lemma 2]), the maximum number of neighbors of a subdomain (see [41, Eq. (4.3)]), or typical subdomain sizes (see [41, Assumption 3]). Additionally, C_1 depends on a tolerance for the lowest coefficient along an acceptable path; see [41, Assumption 1 and 2]; cf. also [116]. The results in [41] are proven for scalar diffusion and linear elasticity problems. Altogether, with another constant C_{VB}, we obtain

$$\frac{\max(\widehat{C},1)}{\min(\hat{c},1)} \leq C_{VB};$$

see also [41, Theorem 1 and 3] where $\widehat{c} = \beta_1$ and $\widehat{C} = \beta_2$ for the constants β_1 and β_2 used in [41]. Typically, we have $C_1 \geq 1$, and we can then define $C_{VB} = C_1 \cdot C_C$. Using Theorem 5.1, we thus obtain the following condition number bound for the vertex-based preconditioner $\widehat{M}^{-1}_{\text{BDDC,VB}}$:

$$\kappa(\widehat{M}^{-1}_{\text{BDDC,VB}} K_g) \leq \frac{\widetilde{C}}{\tilde{c}}\Phi(H,h) \leq C_{VB}\Phi(H,h); \tag{5.34}$$

see also [41, Theorem 3].

5.3.4 The global matrix approach

As already mentioned in Section 5.3.2, good constants $\tilde{c}, \widetilde{C}$ in Assumption 2 or, respectively, $\hat{c}, \widehat{C}$ in Assumption 3, are important for a small condition number bound and therefore a fast convergence of the approximate BDDC method. It is well known that for a good scalability of multigrid methods the preconditioner should preserve the null space or near null space vectors of the operator of the considered PDE. Therefore, the AMG method should preserve the null space of the operator on all levels and this, in turn, means that the null space vectors have to be in the range of the AMG interpolation. In general, classic AMG methods guarantee this property only for constant vectors, i.e., for the null space of scalar elliptic differential equations. Thus, different modified approaches of classic AMG have been developed to handle linear elasticity problems. One of these methods is the global matrix (GM) approach, which was originally introduced in [6]. The GM approach allows the user to specify certain near null space vectors, which are interpolated exactly from the coarsest to the finest level; more details on the method and its scalability for linear elasticity problems can be found in, e.g., [5, 6]. Since we are interested in the solution of linear elasticity problems in Section 5.5, we implement the GM approach and choose the rotational rigid body modes for the exact interpolation. All translations of the computational domain are already interpolated exactly in classic AMG approaches for systems of PDEs since they use classic interpolation applied component-by-component. In $\widehat{M}^{-1}_{\mathrm{BDDC,AMG}}$, the AMG method is applied to the primally assembled matrix $\widetilde{S}_{\Pi\Pi}$ and thus we need to include the three rotations in the space $\widetilde{W}_{\Pi}$, which is the restriction of $\widetilde{W}$ to the primal constraints. To achieve this, we first assemble the rotations of a subdomain Ω_i locally, extract the primal components, and finally insert them into three global vectors in $\widetilde{W}_{\Pi}$. In our implementation, we always use BoomerAMG from the HYPRE package [80], where a highly scalable implementation of the GM2 approach is implemented; see [5] for more details on two different variants how to implement the GM approach, which are named GM1 and GM2. Let us remark that BoomerAMG provides both variants of interpolation, and GM2 is recommended for use instead of GM1. In [5], the GM2 variant also showed a better scalability than GM1. In Section 5.5, we will compare the use of the GM2 approach with a hybrid AMG approach for systems of PDEs. By hybrid AMG approaches we refer to methods, where the coarsening is based on the physical nodes (nodal coarsening) but the interpolation is based on the degrees of freedom. In general, a nodal coarsening approach is beneficial for the solution of systems of PDEs, and all degrees of freedom belonging to the same physical node are either all coarse or fine on each level. The latter condition is also mandatory for the GM2 approach. Therefore, GM2 is based on the same nodal coarsening and can also be considered as a hybrid approach.

5.4 Parallel implementation details of approximate BDDC coarse spaces

As already mentioned, the three approximate BDDC variants which we have introduced in Section 5.2 are implemented in the same parallel software framework and using the same building blocks. Our parallel software is implemented in C/C++ using PETSc version 3.9.2 [10] (for the presented results) and MPI [65,158]. All matrices are implemented as completely local to the computational cores. For the assembly and prolongation operators between the different solution spaces, we use PETSc *VecScatter* and *VecGather* operations. A more detailed description of the parallel data structures of our implementation of the BDDC preconditioner can be found in [101], where different linear and nonlinear BDDC methods are applied to hyperelasticity and elasto-plasticity problems.

Let us describe some more details on the implementation of the different preconditioners for the coarse problem since this is the main difference between the various approximate BDDC coarse spaces. In general, the coarse problem $\widetilde{S}_{\Pi\Pi}$ is always assembled on a subset of the available cores. The number of cores can, in principle, be chosen arbitrarily and should depend on the size of the coarse problem to obtain a good performance. The three-level BDDC preconditioner as well as BoomerAMG for the preconditioning based on AMG can be applied to $\widetilde{S}_{\Pi\Pi}$ in parallel. For the exact BDDC preconditioner, which we always include in our computations to provide a common baseline, a sequential copy of $\widetilde{S}_{\Pi\Pi}$ is sent to each computational core where a sparse direct solver is applied. Thus, the coarse problem is solved redundantly on all cores. Alternatively, one could just create a single sequential copy on a single core without significantly changing the expected time to solution.

For the construction of the vertex-based preconditioner $\widehat{M}^{-1}_{\text{BDDC,VB}}$, we always build Ψ and $\widetilde{S}_{\Pi\Pi}$ as parallel matrices on the same subset of cores and compute a parallel Galerkin product to compute $\widetilde{S}_{\Pi\Pi,r} := \Psi^T \widetilde{S}_{\Pi\Pi} \Psi$. Afterwards, we create sequential copies of $\widetilde{S}_{\Pi\Pi}$ and $\widetilde{S}_{\Pi\Pi,r} := \Psi^T \widetilde{S}_{\Pi\Pi} \Psi$ in order to perform a redundant sparse factorization of $\widetilde{S}_{\Pi\Pi,r}$ and a redundant, i.e., a sequential Gauss-Seidel iteration for $\widetilde{S}_{\Pi\Pi}$. When using a sequential Gauss-Seidel implementation to construct $\widehat{M}^{-1}_{\text{BDDC,VB}}$, for simplicity, we also create a sequential copy of Ψ. In principle, this could also be avoided by slight modifications of our implementation.

For the second implemented variant of $\widehat{M}^{-1}_{\text{BDDC,VB}}$ using PETSc's parallel SOR/Gauss-Seidel implementation, no sequential copies of Ψ are created. Let us note that the parallel SOR/Gauss-Seidel implementation in PETSc is in fact a block Jacobi preconditioner in between the local blocks associated with the different MPI ranks and an SOR/Gauss-Seidel preconditioner on the local blocks themselves. Consequently, its use can obviously decrease the convergence rate of the method. As an advantage, however, we only have to build a sequential copy of $\widetilde{S}_{\Pi\Pi,r}$, which is much smaller compared to $\widetilde{S}_{\Pi\Pi}$. The matrices $\widetilde{S}_{\Pi\Pi}$ and Ψ are then kept in a distributed fashion as described above. Let us finally remark that we can apply the Gauss-Seidel preconditioner additively as described in (5.20) as well as multiplicatively, which is more robust. We will consider both variants in Section 5.5.2.

5.5 Numerical results

In this section, we provide comparative results for the different approximate BDDC coarse spaces as presented in Section 5.2. The key focus of this section is on the numerical investigation of the three-level BDDC preconditioner $\widehat{M}^{-1}_{\text{BDDC,3L}}$. Let us note that $\widehat{M}^{-1}_{\text{BDDC,3L}}$ can also be extended to a multilevel preconditioner by again applying the

BDDC method recursively on the coarse problem of the subregion level; see, e.g., [4,131]. Since the AMG method applied in $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ naturally consists of several levels, both named approaches have a similar parallel potential. Thus, we show comparative numerical results for $\widehat{M}^{-1}_{\text{BDDC,3L}}$ and $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ in Section 5.5.1. We will observe that due to the large coarsening ratio from the second to the coarsest level, both methods have a large parallel potential.

For the vertex-based preconditioner $\widehat{M}^{-1}_{\text{BDDC,VB}}$, the parallel scalability is limited by construction, since the constructed vertex-based coarse space is always solved by a sparse direct solver in our implementation; cf. also Section 5.4. Therefore, we analyze and compare $\widehat{M}^{-1}_{\text{BDDC,VB}}$ separately in Section 5.5.2. Additionally, to provide a common theoretical baseline for all considered approximate BDDC variants, we always include the exact BDDC preconditioner M^{-1}_{BDDC} in all figures in Sections 5.5.1 and 5.5.2.

As a common test problem for the following numerical experiments, we use homogeneous linear elasticity problems, i.e., with constant coefficients; see also Section 2.2. Since the focus of this chapter lays on the parallel efficiency, i.e., the scalability of the presented approaches, here, we will not consider any heterogeneous coefficient distributions. For heterogeneous examples or other model problems than linear elasticity, we refer to [101] for $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ and to [41] for $\widehat{M}^{-1}_{\text{BDDC,VB}}$.

All computations in this chapter are performed on the supercomputers magnitUDE (University of Duisburg-Essen) or JUWELS (FZ Juelich). The results presented in the following sections have already been published in [103].

5.5.1 Comparison of three-level BDDC and BDDC with AMG coarse preconditioner

Let us now compare the three-level BDDC preconditioner $\widehat{M}^{-1}_{\text{BDDC,3L}}$ and the AMG-based approximate BDDC preconditioner $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ which are both based on a multilevel structure. As the test problem, we consider a linear elastic cube decomposed into 512 subdomains with Young's modulus $E = 210\,GPa$ and two different Poisson ratios ν. For the definition of a coarse space, we enforce continuity of the values at all subdomain vertices and in all edge averages.

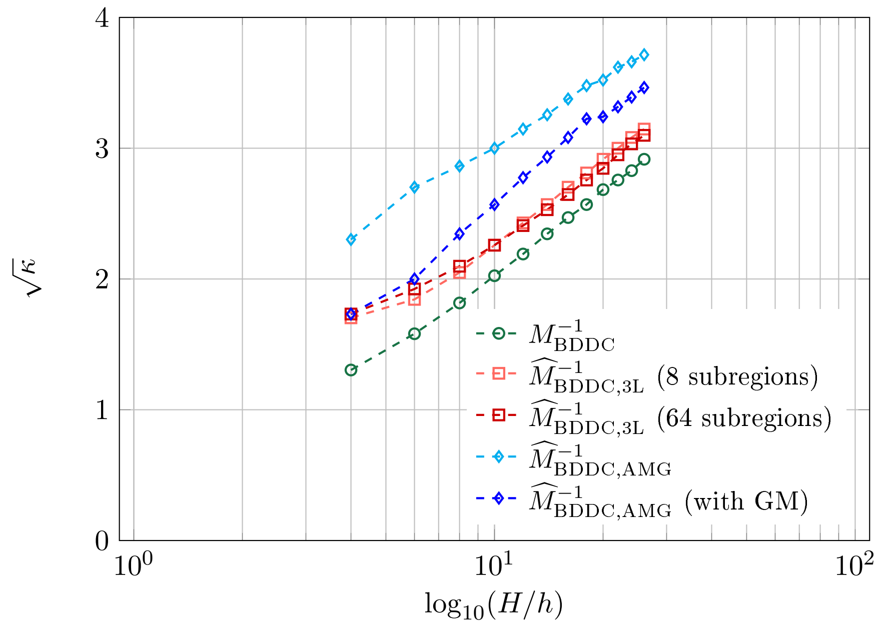
First, we verify the quadratic dependence of the condition number of both preconditioners on the logarithm of H/h, which can be seen as a measure of the subdomain size, in Fig. 5.2. From Fig. 5.2 (top) we can observe that with a Poisson ratio of $\nu = 0.3$, all methods show the expected logarithmic dependence of the condition number on the subdomain size; cf. also Section 5.3. For all tested values of H/h the preconditioner $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ has slightly higher condition numbers than the three-level BDDC methods and the exact BDDC preconditioner. In particular, we can see that it is useful to include the GM approach in $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ since this leads to smaller condition numbers. For the three-level BDDC preconditioner $\widehat{M}^{-1}_{\text{BDDC,3L}}$, both tested setups, i.e., 8 or 64 subdomains per subregion, have nearly the same condition number. Additionally, as expected, the condition numbers are slightly higher than for the exact, i.e., the two-level BDDC preconditioner M^{-1}_{BDDC}; cf. also Section 5.3.1.

For the larger Poisson ratio of $\nu = 0.49$ in Fig. 5.2 (bottom), the BDDC preconditioner $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ based on AMG has significantly higher condition numbers than the competing methods, especially for small subdomain sizes. For larger subdomain sizes and using the GM approach, $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ is again competitive with three-level BDDC. In particular, we observe that $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ for $\nu = 0.49$ shows the logarithmic dependence of the condition number only for H/h larger than 16. Let us remark that for $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$, we always use a highly scalable AMG setup, i.e., aggressive Hybrid Maximal Independent Set (HMIS) coarsening [35], $ext + i$ long range interpolation [34, 172], nodal coarsening, a threshold of 0.3 for the detection of strong couplings, and a maximum of three entries per row in the AMG interpolation matrices. Less aggressive strategies might show lower condition numbers, but we explicitly choose the parameters to obtain good parallel scalability; see [5].

Second, we perform a weak scaling study for up to 4 096 cores in Fig. 5.3 for the same setup as before with a Poisson ratio of $\nu = 0.3$ but fixed $H/h = 24$. By considering the number of CG iterations until convergence in Fig. 5.3 (top), we observe that the GM approach is necessary in $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ to obtain results of similar quality as for $\widehat{M}_{\mathrm{BDDC,3L}}^{-1}$. The same can be observed considering the time to solution; see Fig. 5.3 (bottom). Here, the reported time to solution is always the complete runtime measured for the respective program to finish. In particular, this includes the time for the assembly of the linear system, the setup of the preconditioner, and the iterative solution of the preconditioned system. Additionally, we provide more detailed timings in Fig. 5.4, where the assembly of the stiffness matrix, the BDDC setup, and the time for the iterative solution using CG are shown separately for the largest experiment from Fig. 5.3. Finally, we note from Fig. 5.3 that - as expected - the exact BDDC preconditioner does not scale due to the sequential coarse solve of the relatively large coarse problem on the second level.

5.5.2 Vertex-based BDDC

For completeness, we have also implemented an "economic" variant of the edge-based coarse space as introduced in [41, section 6]. This means that, in our implementation, e.g., three translational degrees of freedom on edges are reduced to three translational degrees of freedom of an adjacent vertex. In Fig. 5.5, we perform a weak scaling test for up to 5 832 cores for the vertex-based preconditioner $\widehat{M}_{\mathrm{BDDC,VB}}^{-1}$ for a similar model problem as in Section 5.5.1, i.e., a homogeneous linear elasticity problem with a Poisson ratio of $\nu = 0.3$ and a Young's modulus of $E = 210\,GPa$. In Fig. 5.5 (top) and Fig. 5.5 (bottom), respectively, we also provide the CG iterations and the time to solution for exact BDDC and $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ using GM for a direct comparison with $\widehat{M}_{\mathrm{BDDC,VB}}^{-1}$. Additionally, we present more detailed timings in Fig. 5.6, where the time for the assembly of the stiffness matrix, the BDDC setup, and the time for the iterative solution of the preconditioned system are shown separately for the largest experiment from Fig. 5.5. As we can observe from Fig. 5.5 and Fig. 5.6, for $\widehat{M}_{\mathrm{BDDC,VB}}^{-1}$, a multiplicative combination of Gauss-Seidel applied to $\widetilde{S}_{\Pi\Pi}$ and the direct solve of the vertex-based coarse problem is a better choice than an additive variant. The parallel Gauss-Seidel method, which - as implemented



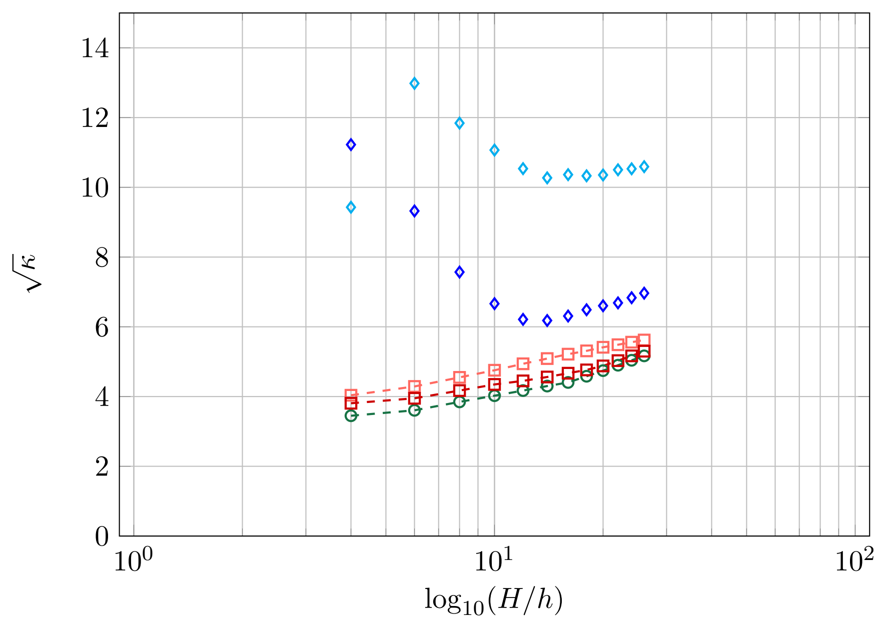
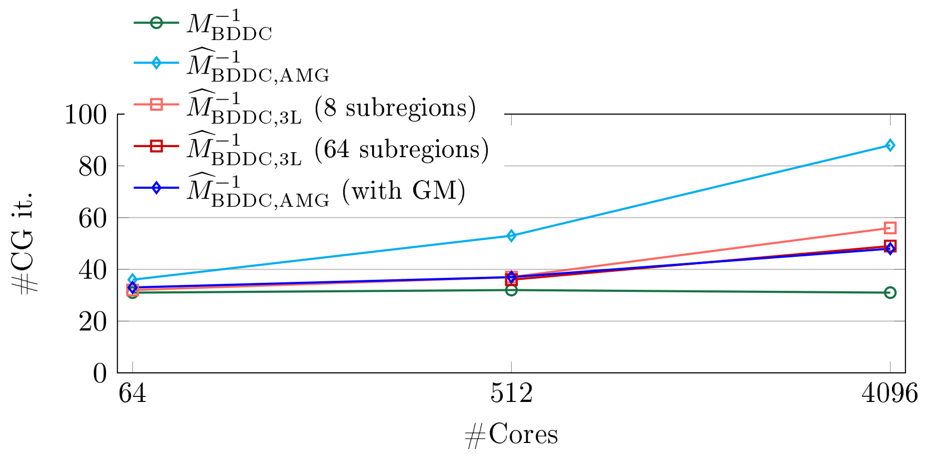


Figure 5.2: Condition number estimates of M_{BDDC}^{-1}, $\widehat{M}_{\mathrm{BDDC,3L}}^{-1}$ with 8 or 64 subregions, and $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ with and without GM for a homogeneous linear elastic cube decomposed into 512 subdomains with $H/h = 4, 6, ..., 26$. **Top:** $E = 210.0$ and $\nu = 0.3$. **Bottom:** $E = 210.0$ and $\nu = 0.49$. Computed on the magnitUDE supercomputer. The shown results have already been published in [103].



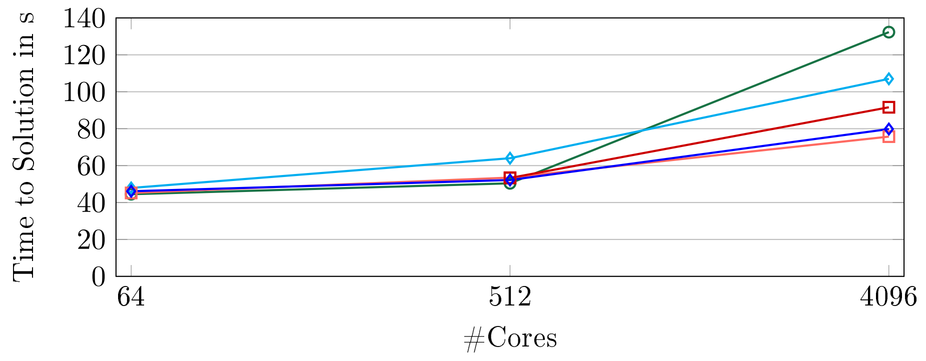


Figure 5.3: Weak scaling study for M_{BDDC}^{-1}, $\widehat{M}_{\mathrm{BDDC,3L}}^{-1}$ with 8 or 64 subregions, and $\widehat{M}_{\mathrm{BDDC,AMG}}^{-1}$ with and without GM. Using vertex and edge constraints. Homogeneous linear elastic cube decomposed into 64, 512, and 4 096 subdomains with $H/h = 24$. **Top:** Number of CG iterations. **Bottom:** Total time to solution including assembly of stiffness matrices, setup of the preconditioner and solution phase. See also Fig. 5.4 for detailed setup and solve times for the largest experiment. Computed on JUWELS. The shown results have already been published in [103].

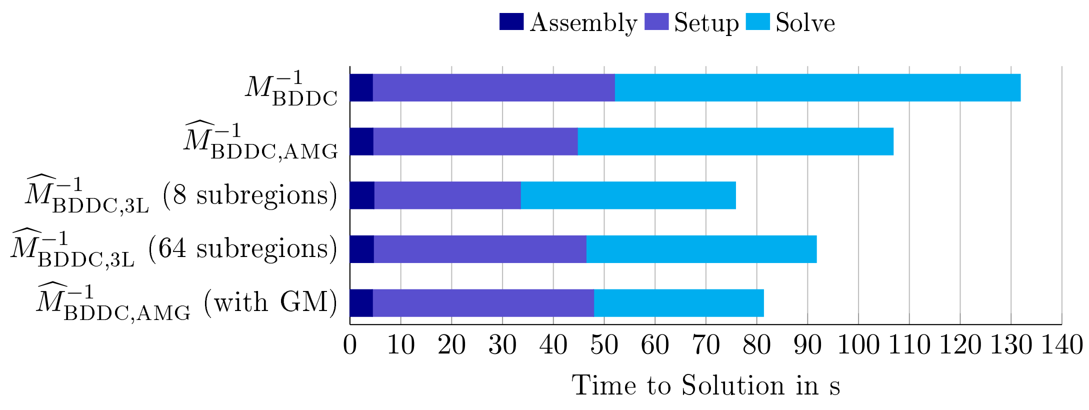


Figure 5.4: Comparison of BDDC setup and solve times of M^{-1}_{BDDC}, $\widehat{M}^{-1}_{\mathrm{BDDC,3L}}$ with 8 or 64 subregions, and $\widehat{M}^{-1}_{\mathrm{BDDC,AMG}}$ with and without GM. Using vertex and edge constraints. Homogeneous linear elastic cube decomposed into 4 096 subdomains with $H/h = 24$. The corresponding weak scaling experiment can be found in Fig. 5.3. Computed on JUWELS. The shown results have already been published in [103].

in PETSc - is in fact a block Jacobi with SOR/Gauss-Seidel blocks, always results in more CG iterations but faster runtimes. With respect to parallel scalability, the best variant of $\widehat{M}^{-1}_{\mathrm{BDDC,VB}}$ is competitive with $\widehat{M}^{-1}_{\mathrm{BDDC,AMG}}$, at least up to the moderate core count of 5 832. For an increasing number of cores, we expect $\widehat{M}^{-1}_{\mathrm{BDDC,\,AMG}}$ to outperform $\widehat{M}^{-1}_{\mathrm{BDDC,VB}}$ due to its inherent multilevel structure. Here, a three-level extension of $\widehat{M}^{-1}_{\mathrm{BDDC,VB}}$ would be needed. Alternatively, $\widehat{M}^{-1}_{\mathrm{BDDC,VB}}$ could be used to precondition the coarsest level of a three-level BDDC method. These ideas will be further discussed in the conclusion of this chapter.

5.5.3 Summarizing remarks

In this chapter, we have presented three different approaches of approximate BDDC coarse spaces and compared them with respect to theory and parallel scalability in a common framework. From the numerical experiments presented above we can observe that the three-level BDDC preconditioner $\widehat{M}^{-1}_{\mathrm{BDDC,3L}}$ and the approximate preconditioner $\widehat{M}^{-1}_{\mathrm{BDDC,AMG}}$ based on AMG show a very similar behavior and parallel scalability, given that an appropriate AMG approach is available, e.g., the GM approach in the case of linear elasticity problems. Thus, with regard to the aim to increase the parallel scalability of the iterative solver, both methods can be recommended as an alternative of the classic, i.e., two-level BDDC method. However, as expected, both approaches also result in higher condition number estimates and thus in a higher number of necessary CG iterations as the two-level BDDC method.

Additionally, up to a moderate number of compute cores, also the vertex-based BDDC

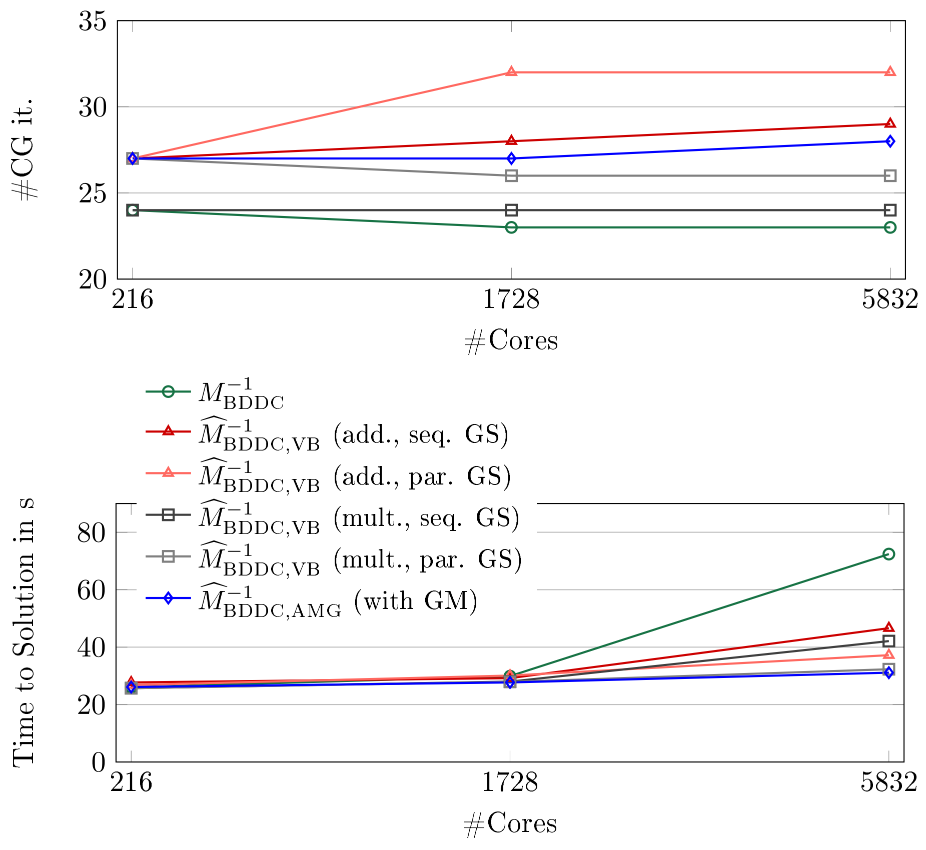


Figure 5.5: Weak scaling study for M^{-1}_{BDDC}, $\widehat{M}^{-1}_{\text{BDDC,VB}}$ using additive/multiplicative sequential/parallel Gauss-Seidel, and $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ with GM. Using only edge constraints. Homogeneous linear elastic cube with $H/h = 22$. **Top:** Number of CG iterations. **Bottom:** Total time to solution including assembly of stiffness matrices, setup of the preconditioner and solution phase. See also Fig. 5.6 for detailed setup and solve times for the largest experiment. Computed on the magnitUDE supercomputer. The shown results have already been published in [103].

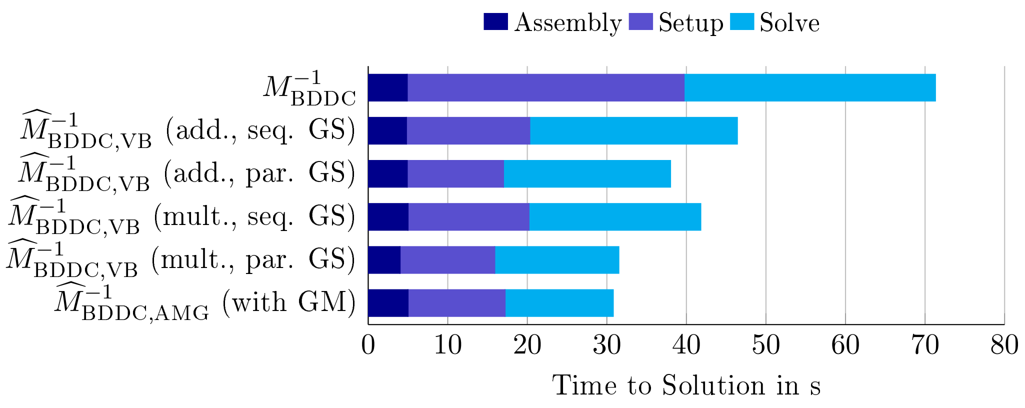


Figure 5.6: Comparison of BDDC setup and solve times of M^{-1}_{BDDC}, $\widehat{M}^{-1}_{\text{BDDC,VB}}$ with different Gauss-Seidel setups, and $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ with GM. Using edge translations as constraints. Homogeneous linear elastic cube decomposed into 5 832 subdomains with $H/h = 22$. The corresponding weak scaling experiment can be found in Fig. 5.5. Computed on magnitUDE supercomputer. The shown results have already been published in [103].

preconditioner $\widehat{M}^{-1}_{\text{BDDC,VB}}$ can be an adequate alternative to the classic two-level BDDC method. An advantage of $\widehat{M}^{-1}_{\text{BDDC,VB}}$ is the fact that neither a further decomposition into subregions is necessary nor an appropriate AMG method has to be chosen. On the other hand, the parallel potential of $\widehat{M}^{-1}_{\text{BDDC,VB}}$ as a two-level method is limited. For a larger number of subdomains, a three-level extension of $\widehat{M}^{-1}_{\text{BDDC,VB}}$ would be necessary, which is, to the best of our knowledge, not available yet. Another possible extension of $\widehat{M}^{-1}_{\text{BDDC,VB}}$ could be to combine it with the recursive structure of the three-level BDDC approach and to compute a vertex-based coarse space on the coarsest level of a three-level BDDC method. A numerical investigation of such a combined method, especially with regards to parallel scalability, is still a topic of future research.

To conclude this section, let us note that the presented approximate BDDC methods are not expected to be robust for completely arbitrary coefficient distributions. Similar as for the classic BDDC method, we expect the condition numbers to deteriorate for high contrasts in the coefficient function or the material distribution of the considered model problem; see, e.g., [101] for $\widehat{M}^{-1}_{\text{BDDC,AMG}}$ and [41] for $\widehat{M}^{-1}_{\text{BDDC,VB}}$. To further improve the robustness of the considered approaches while preserving the advanced parallel scalability, for instance, multilevel adaptive methods have been proposed for the BDDC method [160, 161]. These approaches combine the computation of an adaptive, i.e., a problem-dependent coarse space (see also Section 3.5) with the benefits of a multilevel method to increase the parallel potential. Furthermore, we propose to combine our three-level BDDC implementation with the frugal coarse space which will be presented in Chapter 6.

6 FETI-DP and BDDC with a frugal coarse space

As we have elaborated in the introduction and in Chapter 3, both the FETI-DP and the BDDC method obtain their robustness and parallel scalability from an appropriate coarse space, i.e., a second level. As we have seen in Sections 3.2.2 and 3.3.2, classic coarse spaces which are constructed by simply sub-assembling the system in selected primal variables using geometric information only guarantee a robust condition number estimate under certain restrictive assumptions on the coefficient function of the considered PDE. For more general and arbitrary coefficient functions with jumps along and across the interface, the classic condition number bounds do not hold anymore and the convergence behavior of the respective methods usually deteriorates.

In Section 3.5, we have introduced a very specific adaptive FETI-DP coarse space which relies on the solution of certain local generalized eigenvalue problems and uses selected eigenvectors to enhance the coarse space. By including these adaptive, i.e., problem-dependent constraints, the resulting algorithm is again robust with respect to highly heterogeneous coefficient functions for both, stationary diffusion and linear elasticity problems. In particular, we can prove a contrast-independent condition number estimate; cf. Section 3.5.2. However, as a drawback, in a parallel implementation the setup and the solution of the eigenvalue problems take up a significant amount of time of the entire time to solution. Additionally, for many realistic coefficient distributions, only a small number of the eigenvalue problems is actually necessary for robustness since for many edges and faces, only a single or a small number of constraints is necessary; see also Section 3.6 for a related discussion.

In this section, we propose a different and more heuristic coarse space. In principle, we aim to compute a low-dimensional approximation of the adaptive coarse space described in Section 3.5 by constructing generalized weighted averages along edges and faces of the domain decomposition. The resulting coarse space is supposed to be computationally inexpensive and robust for a broad range of different model problems. Earlier works [68,69,118] showed that heuristic coarse spaces, which approximate adaptive coarse spaces and do not require the solution of local generalized eigenvalue problems, can be constructed for overlapping Schwarz domain decomposition methods. Further approaches to design nonoverlapping domain decomposition methods which are robust for a broader range of coefficient distributions are presented in [2], where the domain is decomposed based on the coefficient distribution, and in [11], where deluxe scaling is introduced to obtain a more robust BDDC or FETI-DP method. In [58, 59], a robust coarse space for almost incompressible elasticity problems for the FETI-DP method is suggested.

Here, we construct constraints in a similar approach. We denote the proposed coarse space as a *frugal* approach due to the fact that the computed constraints do not require the solution of any eigenvalue problems or the explicit computation of Schur complements and are thus computationally very cheap. Let us note that the new frugal coarse space (FR) can be interpreted as a generalization of the weighted edge averages suggested in [111, Sect. 7, p.1412]; cf. also Section 3.4. In Section 6.1, we provide a heuristic motivation of the constructed frugal coarse space, which is strongly inspired by the adaptive coarse space presented in Section 3.5, and explicitly define the computed constraints. For the convenience of the reader, we first explain the construction for stationary diffusion problems in two spatial dimensions, which is the simplest case. We then extend the definitions and provide the respective formulae for both diffusion as well as linear elasticity problems in three spatial dimensions in Sections 6.2 and 6.3, respectively. In Sections 6.4 and 6.5, we provide first serial results for frugal FETI-DP and BDDC as well as results for our parallel frugal BDDC implementation compared to classic averages and the adaptive coarse space presented in Section 3.5. In our numerical experiments, we will consider a broader range of heterogeneities to prove that the frugal approach is robust for a wider range of coefficient distributions.

Parts of this chapter have already been published in modified or unmodified form in [73]. Additionally, first preliminary results for diffusion problems in two dimensions using the frugal constraints instead of constraints resulting from the solution of a specific edge eigenvalue problem were already published in [72, 74].

6.1 Heuristic motivation and construction for stationary diffusion in two dimensions

In the following section, we aim to give a heuristic motivation for the construction of our frugal coarse space, which is closely related to the adaptive FETI-DP coarse presented in Section 3.5 and the approach introduced in [111].

The following section has already been published by the author of this thesis and her coauthors in a slightly modified form in [73].

The construction of the frugal coarse space is strongly motivated by the generalized eigenvalue problem (3.26) from the adaptive coarse space [92, 130, 132]. Equivalently to the definition in (3.26), the respective generalized eigenvalue problem can also be written as

$$\langle \mathcal{H}(P_{D_{ij}} v_{ij}), K_{ij} \mathcal{H}(P_{D_{ij}} w_{ij}) \rangle = \mu_{ij} \langle \mathcal{H}(v_{ij}), K_{ij} \mathcal{H}(w_{ij}) \rangle = \mu_{ij} \langle \mathcal{H}(v_{E_{ij}}), K_{ij} \mathcal{H}(w_{E_{ij}}) \rangle,$$

where $K_{ij} = \operatorname{diag}(K^{(i)}, K^{(j)})$ and $\mathcal{H}(\cdot)$ is the discrete harmonic extension from the interface of Ω_i and Ω_j to the interior of the subdomains Ω_i and Ω_j; cf., e.g., [165, Sect. 4]. Then, as described in Section 3.5, all eigenmodes which fulfill the condition

$$\mu_{ij} = \frac{|\mathcal{H}(P_{D_{ij}} v_{ij})|_{K_{ij}}}{|\mathcal{H}(v_{ij})|_{K_{ij}}} > TOL \tag{6.1}$$

with eigenvalues $\mu_{ij} \geq 0$ are selected and are subsequently used to construct the adaptive constraints which are integrated into the coarse space. In particular, we select the respective eigenmodes where $|\mathcal{H}(P_{D_{ij}} v_{ij})|_{K_{ij}}$ is large, i.e., in the order of the contrast of the coefficient function, while $|\mathcal{H}(v_{ij})|_{K_{ij}}$ is small.

Now, in our new frugal approach, we propose a specific construction of an edge function $v_{E_{ij}}$ which often has the desired properties of the energies $|\mathcal{H}(P_{D_{ij}} v_{E_{ij}})|_{K_{ij}}$ and $|\mathcal{H}(v_{E_{ij}})|_{K_{ij}}$ as described above. Therefore, the space spanned by all edge functions $v_{E_{ij}}$ can be regarded as a lower-dimensional approximation of the original adaptive coarse space.

For simplicity, let us first consider the case of stationary diffusion problems in two spatial dimensions. In this case, we only compute constraints corresponding to the edges of the domain decomposition. As before, we denote by $\mathcal{E}_{ij}$ the edge between two neighboring subdomains Ω_i and Ω_j. We further denote by $\omega(x)$ the support of the finite element basis functions associated with a finite element node $x \in (\Omega_i \cup \Omega_j)$. For each node x on $\partial\Omega_i$ or $\partial\Omega_j$, respectively, we then compute

$$\widehat{\rho}^{(i)}(x) = \max_{y \in \omega(x) \cap \Omega_i} \rho(y)$$

and

$$\widehat{\rho}^{(j)}(x) = \max_{y \in \omega(x) \cap \Omega_j} \rho(y).$$

Then, in a second step, we define $v_{E_{ij}}^{(l)}$ on $\partial\Omega_l$ for $l = i, j$ by

$$v_{E_{ij}}^{(l)}(x) := \begin{cases} \widehat{\rho}^{(l)}(x), & x \in \partial\Omega_l \backslash \Pi^{(l)}, \\ 0, & x \in \Pi^{(l)} \end{cases} \tag{6.2}$$

and

$$v_{E_{ij}}^T := (v_{E_{ij}}^{(i)T}, -v_{E_{ij}}^{(j)T}).$$

Here, $\Pi^{(l)}$ denotes the index set of all local primal variables of Ω_l for $l \in \{i, j\}$. An exemplary visualization of this edge function for a specific heterogeneous coefficient distribution is presented in Fig. 6.1. Additionally, we provide visualizations of the two discrete harmonic extensions $\mathcal{H}(v_{E_{ij}})$ and $\mathcal{H}(P_{D_{ij}} v_{E_{ij}})$ for two different heterogeneous coefficient distributions of an exemplary model problem in Fig. 6.2 and Fig. 6.3. As can be observed from Fig. 6.2 (left) and Fig. 6.3 (left), in both cases, the energy $|\mathcal{H}(v_{E_{ij}})|_{K_{ij}}$ is low. On the other hand, the energy $|\mathcal{H}(P_{D_{ij}} v_{E_{ij}})|_{K_{ij}}$ is large for those two examples; see in Fig. 6.2 (right), where the energy is large due to the homogeneous Dirichlet boundary enforced by $P_{D_{ij}}$, and Fig. 6.3 (right), where the energy is large due to the gradient of the scaled jump $P_{D_{ij}} v_{E_{ij}}$ on the edge.

Finally, we obtain the frugal edge constraint by

$$c_{E_{ij}} := B_{D_{ij}} S_{ij} P_{D_{ij}} v_{E_{ij}} \tag{6.3}$$

as in the adaptive coarse space; cf. Section 3.5. Let us recapitulate that we denote by $B_{D_{ij}}$ the submatrix of $(B_D^{(i)}, B_D^{(j)})$ with the rows restricted to the edge $\mathcal{E}_{ij}$. We further have defined

$$S_{ij} = \begin{pmatrix} S^{(i)} & 0 \\ 0 & S^{(j)} \end{pmatrix}$$

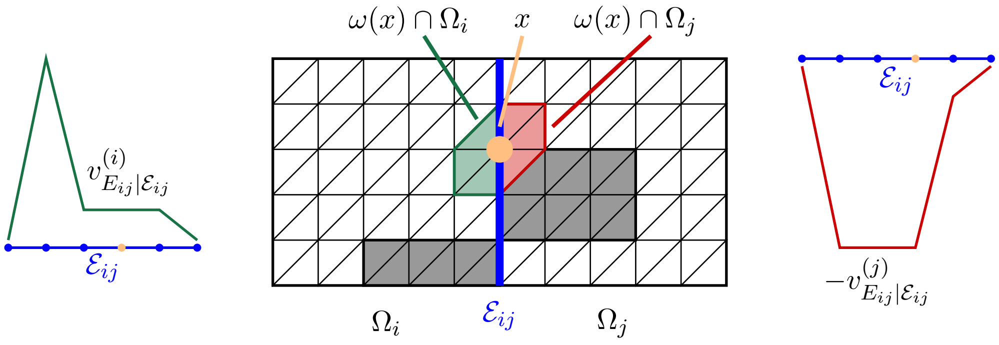

$v^{(i)}_{E_{ij}|\mathcal{E}_{ij}}$
$\mathcal{E}_{ij}$
$\omega(x) \cap \Omega_i$
x
$\omega(x) \cap \Omega_j$
Ω_i
$\mathcal{E}_{ij}$
Ω_j
$\mathcal{E}_{ij}$
$-v^{(j)}_{E_{ij}|\mathcal{E}_{ij}}$

Figure 6.1: Visualization of the construction of a frugal edge constraint in two dimensions for a given heterogeneous coefficient distribution. **Left:** Maximum coefficient per finite element node of $\mathcal{E}_{ij}$ with respect to Ω_i for the coefficient distribution in the middle. **Middle:** Exemplary heterogeneous coefficient distribution for two neighboring subdomains Ω_i and Ω_j sharing the edge $\mathcal{E}_{ij}$. High coefficients are marked in grey and low coefficients are marked in white. **Right:** Maximum coefficient per finite element node of $\mathcal{E}_{ij}$ with respect to Ω_j for the coefficient distribution in the middle. Figure in modified form in [73].

in Section 3.5, where $S^{(i)}$ and $S^{(j)}$ are the Schur complement matrices of $K^{(i)}$ and $K^{(j)}$, respectively, with respect to the interface variables, as well as the operator $P_{D_{ij}} = B^T_{D_{ij}} B_{ij}$.

Let us note that the construction (6.2) can be further simplified by exploiting the fact that the scaled jumped operator $P_{D_{ij}}$ is zero everywhere except on the edge $\mathcal{E}_{ij}$. This has the effect that, after the application of $P_{D_{ij}}$ in (6.3) to a computed constraint vector $v_{E_{ij}}$, all entries which do not correspond to degrees of freedom associated with the edge $\mathcal{E}_{ij}$ are set to zero. Therefore, our new constraint can equivalently also be constructed as

$$v^{(l)}_{E_{ij}}(x) = \begin{cases} \widehat{\rho}^{(l)}(x), & x \in \mathcal{E}_{ij} \\ 0, & x \in \partial\Omega_l \setminus \mathcal{E}_{ij} \end{cases} \tag{6.4}$$

for $l = i, j$; cf. the definition in [72]. In particular, due to the subsequent application of $P_{D_{ij}}$ in (6.3), both definitions of $v^{(l)}_{E_{ij}}$ result in the same edge constraints $c_{E_{ij}}$.

For our parallel implementation, we use the latter definition of $v^{(l)}_{E_{ij}}$. There, we exploit the extension by zero to the remaining interface $\partial\Omega_l \setminus \mathcal{E}_{ij}$ and reduce the applications of several local $P_{D_{ij}}$-operators to a few global applications of P_D; see also Section 6.5 for more details.

6.2 A frugal coarse space for stationary diffusion in three dimensions

In principle, the extension of the proposed frugal edge constraints in Section 6.1 to diffusion problems in three spatial dimensions is relatively straightforward. The main

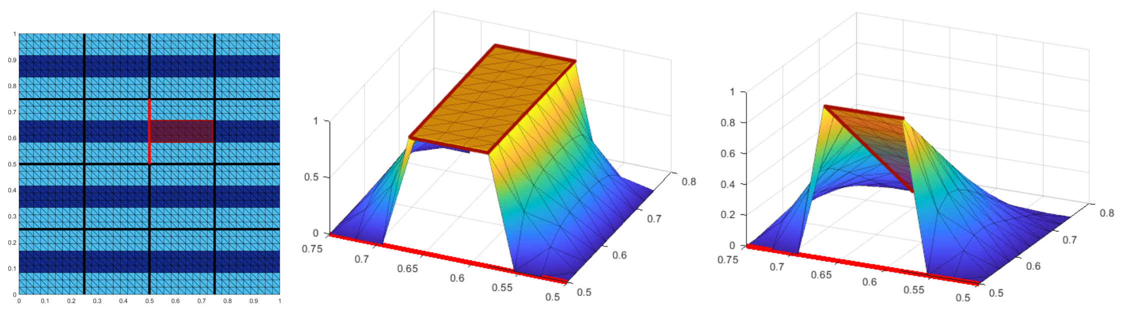


Figure 6.2: Visualization of the discrete harmonic extensions relevant for the motivation of the frugal coarse space for a heterogeneous coefficient distribution which is symmetric with respect to all edges. **Left:** Coefficient distribution with one channel associated with a high coefficient crossing each subdomain. Dark blue corresponds to the high coefficient ($\rho = 1e6$) and light blue to the low coefficient ($\rho = 1$). Visualization for 4×4 subdomains and $H/h = 9$. **Middle:** Visualization of the discrete harmonic extension $\mathcal{H}(v_{E_{ij}})$ for a floating subdomain. The visualized constraint leads to an energy of 17.49. **Right:** Visualization of the discrete harmonic extension $\mathcal{H}(P_{D_{ij}} v_{E_{ij}})$ for the same floating subdomain. The visualized constraint leads to an energy of $6.67e{+}5$. Taken from [73].

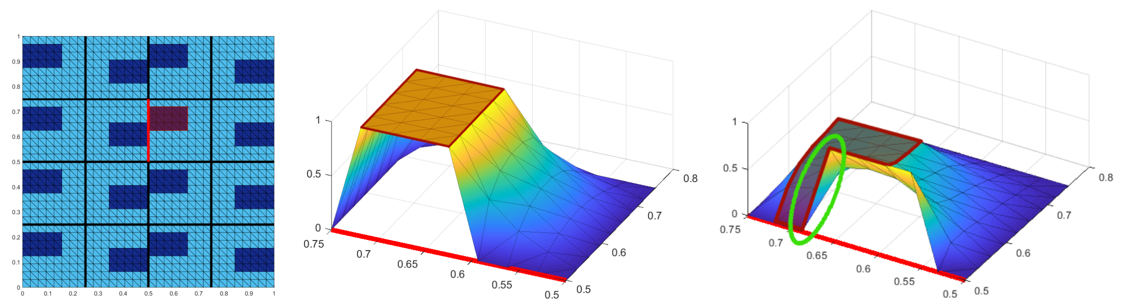


Figure 6.3: Visualization of the discrete harmonic extensions relevant for the motivation of the frugal coarse space for a heterogeneous coefficient distribution which is asymmetric with respect to the relevant edges. **Left:** Coefficient distribution with shifted boxes associated with a high coefficient. Dark blue corresponds to the high coefficient ($\rho = 1e6$) and light blue to the low coefficient ($\rho = 1$). Visualization for 4×4 subdomains and $H/h = 8$. **Middle:** Visualization of the discrete harmonic extension $\mathcal{H}(v_{E_{ij}})$ for a floating subdomain, which is nearly constant in the area with a high coefficient marked in red. The visualized constraint leads to an energy of 17.39. **Right:** Visualization of the discrete harmonic extension $\mathcal{H}(P_{D_{ij}} v_{E_{ij}})$ for the same floating subdomain, which has high gradients (see green ellipse) in the area with a high coefficient marked in red. The visualized constraint leads to an energy of $1.96e{+}6$. Taken from [73].

difference is that we now compute our new constraint for open faces $\mathcal{F}_{ij}$ or, alternatively, closed faces $\overline{\mathcal{F}}_{ij}$ which are shared by the two neighboring subdomains Ω_i and Ω_j instead of for edges $\mathcal{E}_{ij}$.

For a mathematical description, let us first define

$$v_{F_{ij}}^{(l)}(x) := \begin{cases} \widehat{\rho}^{(l)}(x), & x \in \mathcal{F}_{ij}, \\ 0, & x \in \partial\Omega_l \setminus \mathcal{F}_{ij}, \end{cases} \tag{6.5}$$

and

$$v_{\overline{F}_{ij}}^{(l)}(x) := \begin{cases} \widehat{\rho}^{(l)}(x), & x \in \overline{\mathcal{F}}_{ij}, \\ 0, & x \in \partial\Omega_l \setminus \overline{\mathcal{F}}_{ij}. \end{cases} \tag{6.6}$$

Analogously to the two-dimensional case, we then obtain our frugal constraints $c_{\overline{F}_{ij}}$ by applying $B_{D_{ij}} S_{ij} P_{D_{ij}}$ to either $v_{F_{ij}}$ or $v_{\overline{F}_{ij}}$, respectively. Thus, we define

$$v_{F_{ij}}^T := (v_{F_{ij}}^{(i)T}, -v_{F_{ij}}^{(j)T})$$

for open faces and

$$v_{\overline{F}_{ij}}^T := (v_{\overline{F}_{ij}}^{(i)T}, -v_{\overline{F}_{ij}}^{(j)T})$$

for closed faces, respectively, as well as

$$c_{\overline{F}_{ij}} := B_{D_{ij}} S_{ij} P_{D_{ij}} v_{F_{ij}} \tag{6.7}$$

for the constraint vector $v_{F_{ij}}$ on open faces, and replace $v_{F_{ij}}$ by $v_{\overline{F}_{ij}}$ in (6.7) for the corresponding closed face variant. Let us remark that, due to the structures of the Schur complement matrix S_{ij} as well as $P_{D_{ij}}$, in both cases the obtained constraint $c_{\overline{F}_{ij}}$ can have nonzero entries on the closed face, i.e., on the open face $\mathcal{F}_{ij}$ and on all edges $\mathcal{E}_m, m = 1, 2, \ldots, M$ belonging to the closed face $\overline{\mathcal{F}}_{ij}$. We can therefore split a constraint $c_{\overline{F}_{ij}}$ into a constraint $c_{F_{ij}}$ on the open face and constraints c_{E_m}, $m = 1, ..., M$, on the neighboring edges. The same approach of splitting the constraints into face- and edge-related parts has already been proposed in [92, 119] for the adaptive FETI-DP coarse space in Section 3.5.

Depending on the decision whether we also integrate the edge-related components into the coarse space or not, we can define four different possible variants of the frugal coarse space based on face constraints:

FR1: Compute $v_{\overline{F}_{ij}}^T$ for the closed face $\overline{\mathcal{F}}_{ij}$, and enforce both the edge-related components c_{E_m}, $m = 1, ..., M$, as well as the component related to the open face of $c_{F_{ij}}$.

FR2: Compute $v_{\overline{F}_{ij}}^T$ for the closed face $\overline{\mathcal{F}}_{ij}$, but just extract the terms of $c_{F_{ij}}$ related to the open face, while discarding the respective edge-related components.

FR3: Compute $v_{F_{ij}}^T$ for the open face $\mathcal{F}_{ij}$, and enforce both the edge-related components c_{E_m}, $m = 1, ..., M$, as well as the component related to the open face of $c_{F_{ij}}$.

FR4: Compute $v_{F_{ij}}^T$ for the open face $\mathcal{F}_{ij}$, but just extract the terms of $c_{F_{ij}}$ related to the open face, while discarding the respective edge-related components.

In Section 6.4, we will compare the robustness of the four different variants with respect to the condition number estimates and iteration counts for a range of different model problems. Let us remark that FR1 implements the frugal constraints in closest analogy to the two-dimensional case in Section 6.1. With respect to a parallel implementation, FR4 is the most promising variant, since the different open faces have no intersections with each other and therefore many local operations, such as, e.g., local applications of $P_{D_{ij}}$, can be grouped to global operations and can thus be carried out for all faces at once; see also Section 6.5.

6.3 A frugal coarse space for linear elasticity in three dimensions

For the case of three-dimensional linear elasticity problems, we have to slightly extend our definitions made above of the frugal face constraints. For linear elasticity problems in three dimensions, we know that when applying the FETI-DP or BDDC algorithm, we need six constraints, i.e., six rigid body modes to control the null space of subdomains which have boundaries that do not intersect $\partial\Omega_D$; cf. also the explanations at the beginning of Section 3.2.1. Let us recall that for a generic domain $\widehat{\Omega}$ with diameter H, the basis for the null space $\ker(\varepsilon)$ is given by the three translations

$$r_1 := \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad r_2 := \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad r_3 := \begin{bmatrix}0\\0\\1\end{bmatrix}, \tag{6.8}$$

and the three linear (approximations to) rotations

$$r_4 := \frac{1}{H}\begin{bmatrix}x_2-\widehat{x}_2\\-x_1+\widehat{x}_1\\0\end{bmatrix}, \quad r_5 := \frac{1}{H}\begin{bmatrix}-x_3+\widehat{x}_3\\0\\x_1-\widehat{x}_1\end{bmatrix}, \quad r_6 := \frac{1}{H}\begin{bmatrix}0\\x_3-\widehat{x}_3\\-x_2+\widehat{x}_2\end{bmatrix}, \tag{6.9}$$

where $\widehat{x} \in \widehat{\Omega}$ is the center of the linear rotations; see, e.g., [111, Sect. 2]. We now construct six weighted constraints per face to set up a frugal coarse space. In principle, these constraints are based on the maximum coefficients per element, i.e., the maximum Young modulus $E > 0$, as well as on the three translations and the three rotations for the respective face shared by two neighboring subdomains. For a mathematical description, let $\mathcal{F}_{ij}$ be the open face shared by the two neighboring subdomains Ω_i and Ω_j, and $\overline{\mathcal{F}}_{ij}$ the respective closed face. For each finite element node x on $\partial\Omega_i$ or $\partial\Omega_j$, we then compute

$$\widehat{E}^{(l)}(x) = \max_{y\in\omega(x)\cap\Omega_l} E(y)$$

for $l = i, j$. Subsequently, we compute six scaled rigid body modes denoted by $\widehat{r}_m^{(l)}, m = 1,\ldots,6,\ l = i,j$, by pointwise scaling the rigid body modes $r_m, m = 1,\ldots,6$, in (6.8) and (6.9) with the respective vectors of maximum coefficients $\widehat{E}^{(l)}(x), l = i, j$. Note that (in three dimensions) all three degrees of freedom belonging to a given node $x \in \partial\Omega_i \cup \partial\Omega_j$ are scaled with the same value of $\widehat{E}^{(l)}(x)$ for $l = i, j$. For $m = 1, \ldots, 6$ and $l = i, j$, we then define

$$v_{F_{ij}}^{(m,l)}(x) := \begin{cases} \widehat{r}_m^{(l)}(x), & x \in \mathcal{F}_{ij} \\ 0, & x \in \partial\Omega_l \setminus \mathcal{F}_{ij}. \end{cases} \tag{6.10}$$

By again combining the two vectors for both neighboring subdomains to

$$v_{F_{ij}}^{(m)T} := (v_{F_{ij}}^{(m,i)T}, -v_{F_{ij}}^{(m,j)T})$$

we obtain the weights for the six face constraints by

$$c_{F_{ij}}^{(m)} := B_{D_{ij}} S_{ij} P_{D_{ij}} v_{F_{ij}}^{(m)},\ m = 1, \ldots, 6. \tag{6.11}$$

The variants for closed faces are defined completely analogously; see also (6.6). Thus, the four different variants FR1 to FR4 of frugal coarse spaces can be implemented as in the diffusion case. Please note that the resulting constraints can, in certain cases, be linearly dependent and thus result in less than six constraints per face. Therefore, we always apply a modified Gram-Schmidt algorithm after constructing the six aforementioned constraints in our implementation. For the case of linear elasticity in two dimensions, the computation of $c_{E_{ij}}^{(m)}, m = 1, \ldots, 3$, is completely analogous to the three-dimensional case. For two dimensions, we just scale the two degrees of freedom belonging to a given node x with the same value of $\widehat{E}^{(l)}(x)$ for $l = i, j$. Since the three-dimensional case is more general, we here chose to describe the three-dimensional case in more detail.

Due to the possible existence of hinge modes for two neighboring subdomains in case of linear elasticity problems, using only face constraints and edge constraints arising as a byproduct, as, e.g., in variants FR1 and FR3, might not always lead to a robust algorithm for complex coefficient distributions. In particular, in some cases the use of additional edge constraints is necessary to obtain moderate condition number bounds as well as parallel scalability. We will explicitly consider such a coefficient distribution in Table 6.7, where the exclusive implementation of frugal face constraints is not sufficient to obtain a robust coarse space. For this special case, we enforce additional frugal edge constraints besides the generalized weighted face and edge constraints as already introduced. The construction of these edge constraints is, in principle, completely analogous to the aforementioned face constraints. More precisely, the construction is basically the same except that we now operate on the index set of open edges between two neighboring subdomains that do not share a face. Alternatively, we can additionally construct frugal edge constraints for all edges, for simplicity, and finally apply a modified Gram-Schmidt algorithm to eliminate linearly dependent constraints. We thus obtain an additional fifth variant of the frugal coarse space in three dimensions, which we denote by **FR5** and which will be numerically tested in Section 6.4.2. Note that this approach is motivated by an analogous procedure for an adaptive coarse space in [92] which is an extension of the variant presented in [130]. Let us remark that in our implementation, we first use FR1 for all faces and ensure that we have no redundancies in the face-related constraints and additional frugal edge constraints by applying a Gram-Schmidt orthogonalization.

6.4 Numerical results for frugal FETI-DP and BDDC

In this section, we present numerical results for the frugal coarse space obtained using our serial MATLAB [134] implementations of the FETI-DP and BDDC algorithms.

Parts of this section have already been published in modified or unmodified form in [73].

In the following, we consider three-dimensional stationary diffusion and linear elasticity problems on the unit cube, $\Omega = [0,1]^3$, with Dirichlet boundary conditions on the left-hand side of the boundary $\partial\Omega$, i.e., $\partial\Omega_D := 0 \times [0,1]^2$. In all presented numerical experiments, we use the ρ-scaling approach and, as the iterative solver, the PCG algorithm. As the stopping criterion for PCG we use a relative reduction of the preconditioned residual by a factor of $1e$-8. As already mentioned in Sections 3.2.2 and 3.3.2, we obtain the same quantitative condition number bounds for FETI-DP and BDDC since the two methods are dual to each other; see also [125,129]. Therefore, we do not provide results for both methods for all tested coefficient distributions.

Let us remark that, in addition to the implemented face- or edge-based frugal or classic constraints, respectively, we always choose all vertices as primal variables. In all tables and figures, we use the following notation to distinguish between the different classic coarse spaces based on weighted averages; see also Section 3.4:

e: Using vertex constraints and edge constraints (e), i.e., enforcing (3.20) for all edges $\mathcal{E}$. In case of linear elasticity, only weighted translations are enforced, i.e., we have $l = 3$ in (3.20).

f: Using vertex constraints and face constraints (f), i.e., enforcing (3.20) for all faces $\mathcal{F}$. In case of linear elasticity, only weighted translations are enforced, i.e., we have $l = 3$ in (3.20).

f+r: Using vertex constraints and face constraints (f), i.e., enforcing (3.20) for all faces $\mathcal{F}$. In case of linear elasticity, translations and rotations (r) are enforced, i.e., we have $l = 6$ in (3.20).

Let us note that, instead of fixing the global coefficient distribution, we always keep the coefficient distribution fixed for each subdomain in our weak scaling studies. Consequently, the global coefficient distribution actually changes while increasing the number of subdomains.

6.4.1 Straight and shifted beams

As a first set of experiments, we provide numerical results for the frugal FETI-DP coarse space for straight and shifted beams of high coefficients crossing each subdomain as shown in Figs. 6.4 and 6.5. We consider both stationary diffusion problems as well as linear elasticity problems and compare the results for our frugal approach to the classic approach from [111]. Here, we explicitly compare all four variants of the frugal coarse space as introduced in the beginning of this section within our serial implementation to evaluate the differences in their performance and rates of convergence. For the straight channels which only cut through faces in Table 6.1, all four variants FR1 to FR4 of the

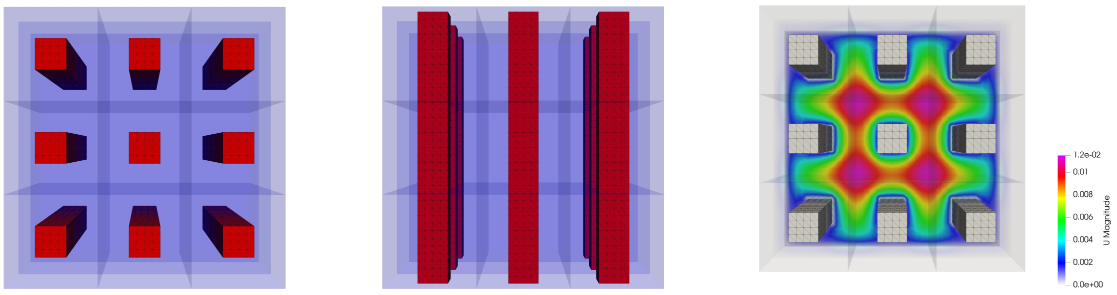


Figure 6.4: **Left** and **Middle:** Coefficient function with one central beam per subdomain. High coefficients are shown in red, and subdomains are shown in purple and by half-transparent slices. **Right:** Visualization of the corresponding solution for a stationary diffusion problem. Visualization for $3 \times 3 \times 3 = 27$ subdomains and $H/h = 12$. Taken from [73].

frugal coarse space show a more or less equivalent performance. In this case, the classic approach also provides comparable and robust condition number bounds and iteration counts. For the shifted channels, see Table 6.2, the edge-related frugal variants FR1 and FR3 show a slightly better performance compared to FR2 and FR4 which exclusively include face-related components. This effect is mostly noticeable for the linear elasticity problem presented in Table 6.2. However, also the exclusively face-based variants FR2 and FR4 show robust condition numbers which are independent of the coefficient jump in all computations. On the contrary, in this case, the classic weighted averages are not sufficient to provide a robust preconditioner since they result in condition numbers which depend on the coefficient contrast. This observation shows that the frugal approach is indeed more general than classic averages and provides robustness for more complex coefficient distributions.

To provide a more detailed investigation of the robustness and performance of the proposed frugal coarse space, we present further results for varying contrasts of the coefficients. In Table 6.3, we show the condition number estimates and iteration numbers for a stationary diffusion problem and the shifted beams as in Fig. 6.5 with an increasing contrast between the two different coefficients for the frugal and the classic FETI-DP coarse space. As we can observe from Table 6.3, the classic coarse space performs clearly poorer than the frugal coarse space for all tested coefficient contrasts. In particular, for the higher coefficient $\rho \in \{1e4, 1e5, 1e6\}$, the classic coarse space is not able anymore to provide a robust algorithm since the condition number estimates increase in proportion to the contrast of the coefficients. On the contrary, the frugal coarse space is proved to be robust with respect to the different coefficient contrasts.

We further provide exemplary experiments for stationary diffusion problems and two straight beams of a high coefficient crossing each subdomain in Table 6.4. We can observe that for a higher number of jumps in the coefficient function across and along faces of the domain decomposition, computing only one frugal constraint per face may not be sufficient. Analogously, also the classic coarse space does not provide robustness for

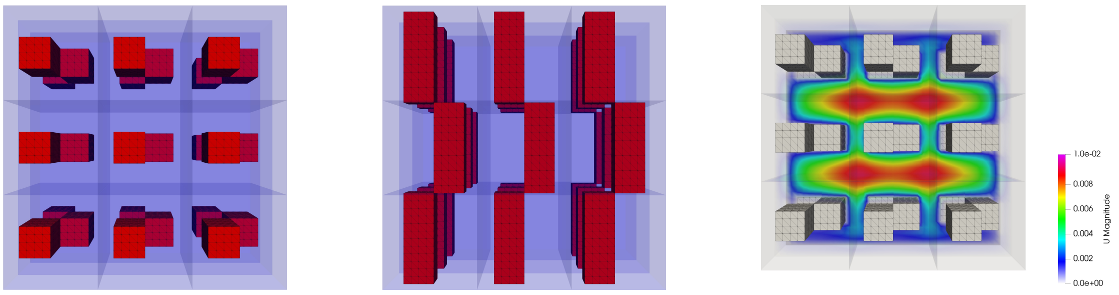


Figure 6.5: **Left** and **Middle:** Coefficient function with one beam per subdomain with offsets. High coefficients are shown in red, and subdomains are shown in purple and by half-transparent slices. **Right:** Visualization of the corresponding solution for a stationary diffusion problem. Visualization for $3 \times 3 \times 3 = 27$ subdomains and $H/h = 12$. Taken from [73].

this example. In this special case, computing a frugal constraint for each separate beam using algebraic information could restore the robustness; cf. also [68,69,118] for a related heuristic approach for overlapping Schwarz methods. In summary, these results provide an evidence that, in general, for completely arbitrary coefficient distributions, adaptive coarse spaces as presented in Section 3.5 are necessary to retain the robustness of the iterative solver.

Finally, we provide numerical results for the shifted channels and the frugal BDDC coarse space in Table 6.5 to prove that the frugal constraints work equally well for the FETI-DP and the BDDC algorithm. As expected, the condition number estimates and iteration counts for the frugal BDDC coarse space are very similar to those of the respective FETI-DP coarse space in Table 6.2. This is in accordance with the theory presented in Sections 3.2.2 and 3.3.2 concerning the duality of the two specific domain decomposition methods.

6.4.2 Cubic inclusions within subdomains

As a next example, we consider the case of inclusions of high coefficients within subdomains; see Fig. 6.6. Let us note that we only consider the case of inclusions which have a nonempty intersection with the respective subdomain boundary. For inclusions within subdomains which have a positive distance to the boundary of the respective subdomain, already standard FETI-DP and BDDC coarse spaces yield a robust domain decomposition method and no additional constraints have to be added to the coarse space; see, e.g., [58] for FETI-DP or [144] for FETI. We partition each subdomain into eight cubes of equal size and define a high coefficient in two of these cubes, which intersect only in a single vertex; see Fig. 6.6 for a schematic visualization. As the results in Table 6.6 show, the frugal face constraints lead to a robust algorithm for the diffusion problem for all variants FR1 to FR4. For elasticity problems, however, the resulting algorithm including only face constraints shows a bad convergence behavior or even diverges; see Table 6.7 (first column). For a comparison, we further include results for an adaptive FETI-DP

N	**frugal**								**classic**	
	FR1		**FR2**		**FR3**		**FR4**		**f**	
stationary diffusion										
	cond	it	cond	it	cond	it	cond	it	cond	it
2^3	1.25	5	1.44	6	1.25	5	1.44	6	1.44	7
3^3	1.25	6	1.51	8	1.25	6	1.51	8	1.51	10
4^3	1.25	6	1.53	8	1.25	6	1.53	8	1.53	10
linear elasticity										
	cond	it	cond	it	cond	it	cond	it	cond	it
2^3	1.59	10	2.70	13	1.59	10	2.71	14	2.71	14
3^3	1.63	11	2.78	16	1.62	10	2.78	16	2.78	15
4^3	1.63	11	2.86	16	1.62	10	2.85	16	2.85	16

Table 6.1: Condition numbers (cond) and iteration numbers (it) for frugal and classic FETI-DP coarse spaces for stationary diffusion and linear elasticity problems in three dimensions with $H/h = 6$ for one straight beam per subdomain as in Fig. 6.4. The higher coefficient is $\rho = 1e6$ for stationary diffusion and $E = 1e6$ for linear elasticity (with $\nu = 0.3$ everywhere). Table already published in [73, Table 4.1].

method in Table 6.7. Here, we use the adaptive coarse space as proposed by Mandel and Sousedík [130] and a variant thereof as implemented by Klawonn, Kühn, and Rheinbach [92] which has been developed to obtain a sound theoretical condition number bound; cf. also Section 3.5. Let us recall that for this specific adaptive FETI-DP method, the solution of certain local generalized eigenvalue problems on faces or edges is used to enrich the coarse space. In Table 6.7, we denote by:

a) **adaptive, face EVP:** the adaptive FETI-DP method from [130,132] using exclusively eigenvalue problems on faces;

b) **adaptive, edge EVP:** the adaptive FETI-DP method from [92], using additional eigenvalue problems on edges to enrich the coarse space.

As the results in Table 6.7 show, the adaptive FETI-DP algorithm also shows a bad convergence behavior for this specific linear elasticity problem when using exclusively face constraints. However, the use of additional edge constraints restores a robust algorithm, as the last column in Table 6.7 shows. This indicates that for this specific coefficient distribution with two cubes per subdomain of high coefficients only intersecting in a single vertex, a hinge mode exists in the case of linear elasticity, which is not controlled by our frugal face constraints. Thus, we also consider an additional, extended variant of the frugal coarse space which explicitly enforces frugal constraints on edges in addition to the already constructed (frugal) face constraints. We denote this variant by **FR5**. In the second column of Table 6.7, we present numerical results for this extended variant, given the inclusions per subdomain intersecting only in a single vertex as before. Please

	frugal								classic	
N	FR1		FR2		FR3		FR4		f	
stationary diffusion										
	cond	it	cond	it	cond	it	cond	it	cond	it
2^3	1.36	8	1.72	10	1.36	8	1.68	10	43 613.4	16
3^3	1.41	9	1.88	11	1.41	9	1.83	11	46 336.5	47
4^3	1.41	9	1.91	12	1.41	9	1.86	12	46 622.0	78
linear elasticity										
	cond	it	cond	it	cond	it	cond	it	cond	it
2^3	1.92	12	3.92	18	1.90	13	3.68	17	37 930.9	54
3^3	1.90	12	4.48	19	1.89	12	4.37	19	68 238.5	124
4^3	1.92	12	4.91	21	1.90	12	4.76	21	76 027.6	264

Table 6.2: Condition numbers (cond) and iteration numbers (it) for frugal and classic FETI-DP coarse spaces for stationary diffusion and linear elasticity problems in three dimensions with $H/h = 6$ for one beam per subdomain with offsets as in Fig. 6.5. The higher coefficient is $\rho = 1e6$ for stationary diffusion and $E = 1e6$ for linear elasticity (with $\nu = 0.3$ everywhere). Table already published in [73, Table 4.2].

note that the additional use of explicit edge constraints, in principle, corresponds to the solution of additional edge eigenvalue problems for subdomains that do not share a face in the context of the adaptive coarse space proposed in [92,119]; see also Section 3.5. To obtain a robust algorithm, we again use a modified Gram-Schmidt algorithm to eliminate all linearly dependent constraints resulting from edge-related parts of weighted face constraints and the explicitly constructed edge constraints.

6.4.3 Non-binary coefficient distribution with slowly varying coefficients

In the previous sections, we have exclusively considered binary coefficient distributions, i.e., coefficient functions with only two different values of the coefficient. In our experience, this is the hardest scenario since it results in a high number of sharp jumps of the coefficient across the interface, which negatively affects the rate of convergence of the iterative solver. However, in order to investigate our frugal coarse space also for more general coefficient distributions, we additionally consider a coefficient function which has more than two different coefficient values and which is constructed as follows. Similar as in Fig. 6.6, each cubic subdomain is decomposed into eight regular cubes of equal size. For each of these cubes, we randomly generate a single coefficient of the form 10^e with the exponent e being randomly uniformly distributed in $[1, 6] \cap \mathbb{Z}$. In Fig. 6.7, we show an exemplary visualization of such a coefficient distribution for $2 \times 2 \times 2$ subdomains and $H/h = 8$. The scaling of the included colorbar refers to the exponent e of the value 10^e for the respective coefficients.

We use this coefficient distribution for both stationary diffusion and linear elasticity

N	coeff$_2$	frugal, FR1			classic, f		
stationary diffusion							
		H/h	cond	it	H/h	cond	it
4^3	1e0	9	1.60	10	9	2.13	13
4^3	1e3	9	1.63	10	9	53.41	38
4^3	1e4	9	1.64	10	9	525.91	59
4^3	1e5	9	1.65	10	9	5 250.98	78
4^3	1e6	9	1.93	12	9	52 501.63	87

Table 6.3: Condition numbers (cond) and iteration numbers (it) for frugal and classic FETI-DP coarse spaces for stationary diffusion problems in three dimensions with $H/h = 9$ for one beam per subdomain with offsets as in Fig. 6.5. The higher coefficient is varied as $\text{coeff}_2 = 10^e, e \in \{0, 3, 4, 5, 6\}$, and the lower coefficient is 1. Table already published in [73, Table 4.4].

N	coeff$_2$	frugal, FR1			classic, f		
stationary diffusion							
		H/h	cond	it	H/h	cond	it
4^3	1e0	10	1.64	10	10	2.23	13
4^3	1e3	10	5.88	20	10	6.18	21
4^3	1e4	10	51.08	40	10	52.51	41
4^3	1e5	10	503.30	64	10	517.20	67
4^3	1e6	10	5 025.51	86	10	5 164.14	92

Table 6.4: Condition numbers (cond) and iteration numbers (it) for frugal and classic FETI-DP coarse spaces for stationary diffusion problems in three dimensions with $H/h = 10$ for two straight beams per subdomain. The higher coefficient is varied as $\text{coeff}_2 = 10^e, e \in \{0, 3, 4, 5, 6\}$, and the lower coefficient is 1. Table already published in [73, Table 4.5].

problems and our frugal coarse space variant FR1. The corresponding numerical results are summarized in Table 6.8. As we can observe from the condition number estimates and the iteration counts in Table 6.8, our proposed frugal approach is robust for this non-binary, random coefficient distribution. In particular, the variant FR1 shows condition numbers which are independent of the (maximum) coefficient contrast also for linear elasticity problems. In contrast, for the binary coefficient distribution in Fig. 6.6, additional frugal edge constraints are necessary to obtain robustness in case of linear elasticity problems; see Table 6.7. Due to these observations from Table 6.8 as well as due to different experimental results using adaptive coarse spaces, we believe that non-binary materials with more than two different random coefficients are in fact computationally easier than binary materials in the sense that a smaller number of jumps with a high contrast occurs at the interface between subdomains. Thus, in Section 6.5, we have decided to focus the numerical investigations on binary materials.

N	FR1		FR2		FR3		FR4	
stationary diffusion								
	cond	it	cond	it	cond	it	cond	it
2^3	1.29	8	1.72	11	1.28	8	1.68	11
3^3	1.35	9	1.88	12	1.33	9	1.83	11
4^3	1.35	9	1.90	13	1.34	9	1.86	12
linear elasticity								
	cond	it	cond	it	cond	it	cond	it
2^3	1.89	12	3.91	18	1.88	12	3.90	17
3^3	1.87	11	4.45	19	1.86	11	4.37	19
4^3	1.88	11	4.92	20	1.88	12	4.76	20

Table 6.5: Condition numbers (cond) and iteration numbers (it) for the frugal BDDC coarse space for stationary diffusion and linear elasticity problems in three dimensions with $H/h = 6$ for one beam per subdomain with offsets as in Fig. 6.5. The higher coefficient is $\rho = 1e6$ for stationary diffusion and $E = 1e6$ for linear elasticity (with $\nu = 0.3$ everywhere). Table already published in [73, Table 4.3].

6.4.4 A frugal coarse space of reduced dimension

As we have observed above, the introduced frugal FETI-DP or BDDC coarse space generally is of a similar size as the classic coarse space introduced in [111]. In particular, we construct frugal constraints for all faces and/or edges of the domain decomposition. However, for many real world problems, i.e., problems with realistic coefficient distributions, constraints on certain faces or edges are not necessary and can be omitted. Therefore, we further propose a modified variant to reduce the size of the frugal coarse space, which only requires moderate additional effort. Since the exact solution of the coarse problem can become a scaling bottleneck in a parallel implementation, the frugal coarse space of reduced size can potentially increase the parallel efficiency and reduce the total time to solution significantly, compared to our original frugal coarse space.

Let us recall that our frugal constraints are inspired by the eigenvalue problems introduced in [130,132]; see also Section 3.5. Hence, we can use the quotient (6.1) corresponding to the eigenvalue problems to select, i.e., to estimate the constraints actually required for a robust coarse space. Numerical experimental results have shown that for faces or edges, for which the left-hand side of the specific eigenvalue problem yields a high energy and the respective right-hand side yields a low energy, additional coarse constraints are essential for robustness. For the convenience of the reader, we again write down the right-hand side (RS) and the left-hand side (LS) of the eigenvalue problem (3.26):

$$LS := P_{D_{ij}}^T S_{ij} P_{D_{ij}} \quad \text{and} \quad RS := S_{ij}.$$

Please see Section 3.5 and [130, 132, 148] for more technical details on the eigenvalue problem. To estimate the energy of our computed frugal constraints, we evaluate the

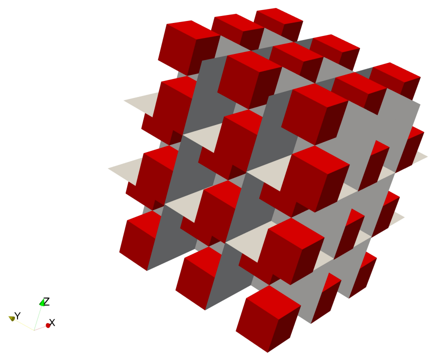


Figure 6.6: Coefficient distribution with inclusions of high coefficients within subdomains. High coefficients are shown in red, and subdomains are shown by grey slices. Visualization for $3 \times 3 \times 3$ subdomains and $H/h = 12$. Taken from [73].

N	**FR1**		**FR2**		**FR3**		**FR4**	
	stationary diffusion							
	cond	it	cond	it	cond	it	cond	it
2^3	3.55	14	3.55	14	3.55	14	3.55	14
3^3	4.05	20	4.05	20	4.05	20	4.05	20
4^3	4.41	22	4.41	22	4.41	21	4.41	22

Table 6.6: Condition numbers (cond) and iteration numbers (it) for the frugal FETI-DP coarse space for diffusion problems in three dimensions with $H/h = 8$ with two inclusions per subdomain as in Fig. 6.6. The higher coefficient is $\rho_1 = 1e6$ and the lower coefficient is $\rho_2 = 1$. Table already published in [73, Table 4.6].

product terms

$$RSe := v_{E_{ij}}^T RS v_{E_{ij}} \quad \text{and} \quad LSe := v_{E_{ij}}^T LS v_{E_{ij}}$$

in two spatial dimensions, or

$$RSe := v_{F_{ij}}^T RS v_{F_{ij}} \quad \text{and} \quad LSe := v_{F_{ij}}^T LS v_{F_{ij}}$$

in three spatial dimensions, respectively. In a next step, we evaluate the ratio LSe/RSe for all faces (and, depending on the chosen variant, for all edges). For the frugal coarse space of reduced size, we now only integrate face constraints into the coarse space, for which the ratio LSe/RSe is above a user-defined tolerance TOL; see, e.g., [78] for a discussion on the choice of TOL. We denote this reduced coarse space variant by **FR2-red.**. We show first numerical results for this reduced variant for straight channels of high coefficients in Table 6.9. For all shown cases, we are able to reduce the dimension of the

N	frugal						adaptive					
	FR1			**FR5**			**face EVP**			**edge EVP**		
	linear elasticity											
	# c.	cond	it	# c.	cond	it	# c.	cond	it	# c.	cond	it
2^3	288	25 158	54	324	1.72	10	39	58 679	58	173	3.70	15
3^3	1 452	18 530	180	1 668	1.73	10	164	87 156	246	838	3.42	20
4^3	4 032	19 626	232	4 680	1.74	10	429	114 882	471	2 319	3.44	20

Table 6.7: Condition numbers (cond), iteration numbers (it), and the size of the coarse space (# c.) for frugal and adaptive FETI-DP coarse spaces for linear elasticity problems in three dimensions with $H/h = 6$ with two inclusions per subdomain as in Fig. 6.6. The higher coefficient is $E_1 = 1e6$ and the lower coefficient is $E_2 = 1$, with $\nu = 0.3$ everywhere. Table already published in [73, Table 4.7].

N	FR1			FR1		
	stationary diffusion			**linear elasticity**		
	H/h	cond	it	H/h	cond	it
2^3	8	2.301	12	6	4.440	18
3^3	8	2.111	14	6	9.141	21
4^3	8	2.602	16	6	19.423	35

Table 6.8: Condition numbers (cond) and iteration numbers (it) for the frugal FETI-DP coarse space for stationary diffusion and linear elasticity problems in three dimensions with a non-binary, randomized coefficient distribution as in Fig. 6.7. For stationary diffusion, the coefficient ρ is of the form 10^e with e randomly uniformly distributed in $[1, 6] \cap \mathbb{Z}$ which is equally valid for the Young modulus E for linear elasticity (with $\nu = 0.3$ everywhere).

frugal coarse space to exactly one third of the original size while preserving both robust condition numbers and iteration counts. In Table 6.10, we further show numerical results for five spherical inclusions of different radii as shown in Fig. 6.12; see also Section 6.5 for detailed parallel results for this specific coefficient distribution. For the diffusion case, we are able to reduce the size of the coarse space up to a factor of 2.4. For the case of linear elasticity, the coarse space dimension is reduced by a factor of up to 1.6. At the same time, the condition numbers and iteration counts increase only in the same order of magnitude and especially remain independent of the coefficient contrast.

6.5 Parallel numerical results for frugal BDDC

For the numerical experiments presented in this section, we have added the frugal coarse space FR4 to our parallel BDDC implementation described in Section 5.4 and, in more details, in [101]. We compare the frugal coarse space with classic coarse spaces based on weighted edge or face translations and rotations as introduced in [111]; see also Sec-

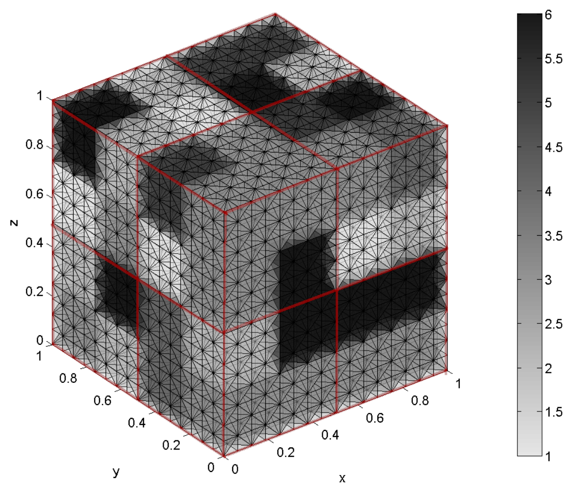


Figure 6.7: Non-binary coefficient distribution with randomly generated coefficients of the form 10^e with e randomly uniformly distributed in $[1,6] \cap \mathbb{Z}$. The scaling of the colorbar on the right refers to the exponent e of the value 10^e for the respective coefficients. Discretization with $2 \times 2 \times 2$ subdomains, visualized by the red slices, and $H/h = 8$.

tion 3.4. As a model problem, we consider stationary diffusion and linear elasticity problems. Unless stated otherwise, we use Dirichlet boundary conditions on the complete boundary of the domain Ω. As in Section 6.4, we use the PCG algorithm as the iterative solver for our computations. As the stopping criterion for PCG, we again use a relative reduction of the preconditioned residual by a factor of 1e-8.

Parts of this section have already been published in modified or unmodified form in [73].

6.5.1 Parallel implementation and computational effort

Let us recall from Chapter 5 that our parallel PETSc-based implementation of the BDDC method is based on a BDDC preconditioner for the complete system K_g; see also Section 5.1 for a mathematical description. Consequently, no Schur complement systems are built explicitly. Therefore, we also avoid the computation of the local Schur complement matrix S_{ij}, which is used for the construction of our frugal edge or face constraints; see (6.11). Instead of computing $S^{(i)} w_\Gamma^{(i)}$, we always compute the equivalent product

$$R_\Gamma^{(i)} \cdot K^{(i)} \cdot \begin{pmatrix} -\left(K_{II}^{(i)}\right)^{-1} K_{\Gamma I}^{(i)T} \\ I \end{pmatrix} w_\Gamma^{(i)}, \tag{6.12}$$

where $R_\Gamma^{(i)}$ is the restriction from the complete subdomain Ω_i to its interface and

$$-\left(K_{II}^{(i)}\right)^{-1} K_{\Gamma I}^{(i)T} w_\Gamma^{(i)}$$

N	FR2			FR2-red.		
linear elasticity						
	# c.	cond	it	# c.	cond	it
2^3	72	2.70	13	24	3.54	15
3^3	324	2.78	16	108	4.17	19
4^3	864	2.86	16	288	4.44	20

Table 6.9: Condition numbers (cond) and iteration numbers (it) for the frugal FETI-DP coarse space for linear elasticity problems in three dimensions with $H/h = 6$ for one straight beam per subdomain as in Fig. 6.4. Variant with **reduced** coarse space dimension and $TOL = 10$. The higher coefficient is $E_1 = 1e6$ and the lower coefficient is $E_2 = 1$, with $\nu = 0.3$ everywhere. Table already published in [73, Table 4.8].

is the discrete harmonic extension of $w_\Gamma^{(i)}$ from the interface to the interior of Ω_i. For the parallel implementation, we chose FR4 out of the different frugal variants, since this coarse space can be computed cheaply with an effort less than a few CG iterations; see the following discussion below. Additionally, FR4 showed promising results for most problems considered in the previous section.

Parallel computation of FR4 For simplicity, we only describe the implementation for the scalar diffusion case. Considering linear elasticity problems, the building blocks are identical and just called six times for each of the six rigid body modes.

Let us consider an exemplary face $\mathcal{F}_{ij}$ shared by the two subdomains Ω_i and Ω_j. Then, in our parallel BDDC implementation, we do not directly enforce the constraints

$$c_{F_{ij}} = B_{D_{ij}} S_{ij} P_{D_{ij}} v_{F_{ij}}$$

in the space of interface jumps, e.g., by a projector preconditioning approach, but equivalently the constraint

$$C_{F_{ij}}^{(i)T} = -C_{F_{ij}}^{(j)T}$$

by a generalized transformation-of-basis approach. In particular, we use an efficient implementation of the transformation of basis using local saddle point problems and an equivalent reformulation for the related Galerkin products; see Section 4.3 and [98] for more details. Here, we consider

$$C_{F_{ij}} = \left(C_{F_{ij}}^{(i)T}, C_{F_{ij}}^{(j)T} \right)$$

with

$$C_{F_{ij}} = P_{D_{ij}}^T S_{ij} P_{D_{ij}} v_{F_{ij}}; \tag{6.13}$$

cf. also (6.7). Let us note that with $P_D = I - E_D$, we compute

$$C_{F_{ij}} = (I - E_{D_{ij}})^T S_{ij} (I - E_{D_{ij}}) v_{F_{ij}}$$

N	FR2			FR2-red.		
stationary diffusion						
	# c.	cond	it	# c.	cond	it
2^3	12	6.90	19	5	17.02	22
3^3	54	3.64	17	29	18.92	27
4^3	144	5.59	22	74	18.55	39
linear elasticity						
	# c.	cond	it	# c.	cond	it
2^3	72	31.89	44	55	61.83	54
3^3	324	70.35	46	199	70.36	64
4^3	864	232.36	67	633	430.11	95

Table 6.10: Condition numbers (cond) and iteration numbers (it) for the frugal FETI-DP coarse space for diffusion and linear elasticity problems in three dimensions with $H/h = 10$ for five spherical inclusions as in Fig. 6.12. Variant with **reduced** coarse space dimension and $TOL = 100$. The higher coefficient is $\rho = 1e6$ for stationary diffusion and $E = 1e6$ for linear elasticity (with $\nu = 0.3$ everywhere). Table already published in [73, Table 4.9].

instead of (6.13), which is more convenient in the context of BDDC.

Let us recapitulate that in the FR4 variant, only open faces are considered and thus the computation of

$$x_{F_{ij}} := (I - E_{D_{ij}})v_{F_{ij}}$$

can be carried out for all faces at once. Here, we exploit the fact that the functions $v_{F_{ij}}$ can be chosen to be zero on the remaining interface; see Section 6.1 and the following arguments. Therefore, all values $v_{F_{ij}}$ for all faces $\mathcal{F}_{ij}$ are collected in a single vector v. The remaining interface components in v can be set arbitrarily. Then,

$$x := (I - E_D)v$$

can be computed in parallel using the parallel implementation of E_D based on PETSc *VecScatter* operations; see [101] for a detailed description of the implementation. All local vectors $x_{F_{ij}}$ for all open faces can be easily obtained from x by extracting local values on faces and extending them by zero to the remaining local interface. Additionally, let us remark that with $x_{F_{ij}} = (x^{(i)}_{F_{ij}}, x^{(j)}_{F_{ij}})$, the computation of $S_{ij}x_{F_{ij}}$ actually decomposes into two local computations $S^{(l)}x^{(l)}_{F_{ij}}$, $l = i, j$, which are carried out following (6.12). This process is completely local but (6.12) has to be computed separately for each face of a subdomain. The results for all local faces can again be collected in a single vector y and subsequently, $I - E_D$ can again be applied globally. Finally, all coarse constraints are locally extracted face by face from $(I - E_D)y$.

Computational effort As we have observed in Section 6.4, the frugal coarse space FR4 is of a similar size compared to classic face- or edge-based coarse spaces. However, we have to take into account that the computation of the constraints is more costly than for classic constraints, even though it requires significantly less computational effort than the computation of adaptive constraints which includes the solution of local generalized eigenvalue problems. Therefore, the effort will only pay off compared to classic approaches such as, e.g., those from [111], if a sufficient number of CG iterations is saved. To obtain a useful estimate for the computational effort of frugal constraints, we compare the cost to compute the coarse constraints to the cost of a specific number of CG iterations, i.e., a specific number of applications of the system matrix and the BDDC preconditioner.

In a single application of the BDDC preconditioner, the operator $(I - E_D)$ is applied twice, as here in the construction of the coarse space FR4. The discrete harmonic extension which appears in (6.12) is also applied twice in each application of the BDDC preconditioner. Finally, in each CG iteration, the matrix $K^{(i)}$ is applied once to a vector in the application of the system matrix. Considering six faces per subdomain for a regular decomposition, the construction of the coarse space has thus cost less than six CG iterations. Assuming that the computation of the discrete harmonic extensions is the most expensive operation in this process, we can approximately compare the computational cost with the cost of three CG iterations. Therefore, if we can save at least three CG iterations, the frugal coarse space FR4 will pay off. Of course, since the computation of the constraints of FR4 does not include any coarse solve, it will be even less expensive, especially for problems with many subdomains and compute cores. Therefore, the estimate with 3 CG iterations is in fact way too pessimistic.

For the case of linear elasticity with six rigid body modes for each of the six faces, we end up with an approximate cost of 18 CG iterations for the construction of the respective frugal face constraints - following the argumentation above. Again, we expect much less effort especially for larger problems. In practical experiments, e.g., considering the example from Fig. 6.11, we measure a time of approximately 8.1 CG iterations to construct the coarse space for the 48 subdomain case and only 2.4 CG iterations for the 3 072 subdomain case. For completeness, let us finally remark that the construction might be more expensive for irregular domain decompositions with more than six faces per subdomain.

6.5.2 Sanity check with a checkerboard problem

As a first sanity check of our parallel software, we provide results for the classic checkerboard problem shown in Fig. 6.8 for a stationary diffusion problem. Here, we have a single constant coefficient inside each subdomain which varies between two different values within the separate subdomains in an alternating checkerboard pattern. In Fig. 6.9, we provide the respective results for frugal and classic BDDC coarse spaces in terms of the number of CG iterations and the time to solution. As expected, using ρ-scaling, the classic coarse space with vertices and edges performs slightly better compared to face-based approaches since an acceptable edge path can be found; see [111]. Additionally, the frugal coarse space FR4 and the classic face constraints defined in Section 3.4 deliver similar results both with respect to the number of CG iterations and the time to solution.

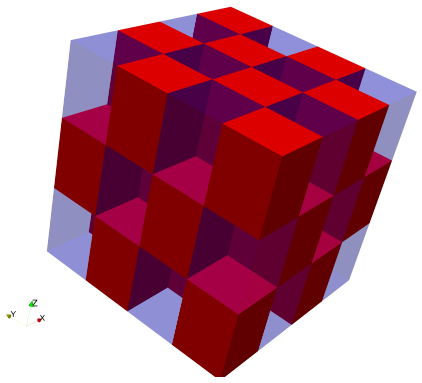


Figure 6.8: Coefficient distribution in a checkerboard pattern with constant coefficients per subdomain. High coefficients are shown in red and subdomains with a low coefficient are indicated by the purple slices. Visualization for $4 \times 4 \times 3$ subdomains and $H/h = 12$. Taken from [73].

6.5.3 Straight and shifted beams

As a next set of experiments, we provide parallel weak scaling results for straight and shifted beams as already considered in Section 6.4.1; see Figs. 6.4 and 6.5. Here, we consider both a stationary diffusion problem as well as a linear elasticity problem. The respective numerical results are presented in Fig. 6.10 and Fig. 6.11, respectively. For all examples, we can observe that face constraints are necessary to obtain satisfactory (weak) parallel scalability and a good rate of convergence. While for the straight channels FR4 is more or less equivalently robust compared to the classic face constraints (see Figs. 6.10 and 6.11 (left)), it is clearly superior for the more complicated model problem with shifted beams (see Figs. 6.10 and 6.11 (right)). In particular, the frugal coarse space FR4 is up to a factor of 9.2 faster in time to solution for the stationary diffusion problem. For the case of linear elasticity, the time to solution is reduced by a factor of up to 2.5.

6.5.4 Five spherical inclusions and an RVE

In this section, we investigate the proposed frugal coarse space for more general and more realistic coefficient distributions, which are chosen independently of the domain decomposition. For this purpose, we provide two additional examples.

Five Spherical Inclusions First, we consider five spherical inclusions of different radii in the unit cube, which share the same high coefficient; see Fig. 6.12. Let us remark that considering our structured mesh, each voxel is discretized by six tetrahedral finite elements and these six elements always share the same coefficient. Each voxel within the five spheres will have an identical high coefficient, i.e., a large ρ in the diffusion case or a large E in the linear elasticity case. The remaining matrix material will have a smaller coefficient. For the exemplary decomposition into 384 regular subdomains, we

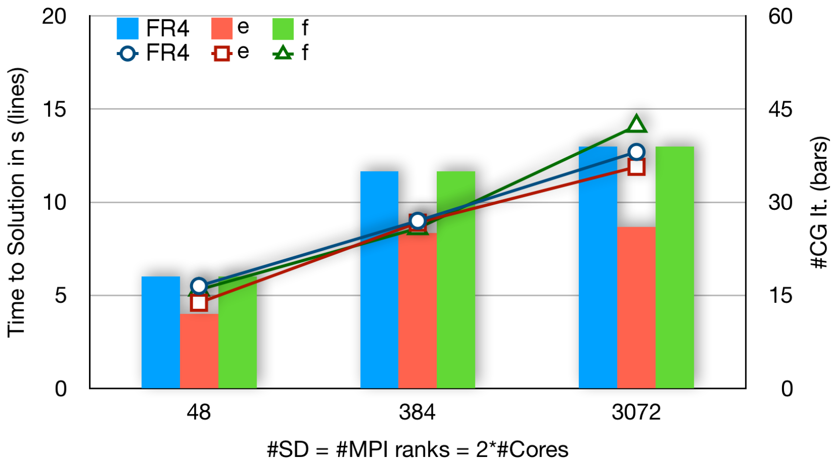


Figure 6.9: Parallel weak scaling test for frugal and classic BDDC coarse spaces; stationary diffusion problem with a constant coefficient on each subdomain, varying in a checkerboard pattern. The higher coefficient is $\rho = 1e6$ and the lower coefficient is $\rho = 1$; see Fig. 6.8. Taken from [73].

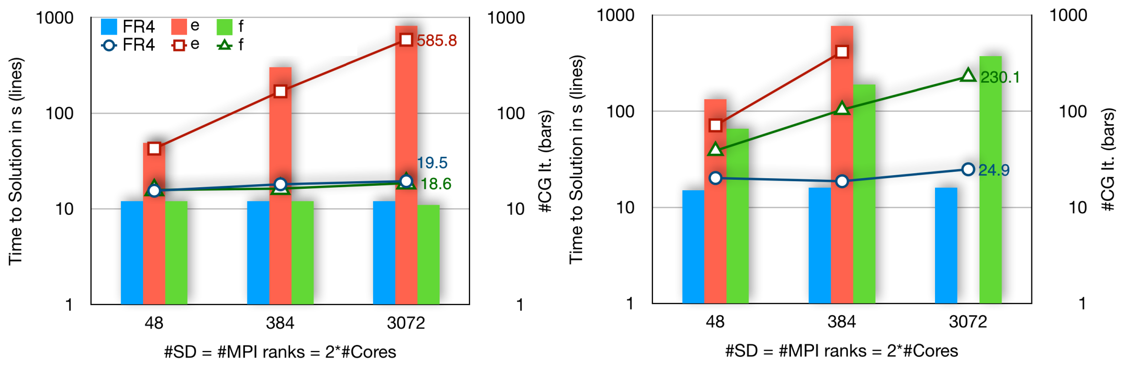


Figure 6.10: Parallel weak scaling test for frugal and classic BDDC coarse spaces; stationary diffusion problem with a single beam crossing each subdomain; higher coefficient $\rho = 1e6$ inside the beams and $\rho = 1$ in the remaining domain; **Left:** Straight beams; see Fig. 6.4. **Right:** Shifted beams; see Fig. 6.5. Missing data corresponds to runs which did not converge within 1 000 CG iterations. Taken from [73].

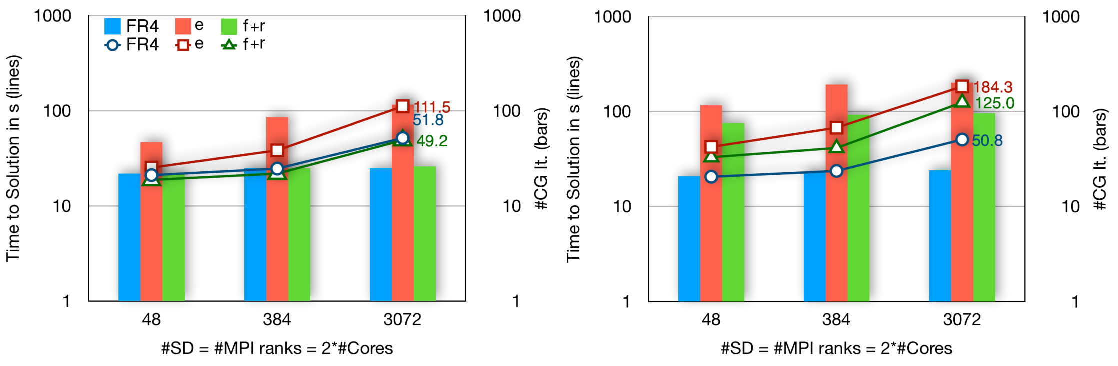


Figure 6.11: Parallel weak scaling test for frugal and classic BDDC coarse spaces; linear elasticity problem with one beam crossing each subdomain; higher coefficient $E = 210\,000$ inside the beams and $E = 210$ in the remaining matrix material; **Left:** Straight beams; see Fig. 6.4. **Right:** Shifted beams; see Fig. 6.5. Taken from [73].

illustrate a face between two subdomains (see Fig. 6.13 (left)) and mark the parts of the spheres which lie inside these two subdomains in blue and red, respectively. Zooming in (Fig. 6.13 (right)), we observe a similar situation as in the case of the shifted channels. Additionally, the spherical inclusions cut or touch also edges and vertices of the interface. In our experience, these coefficient jumps along and across the interface make this model problem difficult for the solution using an iterative solver and can lead to a deteriorating convergence behavior. Considering this model problem, the frugal coarse space FR4 outperforms all tested classic approaches significantly; see Table 6.11. Especially for the largest considered coefficient jump of 1e+6, FR4 alone is robust for both, diffusion and linear elasticity problems. The latter observation indicates, that the proposed frugal coarse space is indeed superior to classic coarse spaces for complex coefficient distributions.

RVE Second, we consider an RVE (representative volume element) of a dual-phase steel which consists of two different material phases, i.e., a martensite and a ferrite phase. The considered RVE represents the microscopic structure of a DP600 steel and is obtained by an EBDS (electronic backscatter diffraction) measurement. Let us note that this RVE is part of a larger structure presented in [23]. The martensitic inclusions are visualized in red in Fig. 6.14 (left) and the ferritic matrix material is marked in purple. The RVE is decomposed into 512 subdomains (see Fig. 6.14 (left)) and a corresponding linear elastic solution is shown in Fig. 6.14 (right). For our parallel computations, we use this coefficient distribution for diffusion and linear elasticity problems and define high coefficients in the martensitic phase and low coefficients in the ferritic phase. We summarize the results of a weak scaling study, i.e., the condition numbers, iteration counts, and the required time to solution as well as the size of the coarse space for frugal and classic BDDC in Table 6.12. Here, the most realistic computation is the linear elasticity problem with

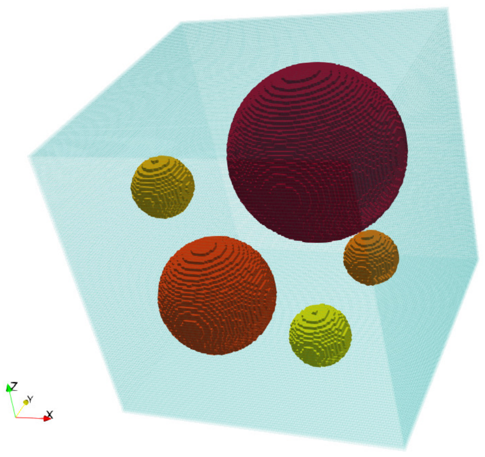


Figure 6.12: Five spheres with different radii in the unit cube. Resolution of $128 \times 128 \times 96$ voxel corresponding to computations with $H/h = 16$ and $8 \times 8 \times 6 = 384$ subdomains in Table 6.11. Taken from [73].

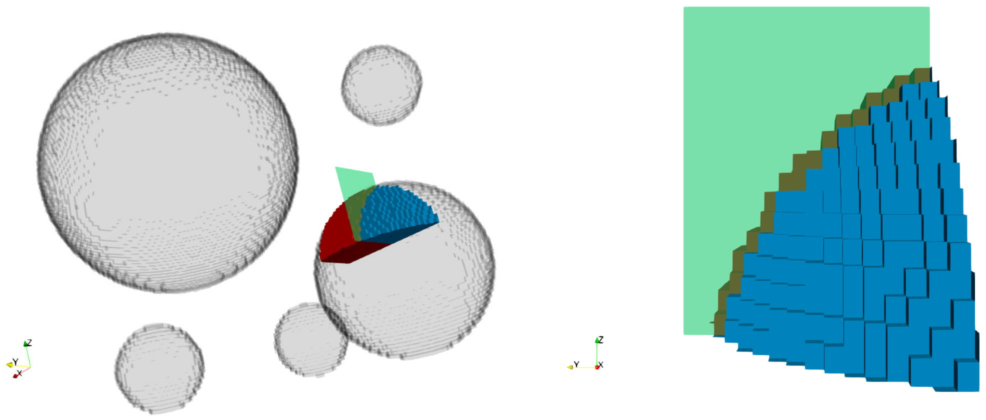


Figure 6.13: **Left:** Example visualization of the coefficient function in Fig. 6.12 for two neighboring subdomains, marked in red and blue, and the face between those subdomains, marked in green. **Right:** Zoom-in of the coefficient jump along the green face between two neighboring subdomains. Taken from [73].

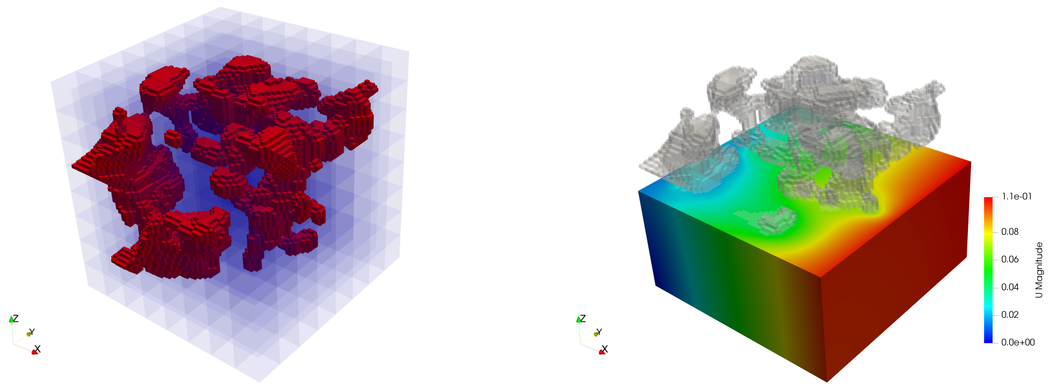


Figure 6.14: Coefficient distribution on a representative volume element (RVE). **Left:** Visualization of the domain decomposition into $8 \times 8 \times 8 = 512$ subdomains and $H/h = 16$. High coefficients are shown in red, and subdomains are shown in purple and by half-transparent slices. **Right:** Visualization of the corresponding linear elastic solution of the RVE. Based on data from [23]. Taken from [73].

a coefficient jump of $1e{+}3$, since this parameters are most representative for a real dual-phase steel. Let us remark that for large deformations steel shows a plastic behavior and therefore a linear elastic material model is not sufficient anymore. Considering the results in Table 6.12, the frugal coarse space FR4 again shows the best performance and the iteration counts are acceptable in all cases, even though for the linear elasticity problem the condition number is also significantly large. This implies that the potential in terms of robustness of the frugal coarse space is also limited, however, it is clearly higher than for the classic approaches. In this case, additional coarse space enhancements are necessary, as, e.g., adaptive constraints obtained by solving certain localized eigenvalue problems as presented in Section 3.5.

6.5.5 Using an approximate coarse solver

Generally speaking, the exact solution of the coarse problem with a sparse direct solver becomes a scaling bottleneck in all domain decomposition methods. This observation, in principle, is valid regardless of the decision which specific coarse space is chosen, since the size of the coarse space grows at least linearly with the number of subdomains. As we have discussed extensively in Chapter 5, this bottleneck can be overcome in BDDC by approximating the coarse solve by, e.g., a recursive application of BDDC [167,168] or an application of an AMG method [39,101,103]. We have implemented both approaches in our parallel BDDC software framework.

Here, we extend the results presented in Section 5.5 and provide numerical results using AMG for an approximate solution of the frugal coarse problem of BDDC. See Section 5.4

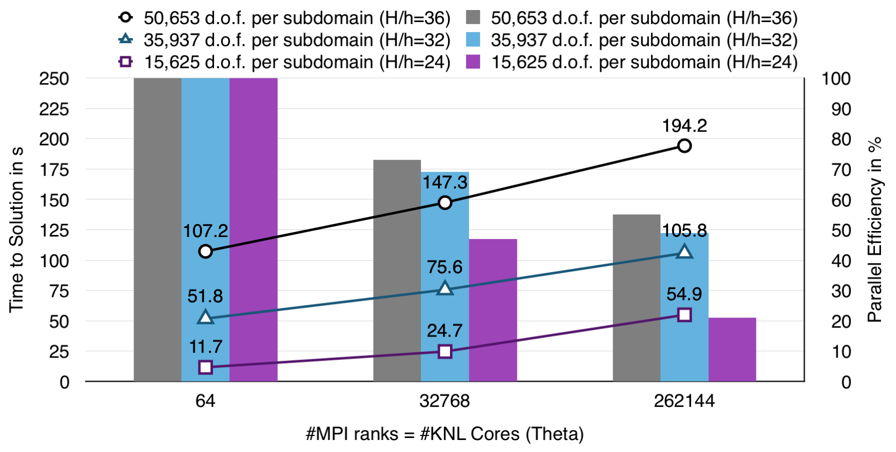


Figure 6.15: Weak scalability of BDDC in three dimensions using BoomerAMG to solve the FR4 coarse problem approximatively. Diffusion problem with high coefficient of $\rho = 1e6$ inside shifted channels and $\rho = 1$ in the remaining domain; see Fig. 6.5 for the coefficient distribution. Computed on Theta at Argonne National Laboratory, USA. Taken from [73].

for more details on our implementation of AMG as an approximate coarse solver for BDDC. In Fig. 6.15, we provide the results of a weak scaling experiment for a stationary diffusion problem. The experiment on Theta presented in Fig. 6.15 has been carried out by Martin Lanser using our parallel implementation since the author of this thesis did not have access to that machine. As already in Section 5.5.1, we use BoomerAMG [80] from the HYPRE package with an aggressive HMIS coarsening [35] and *ext+i* long range interpolation [34, 172]. As a coefficient distribution, we again choose the shifted channels (see Fig. 6.5) and a coefficient contrast of $1e6$. For all tested setups in Fig. 6.15, the number of CG iterations only varies between 18 and 22. In particular, this is also valid for the discretization using 262 144 subdomains on 262 144 cores with a total problem size of more than 12 billion degrees of freedom ($H/h = 36$). Therefore, we can conclude that the frugal coarse space FR4 is combinable with an approximate AMG solve without loosing robustness - at least for the considered coefficient distribution. For larger subdomain sizes (not shown in the figure), the scalability is still satisfying with more than 55% parallel efficiency scaling from one KNL node to 4 096 nodes.

Finally, let us note that it is also possible to combine the frugal coarse space with the three-level BDDC preconditioner presented in Section 5.2.1 as an approximate coarse solver. In particular, it is possible to construct frugal constraints both on the level of the subdomains and the subregions, as well as to combine frugal constraints on the second level with adaptive constraints on the third level, thus leading to smaller eigenvalue problems which have to be set up and solved on the coarsest level. Preliminary numerical results for both approaches exist and look promising. A detailed numerical investigation with respect to robustness and parallel scalability is still ongoing work and a topic of future research. This will be further elaborated in Chapter 9.

	stationary diffusion				**linear elasticity**			
	coefficient jump $1e+3$; $H/h = 16$							
coarse space	# c.	cond	it	TtS	# c.	cond	it	TtS
FR4	1 237	7.42e+0	19	1.7s	6 687	3.05e+1	44	19.9s
f	1 237	1.04e+2	45	2.7s	3 711	1.60e+3	259	55.1s
f + r	-	-	-	-	6 687	9.76e+2	144	38.3s
e	1 141	5.00e+3	135	7.3s	3 423	7.00e+2	212	48.7s
	coefficient jump $1e+3$; $H/h = 24$							
coarse space	# c.	cond	it	TtS	# c.	cond	it	TtS
FR4	1 237	8.73e+0	22	8.3s	6 687	4.77e+1	54	100.5s
f	1 237	3.83e+1	41	9.7s	3 711	1.68e+3	269	264.2s
f + r	-	-	-	-	6 687	2.07e+2	114	143.9s
e	1 141	1.08e+4	194	36.1s	3 423	9.17e+2	238	245.9s
	coefficient jump $1e+6$; $H/h = 16$							
coarse space	# c.	cond	it	TtS	# c.	cond	it	TtS
FR4	1 237	7.51e+0	19	1.7s	6 687	3.22e+1	47	20.7s
f	1 237	1.02e+5	189	11.3s	3 711	1.46e+6	>1000	>204.8s
f + r	-	-	-	-	6 687	5.40e+5	>1000	>222.0s
e	1 141	4.97e+6	283	14.6s	3 423	6.87e+5	>1000	>210.6s
	coefficient jump $1e+6$; $H/h = 24$							
coarse space	# c.	cond	it	TtS	# c.	cond	it	TtS
FR4	1 237	8.84e+0	21	6.7s	6 687	5.14e+1	57	103.1s
f	1 237	3.54e+4	195	36.4s	3 711	1.36e+6	>1000	>889.9s
f + r	-	-	-	-	6 687	1.01e+5	>1000	>915.5s
e	1 141	1.07e+7	434	78.5s	3 423	8.78e+5	>1000	>900.4s

Table 6.11: Condition numbers, iteration numbers, coarse space dimensions, and time to solution for frugal and classic BDDC coarse spaces for the coefficient distribution with five spherical inclusions of different size; see Fig. 6.12; Resolution of $128 \times 128 \times 96$ voxel ($H/h = 16$) or $192 \times 192 \times 144$ voxel ($H/h = 24$); Each voxel is discretized with six finite elements; **stationary diffusion:** coefficients of $\rho = 1e3$ or $\rho = 1e6$ inside the spheres and $\rho = 1$ in the remaining matrix material; **linear elasticity:** coefficients of $E = 210\,000$ or $210\,000\,000$ in the spheres and $E = 210$ in the remaining matrix material; $\nu = 0.3$ everywhere. Decomposition into 384 subdomains; computed on 192 cores. Using **ρ-scaling**. The acronym **v** stands for vertex constraints, **e** for weighted edge translations, **f** for weighted face translations, and **r** for weighted edge or face rotations; **TtS** abbreviates Time to Solution and **# c.** the size of the coarse space. Table already published in [73, Table 5.1].

	stationary diffusion				**linear elasticity**			
	coefficient jump $1e+3$**;** $H/h=24$							
coarse space	# c.	cond	it	TtS	# c.	cond	it	TtS
FR4	1 687	5.17e+1	29	6.6s	9 093	1.67e+2	76	123.8s
f	1 687	2.52e+2	94	14.2s	5 061	1.19e+3	274	275.6s
f + r	-	-	-	-	9 093	5.09e+2	179	211.7s
	coefficient jump $1e+6$**;** $H/h=24$							
coarse space	# c.	cond	it	TtS	# c.	cond	it	TtS
FR4	1 687	7.88e+1	28	6.5s	9 093	2.44e+4	179	210.9s
f	1 687	2.50e+5	910	123.9s	5 061	9.73e+5	>1000	>893.7s
f + r	-	-	-	-	9 093	4.70e+5	>1000	>924.9s

Table 6.12: Condition numbers, iteration numbers, coarse space dimensions, and time to solution for frugal and classic BDDC coarse spaces for the coefficient distribution obtained by an EBSD measurement of a dual-phase steel; see Fig. 6.14; Resolution of $192\times192\times192$ voxel ($H/h=24$); Each voxel is discretized with six finite elements; **stationary diffusion:** coefficients of $1e3$ or $1e6$ inside the inclusions and 1 in the remaining matrix material; **linear elasticity:** coefficients of $E=210\,000$ or $E=210\,000\,000$ in the inclusions and $E=210$ in the remaining matrix material; $\nu=0.3$ everywhere. Decomposition into 512 subdomains; computed on 256 cores. Using **ρ-scaling**. See Table 6.11 for the row and column labeling. Table already published in [73, Table 5.2].

7 A short overview of deep learning

Due to the development towards increasing amounts of data in various fields of research and in various sectors of the industry, using deep learning and other machine learning techniques has experienced increasing popularity during the last decades in order to automatically process these data and to automate processes related to the data; see, e.g., [138]. In general, machine learning algorithms process high amounts of data and are trained to recognize certain patterns and correlations in the data or, roughly speaking, to gain some sort of experience which can then be generalized to other, previously unseen data and situations. Common examples for the application of machine learning techniques are, e.g., image recognition, natural language processing or the identification of disease patterns in medical research, to name only a very small fraction of possible applications. One attempt to structure the wide field of machine learning (ML) is to subdivide it into supervised and unsupervised learning [169]. Additionally, the field of reinforcement learning is often regarded as a third and separate domain within machine learning; see, e.g., [16, 30].

In this thesis, we will make use of techniques from deep learning [62], an important subfield of supervised machine learning, to construct a FETI-DP coarse space which is robust for many heterogeneous coefficient distributions and, at the same time, preferably small. Since the aim of this thesis is to develop robust and efficient FETI-DP and BDDC coarse spaces rather than to give an exhaustive introduction into machine learning, we will concentrate on the description of the concrete machine learning tools which we use for our specific application. For the convenience of the reader, we will also give a short description of deep learning which is by no means exhaustive. For a more comprehensive introduction into deep learning, also with respect to different network architectures, we refer to, e.g., [30, 62, 66] and the references therein.

In the following, we give a short introduction of supervised machine learning in Section 7.1 and continue with a description of dense feedforward neural networks in Section 7.2. In Section 7.3, we briefly comment on some theoretical and numerical details with respect to the training and evaluation procedure of feedforward neural networks.

7.1 Supervised learning as a high-dimensional nonlinear optimization problem

As already mentioned, the wide field of machine learning can roughly be subdivided into the subfields of supervised, unsupervised, and reinforcement learning. For all three types of machine learning models the collection and processing of data is one of the most fundamental tasks. In supervised learning, other than for unsupervised learning, the data usually consist of a set of input data, which are called *features*, and a corresponding set of output data, which are usually referred to as *labels* [169]. In our application, which is the design of a robust and efficient FETI-DP and BDDC coarse space, we will train a carefully designed supervised machine learning model and thus, we will focus on the principles of supervised learning for the remainder of this chapter. The following description of a supervised learning model is based on [72] and [169] and the references mentioned therein.

From a high-level point of view, supervised machine learning models approximate (in general) nonlinear functions F, which associate input and output data:

$$F : I \rightarrow O. \tag{7.1}$$

Here, the input space I can be a product of $\mathbb{R}$, $\mathbb{N}$, and Boolean vector spaces. The output space O is typically either an $\mathbb{R}$ vector space or an $\mathbb{N}$ vector space. Depending on the general numeric type of the output space, the supervised learning problem (7.1) is either a *regression* or a *classification* problem, i.e., the output space O is an $\mathbb{R}$ vector space for regression problems or an $\mathbb{N}$ vector space for classification problems.

In order to compute a machine learning model which approximates the function F in (7.1), a large set of a priori known data $\{i_p, o_p\}_{p=1}^{P}$ is necessary, with $\{i_p\}_{p=1}^{P} \in I$ and $\{o_p\}_{p=1}^{P} \in O$. These data are then used to determine proper parameters which define the machine learning model. The iterative process of determining appropriate parameters is referred to as the *training* of a model and is based on a set of techniques known from mathematical and numerical optimization; cf. also Section 7.3. The amount of data used for the training of a model is typically partitioned into two separate sets of batches of examples which are usually referred to as *training* and *validation data*. In the training and optimization phase, the model is trained to fit the training data. Intuitively, the larger and more diverse the training set, the better is the performance of the trained machine learning model since the model has gained more experience due to a wider breadth of examples [169]. Additionally, the validation data are used to control the generalization properties of the model, i.e., to ensure that the model is not fitted too closely to the training data but is also able to accurately predict the output for new, previously unseen input data; cf., e.g., [169, Sec. 6.4] and [135, pp. 25–29]. In that way, over- or underfitting of the machine learning model should be minimized.

As already mentioned, the aim of training a supervised machine learning model is to train a hypothesis function $h : I \rightarrow O$ which approximates the function F in (7.1) as good as possible. In the machine learning context this task is usually formulated by defining a *cost* or *loss function*

$$l : I \times O \rightarrow \mathbb{R} \tag{7.2}$$

which returns a score how well a given learning task is accomplished with respect to the training and validation data for a specific choice of the model parameters. In particular, a high value of the loss function indicates a choice of the model parameters that results in a poor performance of the learning task, whereas a low value corresponds to a set of parameters that results in a good performance; see [169]. Therefore, we are interested in finding a set of model parameters which provides the smallest value, i.e., a *minimum* of the loss function $l(\cdot)$ in (7.2). For a more mathematical description, we recall that we denote by $\{o_p\}_{p=1}^P$ the output data for given input data $\{i_p\}_{p=1}^P$. We further denote by $\{\widehat{o}_p\}_{p=1}^P$ the approximated output data which we obtain by evaluating the hypothesis function h for given input data $\{i_p\}_{p=1}^P$, i.e., we have $h(i_p) = \widehat{o}_p$ for $p = 1, \dots, P$. Thus, the training of a supervised machine learning model can be written as the following nonlinear high-dimensional optimization problem

$$\min_{\theta \in \mathbb{R}^n} l(h_\theta | o), \tag{7.3}$$

where $\theta \in \mathbb{R}^n$ denotes the model parameters of the hypothesis function h_θ. The specific form or definition of the hypothesis function h_θ depends strongly on the supervised machine learning task, i.e., classification or regression, as well as on the chosen machine learning model which can be, e.g., a linear model or a neural network; see also Section 7.2. The loss function $l(\cdot)$ depends on the given learning problem and also on the chosen optimization method. Common choices for the loss function $l(\cdot)$ are, e.g., the mean squared error (MSE) function

$$l(h_\theta | o) = \frac{1}{P} \sum_{p=1}^{P} (h_\theta(i_p) - o_p)^2 = \frac{1}{P} \sum_{p=1}^{P} (\widehat{o}_p - o_p)^2 \tag{7.4}$$

and the cross-entropy softmax function

$$\begin{aligned} l(h_\theta | o) &= -\frac{1}{P} \sum_k \sum_{p=1}^{P} \Big[o_{p,k} \log(h_{\theta,k}(i_p)) + (1 - o_{p,k}) \log(1 - h_{\theta,k}(i_p)) \Big] \\ &= -\frac{1}{P} \sum_k \sum_{p=1}^{P} \Big[o_{p,k} \log(\widehat{o}_{p,k}) + (1 - o_{p,k}) \log(1 - \widehat{o}_{p,k}) \Big] \end{aligned} \tag{7.5}$$

where the index k denotes the class indices in a multiclass classification problem; see, e.g., [169, Sect. 6.3]. For the minimization of the loss function, concepts from mathematical and numerical optimization are used.

Let us note, that while the training of machine learning models can be computationally very expensive, the evaluation of the model is typically cheap. In particular, the training of a machine learning model corresponds to a nonlinear high-dimensional optimization problem. However, the training can be performed a priori in an offline phase and the resulting model is then saved for online usage, i.e., it can be evaluated with relatively low computational effort for new input data to predict the unknown output for these data.

7.2 Neural networks for the solution of nonlinear classification problems

In this thesis, we propose a hybrid approach, which uses feedforward neural networks for the classification of critical edges or faces of a domain decomposition, where the

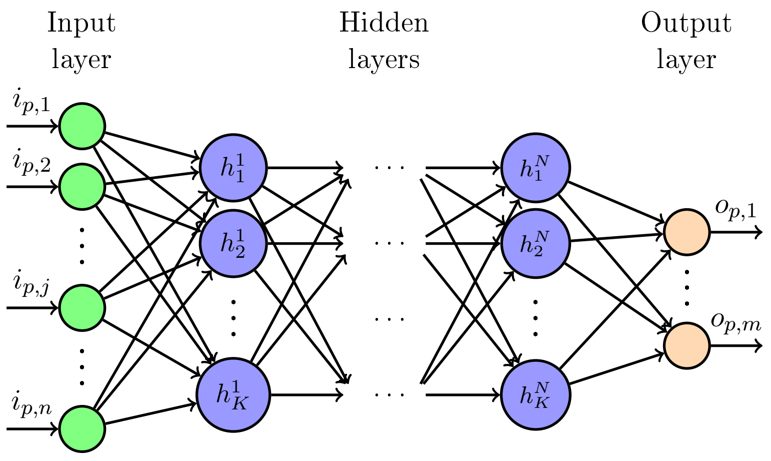


Figure 7.1: Structure of a dense feedforward neural network with n input nodes (marked in green), m output nodes (marked in orange), and N hidden layers with K neurons per layer (marked in blue). Figure in modified form in [72].

computation of additional adaptive constraints is necessary to obtain a robust coarse space; cf. also Section 3.5. This approach can be classified in the new, recently rapidly developing field of scientific machine learning in which existing techniques of scientific computing and machine learning are combined and further developed [7]. The details of the specific approach, which we denote by ML-FETI-DP, will be described in Chapter 8. Since our approach in Chapter 8 is based on deep learning, in this chapter, we give a mathematical introduction of dense feedforward neural networks or, more precisely, multilayer perceptrons; see, e.g., [62, Chapt. 4], [135, pp. 104–119], and [169, Sec. 5.1.4]. Parts of this section have already been published in slightly modified form in [72].

On an abstract level, the concept of neural networks is inspired by a simplification of neurons in the human brain; see, e.g., [16,169]. Here, we will present a rather mathematical presentation of neural networks which interprets a neural network as a graph, which defines a high-dimensional, concatenated nonlinear function.

A possible approach to define a feedforward neural network is to interpret it as a directed, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$, and a weight function $w : \mathcal{E} \to \mathbb{R}$; see, e.g., [155, Chapt. 20.1]. An exemplary visualization of the graph of a dense feedforward neural network is presented in Fig. 7.1. The neural network is assumed to be organized in *layers*, i.e., the set of nodes $\mathcal{V}$ can be represented as the union of nonempty, disjoint subsets $\mathcal{V}_i \subset \mathcal{V}, i = 0, \ldots, N+1$. These sets are defined such that for each edge $e \in \mathcal{E}$ there exists an $i \in \{0, \ldots, N\}$ with e being an edge between a node in $\mathcal{V}_i$ and one in $\mathcal{V}_{i+1}$; see [155, Chapt. 20.1]. In general, different layers can perform different transformations on their input. A neural network usually starts with an input layer (marked in green in Fig. 7.1), where the features $\{i_p\}_{p=1}^P$ of the external data are used as input data, and concludes with an output layer (marked in orange in Fig. 7.1). Depending on the supervised machine learning task (i.e., classification or regression), the output values can be interpreted, e.g., as the probability distribution of the different classes of a multiclass classification problem. The layers in between the input and the output layer are called *hidden layers*. The nodes in a neural network are called *neurons* and, in dense feedforward neural networks, each neuron (in a chosen layer) is influenced by all neurons from the previous layer.

Mathematically, the relation between two consecutive layers is the conjunction of a linear mapping defined by the weight function w and a nonlinear activation function. In particular, the nonlinearity of the activation function enables a neural network to approximate highly complex relations between the input and output data.

Recently, many different choices for the activation function α have been developed, which are often modifications and combinations of relatively simple mathematical functions. For the numerical experiments in Chapter 8, we choose the Rectified Linear Unit (ReLU) function [60, 86, 137] which is defined by

$$\alpha(x) = \max\{0, x\}.$$

This function is almost linear and its evaluation is computationally cheap. However, its nonlinearity is sufficient for the approximation of many nonlinear relations. Other common choices for the activation function α are, for instance, the hyperbolic tangent function (tanh) or further modifications of the ReLU function as, e.g., the Gaussian Error Linear Unit (GELU) [79], the Exponential Linear Unit (ELU) [32], and the Scaled Exponential Linear Unit (SELU) function [91]. The optimal choice of the activation function strongly depends on the given machine learning problem, the network structure as well as on the optimization method used for the training of the model; see [169, Sec. 13.3]. In our case, using the ReLU function as activation function α, the output of the k-th layer of the neural network can be written as

$$y = \alpha^k(x, W^k, b^k) = \max\left\{0, (W^k)^T x + b^k\right\},$$

where $W^k = (w_{ij}^k)_{i,j}$ and b^k are the weight matrix and the bias vector, respectively. Note that an entry w_{ij}^k of W^k corresponds to the value of the weight function w associated with the corresponding edge between layer $\mathcal{V}_{k-1}$ and $\mathcal{V}_k$. Then, the application of a complete neural network with N hidden layers to an input vector $i \in I$ is given by

$$\begin{aligned} h^1 &= \alpha^1(i, W^1, b^1), \\ h^{k+1} &= \alpha^{k+1}(h^k, W^{k+1}, b^{k+1}), \quad 1 \le k < N, \\ o &= (W^{N+1})^T h^N + b^{N+1} \end{aligned} \tag{7.6}$$

where h^k is the output of the k-th hidden layer and $o \in O$ is the (final) output vector. The computation of the output vector o is performed without an additional application of the activation function. Since we use dense neural networks, all entries of the matrices w and W^k, $k = 1, ..., N$, are nonzero, but using a dropout rate, a certain number of randomly chosen entries can be set to zero. The weights $W^k = (w_{ij}^k)_{i,j}$ and bias vectors b^k in (7.6) are optimized in an iterative process, i.e., during the training phase of the model (see Section 7.1) such that the relation (7.1) between the input and the output data is approximated as accurately as possible and the value of the loss function (7.2) is minimized. Typically, the loss functions associated with dense neural network models are highly nonconvex which results in certain difficulties when searching for the global minimum of the loss function in (7.3); see also [169]. This issue will be further addressed in Section 7.3.

Finally, let us note that for our software framework in Chapter 8, we employ machine learning and data analysis implementations from the software libraries Tensorflow [1] and Scikit-learn [145].

7.3 Some remarks on the training procedure of neural networks

Due to the complexity of neural networks and their ability to approximate complicated nonlinear functions, the loss functions of multi-layer neural networks are usually high-dimensional and nonconvex [122, 169]. In particular, the loss surface of the respective loss function often contains long or wide valleys, many flat areas as well as saddle point problems and local minima; see [169, Sec. 13.5]. Additionally, the specific form of the loss surface depends strongly on the chosen network architecture, the initialization of the model parameters and the used mathematical optimiziation method (e.g., batch-based or full batch) [122]. An exemplary visualization of the loss surface for a residual network (ResNet) [67] with 56 layers without skip connections is shown in Fig. 7.2. The shown loss surface in Fig. 7.2 is generated using the proposed visualization techniques in [122] which are based on a projection approach and a filter-wise normalization to obtain a normalized, two-dimensional representation of the high-dimensional loss function of a neural network. Even though we have to consider that the surface in Fig. 7.2 is only a projection of the high-dimensional loss function, we can still observe the previously described difficulties of local minima and saddle points which complicate the training procedure of neural networks.

In general, numerical optimization methods of first- and second-order schemes are used for the optimization, i.e., the minimization of the loss function of neural networks. Popular optimizers which are widely used for the training of neural networks are modified variants of a stochastic gradient descent (SGD) method [19,20,62,151]. In principle, SGD methods use a stochastic approximation of the gradient of a function which is computed from a randomly drawn subset of the data. This subset of data is usually referred to as a *batch* of examples in the machine learning context. Due to the difficulties explained above when minimizing the loss functions of neural networks, a wide range of modifications of the standard SGD method has been developed. These modifications include momentum-based and normalized gradient methods [169, Sec. 13.5]. Additionally, both modifications can be combined to further enhance gradient-based optimization methods such that they can minimize loss functions of deep neural networks more effectively. Popular examples of modified SGD methods which combine the ideas of a momentum-accelerated and a normalized gradient are the Adaptive Moment Estimation (ADAM) [90] and Root Mean Square Propagation (RSMProp) [82, 83] first-order methods. Usually, both methods are

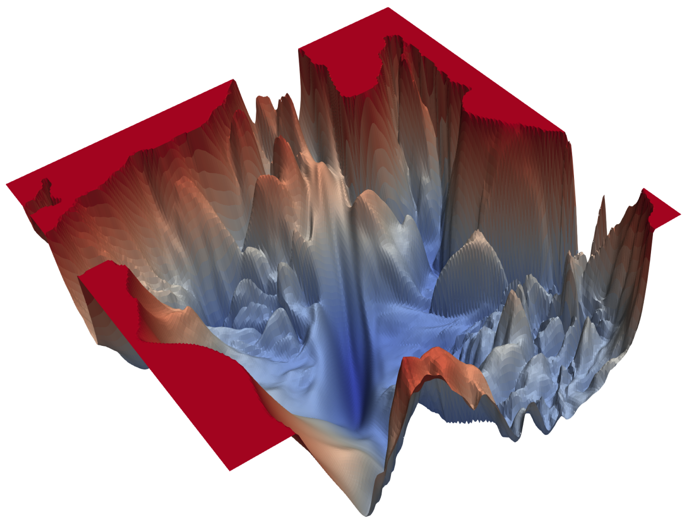


Figure 7.2: Exemplary visualization of the loss surface of a ResNet with 56 layers without skip connections using the projection techniques proposed in [122]. Blue areas correspond to low values of the loss function, whereas red areas correspond to high values.

implemented as a batch-based approach, i.e., using only a batch of examples to compute approximations of the gradient of the loss function instead of using the entire set of data at once. Since the neural networks which we train in Chapter 8 are relatively shallow and thin, in our experiments, we use a batch-based approach of the ADAM optimizer for the training; cf. also Chapter 8.

For completeness, let us mention that also optimization methods based on second-order schemes as, e.g., stochastic variants of the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method [27,28,140] have drawn an increasing interest in recent years to accelerate the training of neural networks and obtain a better performance with respect to a given learning problem; see, e.g., [15, 18, 26, 171]. However, the relation between different optimization methods and specific network architectures or specific learning problems is still ongoing research and would be beyond the scope of this thesis.

8 Designing an efficient and robust adaptive FETI-DP coarse space using deep learning

In Chapter 6, we have observed that the new frugal coarse space which is a heuristic extension, i.e., a generalization of weighted averages along edges and/or faces, is able to provide a robust algorithm for a wider range of different heterogeneous coefficient distributions. However, for completely arbitrary coefficient distributions with numerous sharp jumps along and across the interface of the domain decomposition, the frugal coarse space can also deteriorate in convergence; cf. also the examples considered in Section 6.5.4. In this case, adaptive, i.e., problem-dependent coarse spaces are necessary to retain an algorithm for which we can again prove a condition number bound which is independent of the contrast of the coefficient function.

As explained in Section 3.5, most adaptive coarse spaces rely on the solution of certain localized eigenvalue problems on parts of the domain decomposition interface and use a selection of the corresponding eigenmodes to construct appropriate constraints. These constraints are integrated into the coarse space prior to the first iteration of the iterative solver. In general, the respective eigenvalue problems in the adaptive approaches are typically small and related only to a small number of neighboring subdomains; the concrete number depends strongly on the specific approach. For the adaptive FETI-DP coarse space which we have introduced in Section 3.5, we always consider eigenvalue problems on edges or faces between two neighboring subdomains. Hence, it is feasible to parallelize the setup and the solution of the different local eigenvalue problems and thus the computation of the adaptive constraints; see also, e.g., [96, 119]. Nevertheless, in a parallel implementation, a significant number of subsequent eigenvalue problems can occur on single compute cores. Thus, building the adaptive coarse space can make up the larger part of the total time to solution. On the other hand, as we have also seen in Section 6.4.4, for many realistic coefficient distributions, only a few adaptive or frugal constraints on a few edges or faces are necessary for a robust coarse space. This means that, practically, often many of the eigenvalue problems are indeed unnecessary to be solved. It is usually not known a priori which of the eigenvalue problems are necessary to obtain a robust coarse space. A heuristic approach to decide which of the eigenvalue problems can be omitted based on the coefficient jumps as well as on the residual after one step of the FETI-DP or BDDC method has been considered in [93, 94, 119]; see also [107] for first preliminary results in this direction. Additionally, a similar heuristic approach for the detection of necessary eigenvalue problems and the construction of adaptive constraints without the solution of eigenvalue problems for overlapping Schwarz methods is proposed in [68, 69].

Here, we propose a fundamentally different and more general approach. Our key idea is to train a neural network to make an automatic decision whether we have to solve a certain eigenvalue problem or not. In the remainder of this chapter, we will apply this idea to a specific adaptive FETI-DP coarse space as described in Section 3.5. Since the FETI-DP and the BDDC domain decomposition method are dual to each other, the described procedure can equally be applied to an adaptive BDDC method. In particular, our approach can - in principle - be generalized to any adaptive DD algorithm which relies on the solution of local eigenvalue problems. In [71], we have also shown that it is possible to adapt our approach to the selection of critical edges in an adaptive GDSW (Generalized Dryja–Smith–Widlund) method [70], i.e., an overlapping DD method. This will be not discussed in detail in this thesis.

To extensively test our approach for adaptive FETI-DP, we consider both, two- and three-dimensional model problems. This implies that we use neural networks both for the classification of critical edges (in the two-dimensional case) and critical faces (in the three-dimensional case). Let us recapitulate that our predominant goal throughout this thesis is to design a robust and efficient FETI-DP or BDDC coarse space. Thus, we aim at a coarse space which is on the one hand robust in the sense that it has a condition number which is independent of the contrast of the considered coefficient function, and on the other hand, is preferably small and can be computed cheaply. For this purpose, we generate an appropriate amount of training and validation data to train a neural network in an offline phase. In particular, the neural network is designed such that it makes a decision whether for a specific edge or face, the eigenvalue problem is necessary for robustness, based on local information of the coefficient distribution. The trained neural network is then saved with the obtained weights and hyper parameters. Then, in an online phase, we generate the respective input data for the neural network for a specific test problem by using again a carefully designed representation of the local coefficient function associated with an edge or a face, and evaluate the already trained network for the respective input data. As a result, we obtain an automated decision, whether for the respective edge or face, the eigenvalue problem needs to be solved or not. The basic tools for designing and training a dense feedforward neural network have already been presented in Chapter 7. In this chapter, we will extend this rather general description by focusing on the concrete adaptation of neural networks to our specific application which is the detection of critical edges or faces to construct a robust and efficient coarse space.

The remainder of this chapter is basically subdivided into two parts. In the first part, in Section 8.1, we present the theoretical considerations to design a neural network which is specifically tailored to our application. Here, we explicitly focus on the design and generation of the input and the output data which are necessary to train the neural network in a preprocessing phase. In the second part of this chapter, in Sections 8.2 and 8.3, we will present a range of different numerical experiments using the proposed approach for both two- and three-dimensional stationary diffusion and linear elasticity problems. We will explicity use test problems which are highly heterogeneous and which are not included in the training and validation data. Since the proposed approach makes use of supervised machine learning techniques as presented in Chapter 7 and will be applied to the adaptive FETI-DP method introduced in Section 3.5, we denote our new approach by *ML-FETI-DP*. The proposed method is a hybrid approach since it combines the principles of deep learning with adaptive domain decomposition methods. The idea of using machine learning to improve numerical algorithms (and vice versa) is part of the rapidly developing field of scientific machine learning [7] which has drawn increasing attention in recent years; cf. also Chapter 7.

Let us note that we explicitly split the theoretical description of our techniques as well as the presentation of the numerical experiments into separate subsections for two-dimensional and for three-dimensional problems. In particular, the generation of sufficient training and validation data for the network is more complex in three dimensions than for two dimensions which is why, for three dimensions, additional effort and explanations are necessary. This will be further discussed in Section 8.1.4.

Parts of this chapter have already been published in modified or unmodified form in [72, 74–77].

8.1 ML-FETI-DP – FETI-DP with an adaptive coarse space based on deep learning

As described in the beginning of this chapter, we propose to train a feedforward neural network to detect edges (in two spatial dimensions) or faces (in three spatial dimensions) for which the solution of the eigenvalue problem presented in Section 3.5.1 is actually necessary to ensure a robust convergence behavior of the FETI-DP method. We will denote these edges or faces, respectively, by *critical* edges or faces for the remainder of this chapter. Edges and faces where the eigenvalue problem is not needed for robustness are denoted as *uncritical*.

8.1.1 The idea of using a neural network to detect critical edges or faces

Let us now describe in more detail how we design the preprocessing step in ML-FETI-DP to identify critical edges and/or faces using a classification neural network. Parts of this section have already been published in modified or unmodified form in [72, 75].

Prior to the first iteration of the iterative solver, we aim, for each edge or face of the domain decomposition, to obtain a classification whether the local eigenvalue problem should be set up and solved, or not. Subsequently, we only compute the local eigenvalue problems on edges or faces which are classified as critical by the neural network. On all uncritical edges or faces, we do not set up nor solve the respective eigenvalue problem and do not enforce any constraints. Consequently, depending on the number of edges and faces which are categorized as uncritical by the neural network, a potentially high number of eigenvalue problems can be avoided to be computed and the computational effort of the adaptive DD method can be reduced.

As we have seen in Section 7.2, a fundamental task when defining a neural network to approximate the solution of a specific classification problem is the design of appropriate input and output data for the network. In our application within the context of adaptive

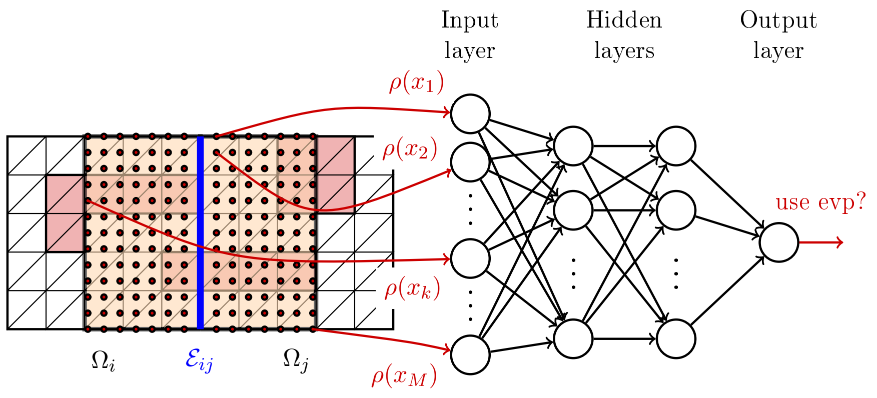


Figure 8.1: Sampling of the coefficient function in two dimensions; white color corresponds to a low coefficient and red color to a high coefficient. In this representation, the sampling points are used as input data for a neural network with two hidden layers. Here, only sampling points from slabs around the edge are chosen (marked in orange). Figure in modified form in [77].

coarse spaces, we know from our experience that, in general, the distribution of the coefficient function in the neighborhood of an edge or a face is relevant for the decision whether adaptive constraints are necessary for the respective equivalence class. Thus, as input for our neural network, we use samples, i.e., function evaluations of the coefficient function within the two subdomains adjacent to an edge (in two dimensions) or a face (in three dimensions); cf. Fig. 8.1 for an exemplary visualization of computed sampling points around an edge $\mathcal{E}_{ij}$. As output for the neural network, we save the information whether at least one adaptive coarse constraint has to be computed on the corresponding edge (or face) or not. Thus, we so far obtain a two-class classification problem where the neural network is trained to distinguish between critical and uncritical edges or faces. Additionally, this two-class classification approach can be extended by a third class for which we impose the frugal constraint as introduced in Chapter 6. This will be further discussed in Section 8.1.2.

In general, there are some common considerations for the generation of the input data for the neural network which are valid for both the two- and the three-dimensional case. Let us clarify that we need to train (at least) two separate neural networks, i.e., (at least) one network which can be applied to two-dimensional problems and (at least) one network which can be applied to three-dimensional problems. Especially, as already mentioned, the generation of the training data is more complex in three dimensions than for two dimensions. Before we describe the concrete generation of training data for both cases in Sections 8.1.3 and 8.1.4, let us briefly comment on the similarities between them. In order to design our machine learning approach as efficient as possible, we aim to train only one neural network for each case with a fixed number of input nodes which can be evaluated for a range of different coefficient distributions and mesh discretizations. Therefore, we have carefully designed a technique to generate the input for the network

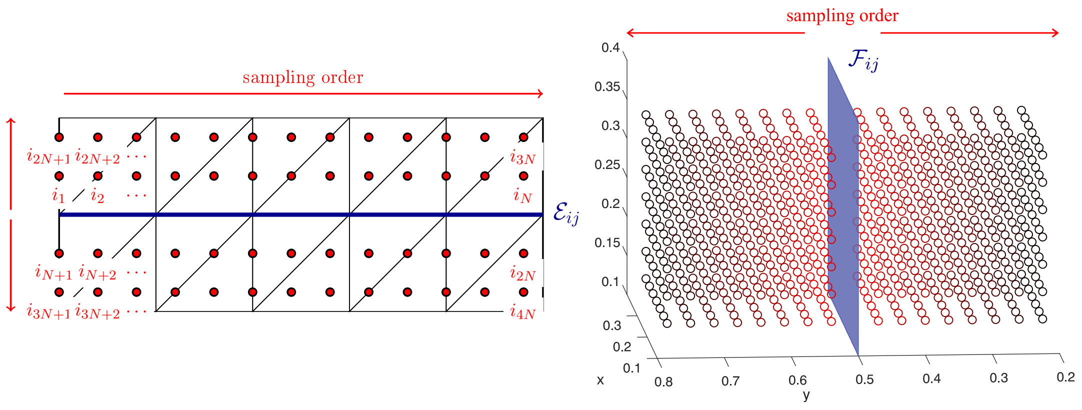


Figure 8.2: **Left:** Visualization of the ordering of the sampling points in two dimensions (marked in red) for a straight edge (marked in blue). **Right:** Visualization of the computed sampling points in three dimensions (marked in red) for a regular face (marked in blue) between two neighboring subdomains. The different shades correspond to increasing distance of the sampling points to the face and therefore to a higher numbering of the sampling points. Figure in modified form in [75].

that is independent of the underlying finite element discretization and ensures a consistent ordering of the input data. In our context, we refer to this approach as *sampling procedure* and to the computed function evaluations of the underlying coefficient function as *sampling points*. For both the two-dimensional as well as the three-dimensional case, we always use a fixed number of sampling points as input data for all mesh resolutions. In particular, we choose the sampling to be finer than all meshes used in our computations. As a rule of thumb, we assume that the sampling grid resolves all geometric details of the coefficient function. In Fig. 8.2, we show an exemplary visualization of the location and order of the sampling points for a two-dimensional (left) and a three-dimensional problem (right); note that other orderings are possible as well. However, for a satisfactory performance of the neural network, it is essential that the ordering of the sampling points is consistent among all data. For all cases, the aim is to obtain an input vector for the neural network which is of fixed length; i.e., number of sampling points in the direction of the edge times number of sampling points orthogonal to the edge for the two-dimensional example visualized in Fig. 8.2 (left).

Additionally, we ensure that all input values are real numbers and use dummy values, i.e., values of -1, to encode computed sampling points which lay outside the two neighboring subdomains of an edge or a face, respectively. This will especially become relevant for the computation of input data for irregular domain decompositions, as, e.g., obtained by METIS [87]; cf. Sections 8.1.3 and 8.1.4. In addition to that, we scale all input values using a min-max-scaling before starting the training of the neural network. Thus, we obtain input values which range only between zero and one. This is beneficial for the training performance and the classification accuracy of a neural network.

8.1.2 An extended three-class classification approach using frugal constraints

In Chapter 6, we have presented a frugal coarse space which, in many cases, provides a robust FETI-DP or BDDC preconditioner also for heterogeneous coefficient functions or material distributions. In particular, our experiments presented in Sections 6.4.1 and 6.5.3 have shown that for edges or faces which require only adaptive constraints resulting from the first eigenmode of the respective eigenvalue problem, the constraint can successfully be replaced by the manually constructed frugal constraint. Consequently, if known a priori, it is not necessary to solve any eigenvalue problems on these edges or faces. Based on this observation, we also propose an extended approach of ML-FETI-DP, which uses a three-class classification. For this purpose, we train an adapted neural network which - for stationary diffusion problems - distinguishes between the following three classes: edges (or faces), where the eigenvalue problem is unnecessary (class 0), edges (or faces) where the eigenvalue problem results in exactly one additional adaptive constraint (class 1), and those where the eigenvalue problem selects more than one constraint (class 2). For linear elasticity problems, the three classes are defined analogously except that class 1 contains edges (or faces) where the eigenvalue problem results in no more than three (in two dimensions) or six (in three dimensions) constraints, respectively, and the definition of class 2 is adjusted accordingly. For all named cases, we do not enforce any constraints on an edge or a face assigned to class 0. If an edge or a face is assigned to class 1, we enforce the frugal constraints on the respective edge or face as defined in Section 6.2 for the case of a stationary diffusion problem or as defined in Section 6.3 for the case of a linear elasticity problem. For edges and faces assigned to class 2, we set up and solve the local eigenvalue problem as defined in (3.26) and subsequently enforce the computed adaptive constraints.

Let us mention that for the numerical experiments presented in Sections 8.2 and 8.3, we use different thresholds $\tau \in (0, 1)$ for the decision boundary between the two or three classes of the classification obtained by the neural network. For the case of two classes, the interpretation of the value of τ is straightforward. If the obtained probability for class 1 ist lower or equal to the threshold value τ, the respective edge or face will be assigned to class 0, otherwise to class 1. For the three-class classification, we use a modified approach which works as follows. We first apply the standard multi-class classification rule

$$\arg \max_{c=0,1,2} \{P(p \in \omega_c)\}$$

for a classification problem with more than two classes; cf. [169, p. 100]. Basically, this implies that we assign the data point p to the class of class 0, 1, or 2, for which we obtain the highest probability value as output of the neural network. Then, if p is assigned to any of {class 1, class 2}, we rescale the probabilities for the final classification in class 1 or 2 using the rule:

$$\arg\max \left\{ \frac{P(p \in \omega_1)}{(1-\tau)(P(p \in \omega_1) + P(p \in \omega_2))}, \frac{P(p \in \omega_2)}{\tau(P(p \in \omega_1) + P(p \in \omega_2))} \right\}.$$

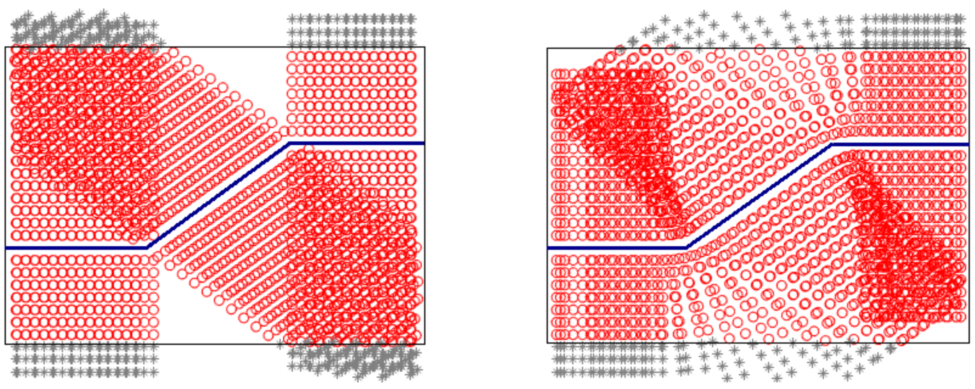


Figure 8.3: Sampling points for an irregular edge without smoothing the edge a priori (left) and using the smoothing strategy shown in Fig. 8.4 (right). Figure in modified form in [72].

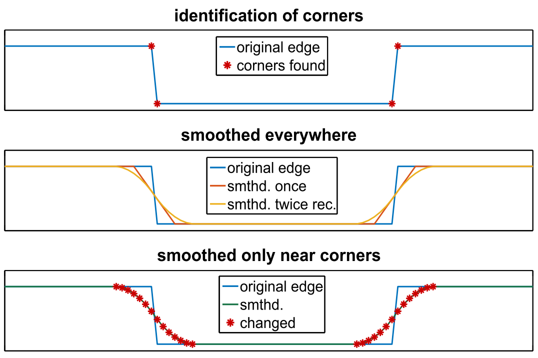
Consequently, a threshold of $\tau = 0.5$ results in an equal scaling between class 1 and class 2, whereas, e.g., a threshold of $\tau = 0.4$ results in more edges assigned to class 2. Let us remark that a proficient choice of the decision threshold τ can improve the robustness of our approach. In particular, the specific value of τ can be chosen by using the Receiver Operating Characteristics (ROC) curve as well as the precision-recall graph of the training and validation data; cf. also [71]. Please refer to [135, Sec. 5] for a definition of a precision-recall graph and a ROC curve. The specific choice of the threshold τ will be further motivated and discussed in Section 8.1.5.

8.1.3 Generation of training and validation data in two dimensions

In this section, we describe the specific procedure to generate a sufficient and appropriate amount of training and validation data to apply our ML-FETI-DP algorithm to two-dimensional model problems, i.e., stationary diffusion or linear elasticity problems in two dimensions. Parts of this section have already been published in modified or unmodified form in [72].

8.1.3.1 Sampling procedure for regular and irregular edges

As we have described in the beginning of this section, our goal is to provide a sampling procedure which always uses a fixed number of sampling points and, at the same time, ensures a consistent order of the sampling points for all mesh resolutions as well as for regular and irregular domain decompositions. In Fig. 8.1 and Fig. 8.2 (left), we show an exemplary visualization for the location and the order of the sampling points in two dimensions in the surrounding of an edge $\mathcal{E}_{ij}$. By using the specific ordering of the sampling points as visualized in Fig. 8.2 (left), we ensure that the input vector of the neural network is of fixed length, i.e., number of sampling points in direction of the edge times number of sampling points orthogonal to the edge, as well as a consistent ordering of the



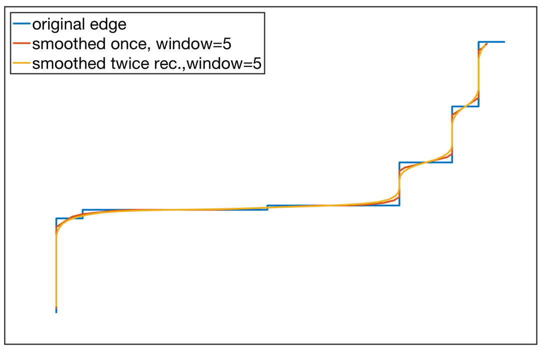


Figure 8.4: Exemplary visualization of the smoothing procedure for an irregular edge prior to the computation of sampling points. **Left:** Smoothing a jagged edge by identifying the kinks and using a moving average close to the kinks of the edge. **Right:** Smoothing of an example for an irregular edge. Figure in modified form in [72].

sampling points among all data. As we can observe from Fig. 8.2 (left), by construction, our sampling grid is oriented to the tangential and orthogonal direction of an edge. Thus, our sampling strategy is not restricted to the case of square subdomains, as indicated in Fig. 8.1, but can also be extended to more general subdomain geometries. In this case, we make sure to only use sampling points within the two subdomains adjacent to the considered edge in order to reflect the structure of the edge eigenvalue problems; cf. Fig. 8.3 where computed sampling points which lay outside the two neighboring subdomains are indicated as grey stars. For all sampling points which are outside the two subdomains, we use -1 as input data.

However, computing the sampling points strictly in the direction of the edge and in the directions orthogonal to the edge may lead to gaps within the sampling grid for non-smooth edges; cf. Fig. 8.3 (left). In particular, this results in gaps in the sampling grid in the close surroundings of an edge. Since, usually, especially the coefficient distribution in this area is highly relevant for the decision whether adaptive constraints are necessary, we aim to cover this area as carefully as possible with the computed sampling points. Therefore, we use a moving average to smooth out discontinuities in the tangential and orthogonal vectors of the edge before computing the respective sampling points. In particular, we use a fixed window length of five sampling points, and slide this window stepwise along the edge while computing the average of the subset of sampling points in each local window. As shown in Fig. 8.4, we smooth out kinks twice using a moving average recursively. Let us remark that, for our purpose, it is sufficient to consider a neighborhood of a kink instead of applying the moving average to the full edge. To identify all kinks of a given edge, we compute an approximation of the discrete second derivative for the entire edge. Subsequently, we use the smoothed version of the edge, i.e., the smoothed normal vectors, to compute the sampling points in both neigboring subdomains. In Fig. 8.3 (right), we show the resulting grid of sampling points

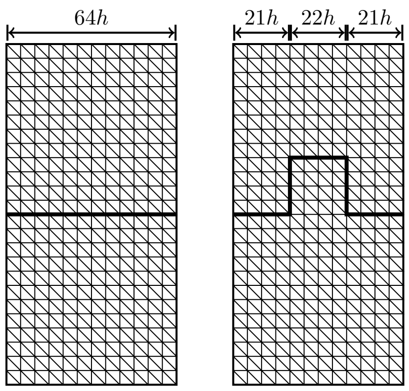


Figure 8.5: Geometric configurations used in the training data for two-dimensional problems: straight and jagged edges; subdomain size $H/h = 64$. Taken from [72].

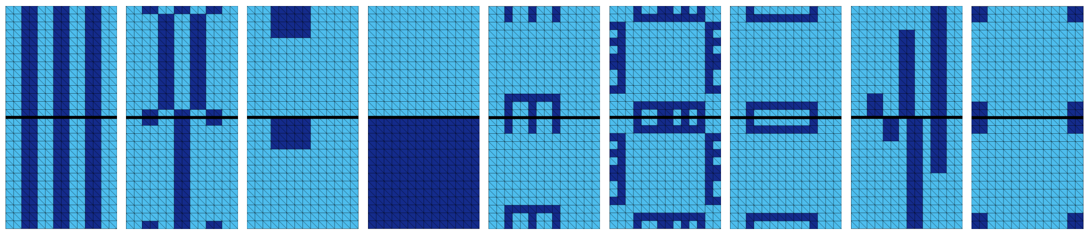


Figure 8.6: Nine different types of coefficient functions used for the training and validation of the neural network for two-dimensional problems. The inclusions, channels, boxes, and combs with high coefficient are displaced, modified in sized, and mirrored with respect to the edge in order to generate the complete training data set. We refer to the resulting data set as *smart data*. Taken from [72].

after the described smoothing procedure of the orthogonal vectors of the edge. As we can observe from Fig. 8.3, the smoothing procedure ensures that we cover at least the coefficient function very close to the edge which is decisive for the classification whether the eigenvalue problem is necessary for the respective edge or not.

For the training and validation of the neural network, we have generated and numerically tested different sets of training and validation data for two-dimensional problems. All in all, we have obtained the best results, i.e., the highest accuracy values for the training and validation data as well as the best generalization properties for a carefully selected data set which is inspired by the coefficient functions used in [69,108]. Since the included coefficient distributions are, to a certain extent, manually designed such that they cover a wide range of configurations which occur in practical model problems, we refer to this data set as *smart data*. Additionally, we have also generated different sets of randomized training data, to which we refer to as *random data*, as well as combinations of both. A more detailed comparison of the different sets of training data as well as a numerical investigation of their generalization properties with respect to different test

Hyper parameter	**Range tested by grid search**	**Optimal choice**
# hidden layers	{1, 2, 3, 4}	3
# neurons per layer	{10, 20, 30, 50}	30
dropout rate	{0, 0.2, 0.25, 0.5}	0.2
learning rate	{0, 0.001, 0.005, 0.01, 0.1, 1}	0.01
optimization algorithm	{Adam, AdaGrad}	Adam

Table 8.1: Hyper parameters of the neural network for two-dimensional problems and its training, and the optimal choice obtained by a grid search. Table already published in [72, Table 1].

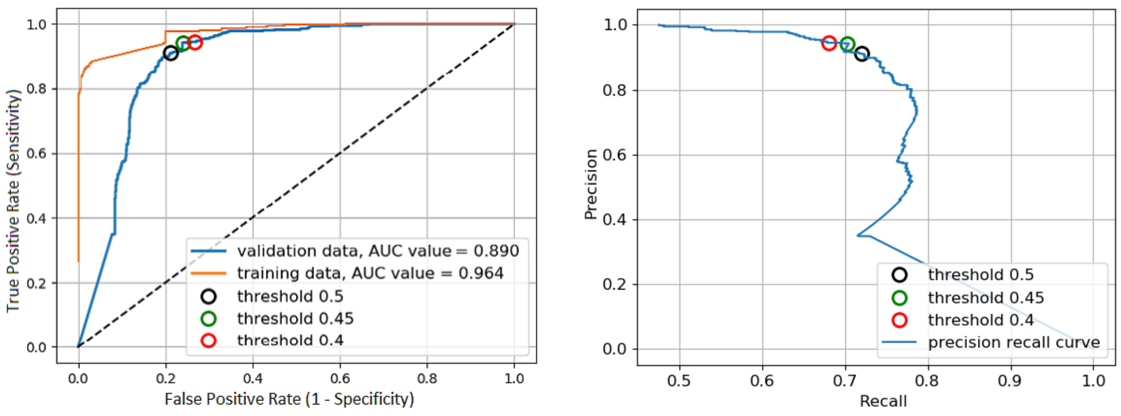


Figure 8.7: ROC curve and precision-recall plot for the optimal model for two-dimensional problems obtained by a grid search; cf. Table 8.1. We define precision as *true positives* divided by (*true positives*+*false positives*), and recall as *true positives* divided by (*true positives*+*false negatives*). The thresholds used in Section 8.2 are indicated as circles. Taken from [72].

problems will be presented in Section 8.2.4. Since the smart data, all in all, have shown the best performance properties for two-dimensional problems, we now continue with a detailed description of this data set and use it as an illustrating example to provide more details on the training of our specific neural network.

The training and validation data to which we refer to as smart data consist of 4 500 data configurations with varying coefficient functions and two different geometries of an edge $\mathcal{E}_{ij}$ which is shared by two subdomains Ω_i and Ω_j. To generate the corresponding output data that are necessary for the training of the neural network, we have to solve the eigenvalue problem described in Section 3.5 for each of the 4 500 training and validation configurations and save the classification, whether for a considered edge, at least one adaptive constraint has to be computed.

In a preliminary phase of our machine learning approach, we have observed that it is sufficient to train the network on two geometric configurations, i.e., two regular subdomains sharing a straight edge and two regular subdomains sharing a jagged edge (see Fig. 8.5), in order to generalize to arbitrary shapes of subdomains. For the sampling procedure in the generation of training and validation data, we select 127 points in the direction of the edge and 2×127 points in the orthogonal direction. Thus, we roughly obtain two sampling points in each finite element for the subdomain size defined by $H/h = 64$. We combine the two geometric configurations shown in Fig. 8.5 with coefficient functions of the types shown in Fig. 8.6. In order to obtain the full set of training data for the smart data, the inclusions, channels, boxes, and combs with high coefficient are varied in size, location, and orientation which actually leads to more configurations than the nine basic ones given in Fig. 8.6. Let us note again that in Section 8.2.4, we will provide comparative results for different types of training data sets for two-dimensional domains, i.e., the manually constructed training data presented here and training data sets obtained by a randomized coefficient distribution. For the remainder of this subsection and for the numerical results presented in Sections 8.2.2 and 8.2.3, however, we will use the training data set shown in Fig. 8.6 since it produces the best results in terms of the highest acccuracy values; cf. also Section 8.2.4.

During the iterative training of the neural network, we minimize the softmax cross-entropy loss function

$$g(b^1, ..., b^{N+1}, W^1, ..., W^{N+1}) = \sum_{c=1}^{C} \sum_{p \in \omega_c} \left(\log \left(\sum_{j=1}^{C} e^{o_{p,j}} \right) - o_{p,c} \right) \tag{8.1}$$

with respect to the weights and bias vectors of the neural network; cf. [169, Sect. 6.3] and Section 7.1. Here, C is the total number of classes in the classification problem, $o_{p,j}$ is the output corresponding to class j in the output vector of data p, and ω_c is the subset of the training data corresponding to class c. Thus, the softmax cross-entropy loss function minimizes the cross-entropy between the deterministic class labels of the training and validation data and the model's prediction for the same data. Minimizing the cross-entropy is equivalent to minimizing the Kullback-Leibler divergence, which is a measure for the difference of two probability distributions from information theory; cf., e.g., [62, Sec. 3.13]. Note that the prediction

$$P(p \in \omega_c) = \frac{e^{o_{p,c}}}{\sum\limits_{j=1}^{C} e^{o_{p,j}}} \tag{8.2}$$

of the output layer is the probability that the input with index p belongs to class c. Hence, using the softmax function as the activation function for the output layer of the neural network, the output values can be interpreted as a probability distribution of the different classes of our classification problem.

To find an approximative solution of this nonlinear optimization problem, we apply an SGD method with an adaptive scaling of the learning rate and a batch size of 100. For the adaptive scaling of the learning rates, we consider the AdaGrad (Adaptive Gradient) [47] and the Adam [90] algorithm.

In order to optimize the hyper parameters of the neural network, we apply a grid search algorithm on a discrete search space of parameters. Here, we use as hyper parameters the

classification type	threshold τ	fp	fn	acc
two-class classification	0.45	8.8%	1.9%	89.2%
	0.5	5.4%	5.1%	89.5%
three-class classification	0.4	5.1%	1.0%	93.9%
	0.5	3.2%	2.3%	94.5%

Table 8.2: Results on the complete smart training data set in two spatial dimensions. The numbers are averages over all 4 500 training configurations. We define the accuracy (acc) as the number of true positives and true negatives divided by the total number of training configurations. Table already published in [72, Table 2].

number of hidden layers, the number of neurons per layer, the dropout rate, the learning rate, and the type of the optimization algorithm. The corresponding hyper parameters, the search space, and the optimal choices for the parameters are given in Table 8.1. We compare and optimize the generalization properties of the neural network by cross-validation using a random splitting of our data set into 80 % training and 20 % validation data for each iteration of the grid search algorithm. The ROC curve and a precision-recall plot of the neural network with optimal hyper parameters are shown in Fig. 8.7. In both plots, the threshold τ for the decision boundary between critical and uncritical edges is varied between zero and one. When increasing τ, the false positive rate, which corresponds to the number of critical edges that are not detected by the algorithm, decreases. Consequently, the robustness of our ML-FETI-DP approach is improved. In Fig. 8.7, we additionally indicate the thresholds used in the numerical experiments in Section 8.2 as colored circles. More details on heuristic strategies for the choice of the ML threshold τ are given in Section 8.1.5.

8.1.3.2 Results on the training data in two dimensions

On the complete set of the smart training data in two dimensions, we obtain the results listed in Table 8.2. Here, we use the hyper parameters of the neural network and the training procedure which have performed best in Table 8.1. We observe a significantly better accuracy for the three-class classification compared to the two-class classification. Furthermore, we observe that a classification threshold of $\tau = 0.5$ yields the best accuracy for both types of classification. However, for the cases of irregular subdomains in Section 8.2, we will use a lower threshold τ to improve the robustness of the ML-FETI-DP approach. This will be further discussed in Section 8.2.

8.1.4 Generation of training and validation data in three dimensions

In this section, we describe the specific procedure to generate appropriate training and validation data to extend our ML-FETI-DP approach to three-dimensional problems, i.e., stationary diffusion or linear elasticity problems in three spatial dimensions. Parts of this section have already been published in modified or unmodified form in [75].

In analogy to Section 8.1.3, we aim to train and test our neural network for both, regular domain decompositions and for domain decompositions obtained from the graph partitioning software METIS [87]. We will observe that extending our methods introduced in Section 8.1.3 from two dimensions to three dimensions causes substantial challenges and additional effort is needed to preprocess the input data for our machine learning model. Consequently, we have to adapt and extend our sampling procedure, especially for irregular decompositions of three-dimensional domains. Let us note that the preprocessing of the input data is at the core of our hybrid ML-FETI-DP algorithm. Hence, the preprocessing of the three-dimensional input data is one of the main novelties compared to the two-dimensional case.

As described in Section 3.5, for adaptive FETI-DP in three spatial dimensions, local eigenvalue problems on edges as well as faces have to be solved to set up a robust coarse space and to obtain a sound theoretical condition number bound; cf also [92]. However, the respective edges typically only possess a relatively small number of nodes, and hence, the corresponding eigenvalue problems are rather small. Therefore, in our three-dimensional ML-FETI-DP approach, we restrict ourselves to the identification of necessary face eigenvalue problems and solve all eigenvalue problems for edges which belong to more than three subdomains. Note that, for unstructured domain decompositions, these edges are rather rare. In the following, we will thus describe the design of a neural network which is trained for the classification of critical faces in a three-dimensional domain.

Generally, the sampling should cover all elements in a neighborhood of the respective interface component. Therefore, in analogy to irregular edges in two dimensions, a smoothing procedure is necessary for irregular faces in three dimensions to prevent an incorrect or incomplete picture of the material distribution resulting from gaps in the sampling grid. Moreover, an additional challenge in the sampling procedure for irregular faces, such as faces obtained using METIS (in the following denoted by METIS faces), with an arbitrary orientation in the three-dimensional space, arises. In particular, a consistent ordering of the sampling points is neither a priori given nor obvious in most cases. More precisely, there is no natural ordering of a grid of points on an irregular face, such as going from the lower left corner to the upper right corner. A consistent ordering of the sampling points is, however, essential when using them as input data to train a neural network. In particular, given that neural networks rely on input data with a fixed structure, an important requirement of our data preprocessing is to provide samples of the coefficient distribution with a consistent spatial structure in relation to each face in our domain decomposition, even though the faces may vary in their location, orientation, and shape. For our generalized approach presented below, some of the computed sampling points may lay outside the two subdomains adjacent to a face. As already done for the two-dimensional case, we encode these points using a specific dummy value which clearly differs from all true coefficient values. Since all coefficient values are positive, we encode sampling points outside the adjacent subdomains by the value -1. This is essential to ensure that we always generate input data of a fixed length for the neural network for all mesh resolutions; see also Section 8.1.3.

8.1.4.1 Sampling procedure for regular faces

In the case of regular faces, the sampling procedure is fairly similar to the approach for straight edges in a two-dimensional domain decomposition; see Section 8.1.3. Basically, we compute a tensor product sampling grid by sampling in both, tangential directions of a face and in the directions orthogonal to the face. This results in a box-shaped structure of the sampling points in both neighboring subdomains of the face; see also Fig. 8.2 (right). A required consistent ordering of the sampling points for this case is naturally given by passing through the sampling points layer by layer with growing distance relative to the face. The ordering of the sampling points in layers parallel to the face is indicated by the changing color gradation in Fig. 8.2 (right).

8.1.4.2 Sampling procedure for METIS faces

Our sampling procedure for METIS faces consists of two essential steps. First, we construct a consistently ordered two-dimensional auxiliary grid on a planar projection of each face. Second, we extrude this auxiliary grid into the two adjacent subdomains of the face. The resulting three-dimensional sampling grid has both a fixed size and a consistent ordering for all faces. Sampling points which do not lie on the face or within the two adjacent subdomains are encoded using the dummy value -1.

First step – Construction of a consistently ordered auxiliary grid for METIS faces In order to construct the auxiliary grid for a METIS face, we first compute a projection of the original face represented in the three-dimensional Euclidean space onto an appropriate two-dimensional plane. In particular, we project a given METIS face onto a two-dimensional plane, such that we obtain a consistently sorted grid covering the face. This grid is induced by a tensor product grid on the two-dimensional projection plane. Note that since we use tetrahedral finite elements in three dimensions, each METIS face is naturally decomposed into triangles. Due to the projection from three dimensions to two dimensions, elements, i.e., triangles of the face, can be degraded or deformed, i.e., they can have a large aspect ratio. On top of that, we can also obtain flipped triangles; see Fig. 8.8 for an example where both cases occur. Hence, we have to regularize the two-dimensional projection of the face before constructing the sampling grid.

To obtain a well-shaped projection of the face which is appropriate for our purpose, we numerically solve an optimization problem with respect to the two-dimensional projection of the face. More precisely, the objective functions of the optimization problem are carefully designed such that flipped triangles (phase 1) as well as sharp-angled triangles (phase 2) are prevented:

$$\min_x \sum_{T_j} \lambda_1 \cdot e^{-\lambda_2 \cdot \det(T_j(x))} + \lambda_{\text{reg}} \cdot \|d(x)\|_2^2 \quad \textbf{(phase 1)} \tag{8.3}$$

and

$$\min_x \sum_{T_j} \frac{l_{p_j}^2(x) + l_{q_j}^2(x)}{2 \cdot A_j(x)} \quad \textbf{(phase 2)}. \tag{8.4}$$

Here, we denote by x the coordinate vector of all corner points of all triangles of a given face after the projection onto the two-dimensional plane, by $A_j(x)$ the area of a given triangle $T_j(x)$, and by $l_{p_j}(x), l_{q_j}(x)$ the lengths of two of its edges. By $d(x)$ we denote the displacement vector containing the displacements of all points x from the initial state prior to the optimization process. Furthermore, we denote by $\det(T_j(x))$ the determinant of the transformation matrix which belongs to the affine mapping from the unit triangle, i.e., the triangle with the corner points $(0,0),(1,0)$, and $(0,1)$, to a given triangle $T_j(x)$. We also introduce scalar weighting factors λ_1, λ_2, and λ_{reg} to control the ratio of the different terms within the objective functions. The specific values for these weights were chosen heuristically and for all our computations, we have used the values $\lambda_1 = 1, \lambda_2 = 50$, and $\lambda_{\text{reg}} = 10$.

Let us briefly motivate our objective functions in more details. Prior to the optimization of phase 1, we locally reorder the triangle corners, such that $\det(T_j(x))$ is negative for all flipped triangles. In order to do so, we start with one triangle and define it either as flipped or non-flipped. Then, we go through the remaining triangles of the projected face and classify them based on the following equivalence relation: two adjacent triangles are equivalent if and only if they do not overlap. Depending on the label of the initial triangle, we obtain two values for the objective function of phase 1, and we choose our classification into flipped and non-flipped triangles such that we start with the lower value. After this, flipped triangles can always be identified by a negative determinant of the respective transformation matrix. Therefore, we explicitly penalize such negative determinants in phase 1 of our optimization by minimizing the factors $\lambda_1 \cdot e^{-\lambda_2 \cdot \det(T_j(x))}$. Note that we also add the regularization term $\lambda_{\text{reg}} \cdot \|d(x)\|_2^2$ to the objective function to prevent that the projection can be arbitrarily shifted or rotated in the given plane. In phase 2, we minimize the sum of all fractions

$$\frac{l_{p_j}^2(x) + l_{q_j}^2(x)}{2 \cdot A_j(x)}.$$

This specific fraction is inspired by geometrical arguments; see also [56, Sect. 4]. It is minimized for equilateral triangles, i.e., a high value in this fraction corresponds to a triangle with a large aspect ratio. Note that the fraction may actually be infinity if $A_j(x) = 0$. This may happen if a triangle is initially projected onto a straight line. However, in the first optimization phase, small areas are penalized in terms of the determinant, such that we do not obtain values close to zero in the second phase.

We start the optimization procedure with the initial projection onto the plane $x = 0$, $y = 0$, or $z = 0$ that results in the lowest objective value when adding the objective functions (8.3) and (8.4) of phase 1 and phase 2, respectively. Then, we use the gradient descent algorithm as an iterative solver and optimize, i.e., minimize, alternating in succession the two aforementioned objective functions. The optimization procedure is stopped if the norm of the relative change of the coordinate vector of the triangles with respect to the previous iteration is below a factor of 1e-6 in both phases. In Figs. 8.9 and 8.10, we show an exemplary visualization of the different steps of the optimization procedure in phase 1 and phase 2, respectively, for a METIS face consisting of ten triangles. Let us note that for all tested faces in Section 8.3, the optimization procedure did

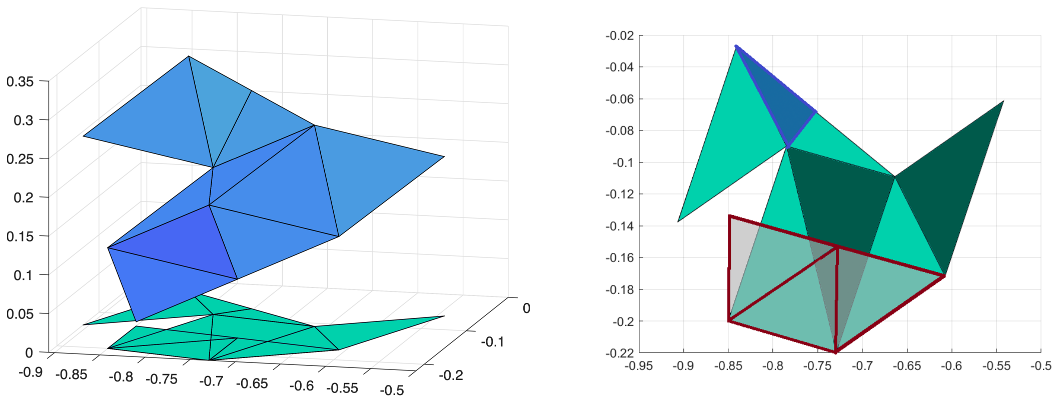


Figure 8.8: **Left:** Example of a typical METIS face in the three-dimensional space (blue triangles) and its corresponding projection onto the two-dimensional plane $z = 0$ (green triangles). **Right:** Due to the projection, we obtain both flipped triangles, which are marked in grey with red edges, and degraded triangles with a large aspect ratio, of which one is marked in blue. The different shades of green are only introduced for visualization purposes and do not have any physical meaning. Taken from [75].

always converge in phase 1 and phase 2 before the maximum number of iterations was reached, which we set to 500. Additionally, in almost all cases, only optimizing twice in phase 1 and once in phase 2 - alternating in succession - was necessary to obtain an appropriate projection of a given METIS face.

As the next step, we construct the smallest possible two-dimensional tensor product grid aligned with the coordinate axes covering the obtained optimized two-dimensional projection of the face; see also Fig. 8.11 (left) for an example. Let us remark that this grid has a natural ordering of the grid points, e.g., starting in the lower left corner and proceeding row by row to the upper right corner. We then make use of barycentric coordinates to map the grid, while conserving the corresponding ordering, back into the original triangles in the three-dimensional space. Based on the ordering of the grid points in two dimensions, we can now establish a consistent ordering of the points in three dimension; see also Fig. 8.11 (right).

Let us briefly recapitulate the complete process for obtaining the auxiliary grid points with a consistent ordering for each face. First, we project the face from the three-dimensional space onto a two-dimensional plane; see Fig. 8.8 (left). Second, we remove all flipped triangles (phase 1) and optimize the shape of all triangles (phase 2) of the projected face in an iterative optimization process; see Figs. 8.9 and 8.10. Finally, we cover this optimized face by a two-dimensional tensor product grid with a natural ordering and project these points back to the original face in three spatial dimensions; see Fig. 8.11. For this purpose, local barycentric coordinates can be used.

Let us note that, in our numerical experiments in Section 8.3, the described procedure was always successful. However, in general, there may be rare cases where our optimiza-

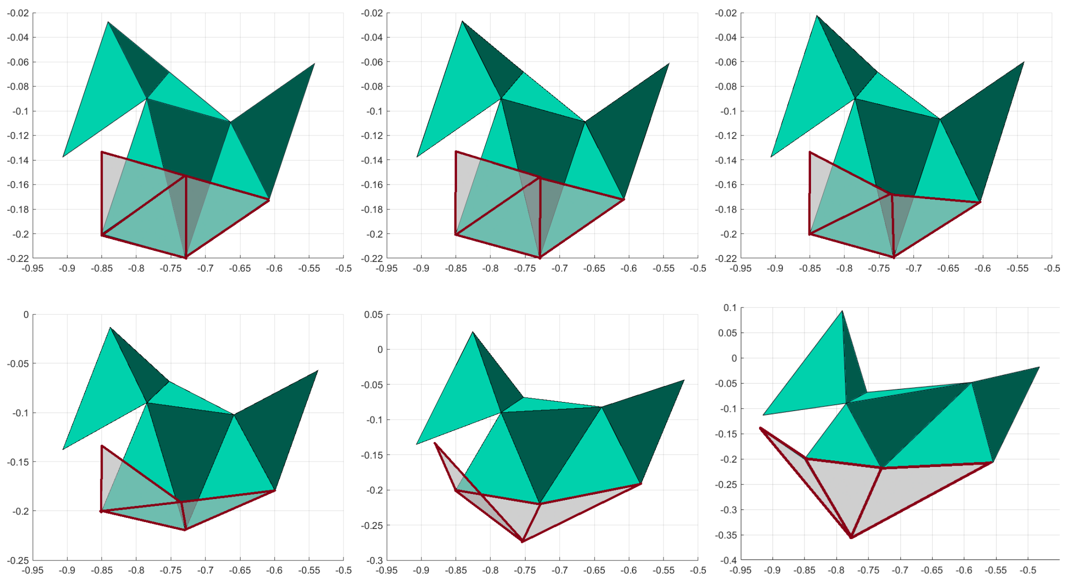


Figure 8.9: Visualization of the optimization process of the original projection of a METIS face in two dimensions in **phase 1** after 0, 20, 30, 50, 100 and 150 iteration steps (from upper left to lower right). Taken from [75].

tion does not converge to an acceptable two-dimensional triangulation. For instance, in case of irregular decompositions, it is possible that a subdomain is completely enclosed by another subdomain, such that the face between the two subdomains is actually the complete boundary of the interior subdomain. In this case, we cannot remove all flipped triangles without changing the structure of the face. If we detect that our optimization does not converge to an acceptable solution, we can still proceed in one of the two following ways: either we mark the eigenvalue problem corresponding to the face as necessary, or we split the face into smaller faces and consider each of the smaller faces separately in our ML-FETI-DP algorithm.

Second step – Extrusion of the auxiliary grid into three dimensions Starting from the ordered auxiliary points on the face, we can now build a three-dimensional sampling grid. For this purpose, for each of the auxiliary points on the face, we first define a *sampling direction vector* pointing into one of the two adjacent subdomains. Second, we extrude the two-dimensional auxiliary grid on the actual METIS face into the two neighboring subdomains along the sampling directions, resulting in a three-dimensional sampling grid. Note that the first layer of sampling nodes does not lie on the face itself but next to it; cf. Fig. 8.2. Moreover, we neglect all points of the auxiliary grid, which lay outside the METIS face, and encode all corresponding points in the three-dimensional sampling grid by the dummy value -1. Similar as for edges obtained by a two-dimensional METIS decomposition, choosing the normal vectors of the triangles as sampling direction vectors in the extrusion process can lead to gaps in the three-dimensional sampling grid close to

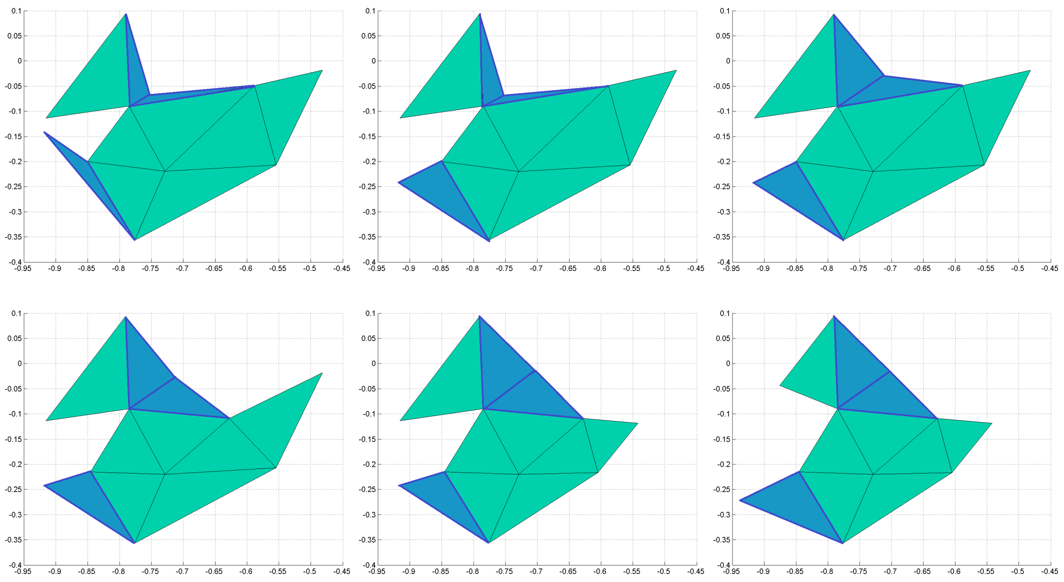


Figure 8.10: Visualization of the optimization process of the original projection of a METIS face in two dimensions in **phase 2** after 0, 10, 30, 50, 70 and 100 iteration steps (from upper left to lower right). Here, the initial state is the same as the final state in Fig. 8.9. Taken from [75].

the face; see also Fig. 8.3 for a two-dimensional graphical representation. This is caused by the fact that, in general, METIS faces are not smooth. As we have already explained for edges in the two-dimensional case, the neighborhood of an equivalence class is usually the most relevant for the decision if adaptive constraints are necessary or not. Therefore, the aforementioned gaps in the sampling grid should be minimized; see also Section 8.1.3 and the related numerical experiments in Section 8.2.5. To avoid these gaps and to obtain sampling points in a preferably high number of finite elements close to the face, we suggest the use of sampling directions obtained by a moving average iteration over the normal vectors of the face. In some sense, this can be interpreted as a smoothing of the face or, more precisely, a smoothing of the field of normal vectors of the face.

The following procedure turned out to be the most appropriate for our purposes in the sense that, on average, for each face and each neighboring subdomain, it results in the highest number of sampled elements relative to the overall number of elements in the subdomain. Here, we first uniformly refine all triangles of a given METIS face once by subdividing each triangle of the face into four new regular triangles. For each of the resulting finer triangles, we compute the normal vector originating in its centroid. We then use the normal vectors of the refined triangulation to compute a single sampling direction for each triangle of the original triangulation of the face. For this purpose, we first smooth the field of normal vectors of the refined triangulation by using a component-wise moving average, applied twice recursively with a fixed window length of 3. Subsequently, we obtain the final sampling direction of the original triangles by computing the average

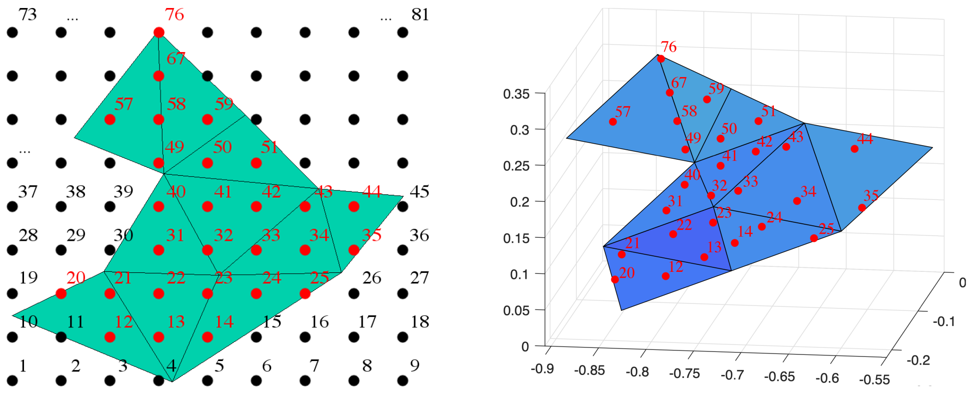


Figure 8.11: Visualization of the natural ordering of the grid points obtained by projecting and optimizing a METIS face. **Left:** Two-dimensional projection of the original face (shown on the right) after both optimization phases have been carried out; the optimized projection is covered by a regular grid with natural ordering; same face as in the last picture of Fig. 8.10. **Right:** Original face in three dimensions with corresponding grid points; numbers are obtained by a projection from two dimensions (left) back to three dimensions using barycentric coordinates. Taken from [75].

of the resulting normal vectors of all corresponding four finer triangles. Subsequently, we use the same sampling direction for all points of the auxiliary grid which are located in the same triangle.

Let us briefly describe the moving average approach and the meaning of the window length in more detail. For each triangle of the refined face, one after another, we replace the normal vector by a component-wise average of the normal vector itself and the normal vectors of certain surrounding triangles. The triangles considered in the averaging process are aggregated recursively as follows. In a first step, for a given triangle, we consider all neighboring triangles that share an edge with the given triangle to obtain a patch of triangles with a window length of 1. Recursively, for an increasing window length, we additionally consider all triangles that share an edge with a triangle that has been selected in the previous step. Please see also Fig. 8.12 for an exemplary visualization of all considered triangles for a moving average with the window length of 3.

Finally, we use the obtained sampling directions to compute the final three-dimensional sampling grid in the two neighboring subdomains of the face. In Fig. 8.13, we visualize all sampled (middle) and non-sampled (right) finite elements using the described procedure for an exemplary METIS face. Here, we denote a finite element as *sampled* if it contains at least one sampling point, otherwise as *non-sampled*. We can observe from Fig. 8.13 that, especially in the close neighborhood of the face, we obtain sampling points in almost all finite elements. In analogy to the two-dimensional case, we obtain the final input for our neural network by creating a vector which contains the evaluations of the coefficient function ρ or the Young modulus E, respectively, for all points in the computed sampling grid.

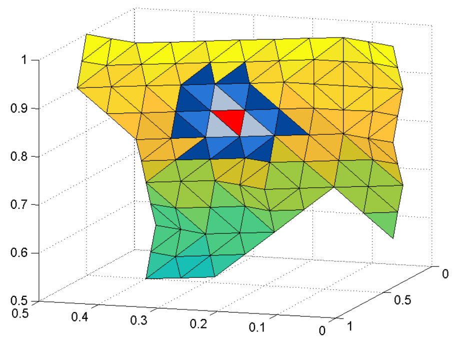


Figure 8.12: Visualization of the moving average procedure for METIS faces to obtain the sampling direction vectors for the extrusion of the auxiliary grid. For the red triangle, all grey, light blue and dark blue triangles are considered recursively as grouped by colors for a moving average with a window length of 3. Taken from [75].

8.1.4.3 Training and validation phase

For the training and validation of the neural network applicable to three-dimensional problems, we use a data set containing approximately 3 000 configurations of pairs of coefficient functions and subdomain geometries for two subdomains sharing a face. To obtain the output data, i.e., the correct classification labels for the training of the neural network, we solve the face eigenvalue problem described in Section 3.5 for each of these configurations. Note that the correct classification label for a specific face $\mathcal{F}_{ij}$ does not only depend on the geometry and the coefficient distribution, but also on the underlying PDE. Therefore, we will use the same configurations for diffusion and elasticity problems but compute the correct classification labels separately.

For the two-dimensional case, we use only two edge geometries, i.e., a regular edge and an edge with a single jag, and combine them with a set of carefully designed coefficient distributions, resulting in a total of 4 500 configurations; see Section 8.1.3. We refer to this data set based on manually designed coefficient distributions as *smart data*. Since both the domain decomposition and the coefficient distribution may be more complex in three dimensions compared to two dimensions, we need to use a modified approach for the generation of training and validation data for three-dimensional decompositions. In particular, we consider six different mesh discretizations resulting from regular domain decompositions of the unit cube into $4 \times 4 \times 4 = 64$ or $6 \times 6 \times 6 = 216$ subdomains of a size defined by $H/h \in \{6, 7, 8\}$. For each of these mesh discretizations, we generate 30 different randomly generated coefficient distributions based on the approach discussed in Section 8.2.4. More precisely, we control the ratio of high versus low coefficient voxels and impose some light geometrical structure. In particular, we build connected beams of a high coefficient with a predefined length in x, y, or z direction, and additionally combine them by a pairwise superimposition; cf. Section 8.2.4 for a more detailed description of the analogous two-dimensional case. In Fig. 8.14, we show an exemplary coefficient distribution in three dimensions which has been generated using the described technique.

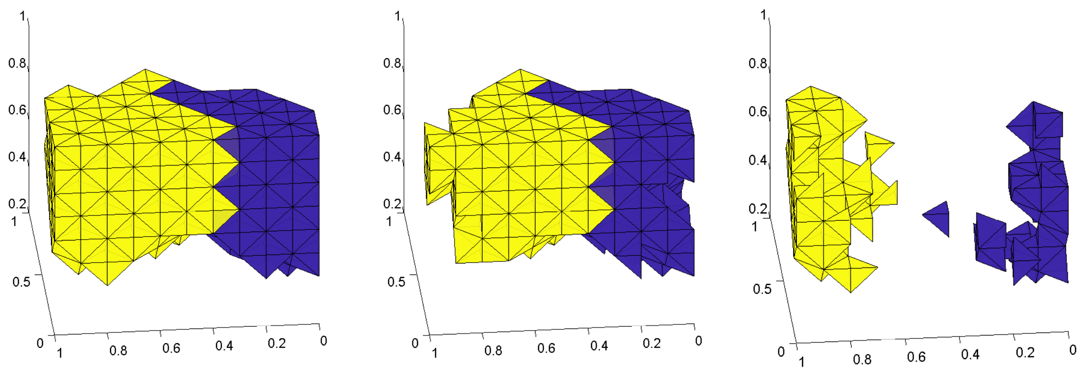


Figure 8.13: Visualization of a METIS face between two neighboring subdomains **(left)** and all sampled **(middle)** and non-sampled **(right)** FE's when using the described sampling procedure. The two different subdomains are visualized by yellow and blue FE voxels. Taken from [75].

In analogy to the two-dimensional case, we refer to this set of coefficient functions as *random data*. For each combination of mesh and coefficient distribution, we now consider the aforementioned regular domain decomposition as well as a corresponding irregular domain decomposition into 64 or 216 subdomains, respectively, obtained using the graph partitioning software METIS [87]. Finally, we consider the eigenvalue problems corresponding to all resulting faces combined with the different coefficient distributions. As mentioned before, we obtain a total of approximately 3 000 configurations. Note that, in general, using a smaller number of METIS subdomains than 64 for the generation of the training data, we obtained face geometries which resulted in poor generalization properties of our neural network. Moreover, in contrast to the two-dimensional case, where we need at least 4 500 smart data configurations, we are now able to obtain very good accuracy values for a total of only roughly 3 000 data configurations. This is likely due to the much smaller number of finite elements per subdomain in practical experiments with three-dimensional domains, compared to the two-dimensional experiments in Section 8.2.

For the sampling within the generation of training and validation data, we select 22 points in both of the two tangential directions of the auxiliary grid of a face and 22 points in the orthogonal direction for each of the two adjacent subdomains. Hence, we obtain approximately two sampling points in each finite element when using a subdomain size of $H/h = 10$.

As in Section 8.1.3, we train the neural network using the Adam [90] optimizer, a specific variant of the SGD method with an adaptive learning rate. The hyper parameters for the training process and the neural network architecture are again chosen based on a grid search with cross-validation. More precisely, we have compared the training and generalization properties of different neural networks for several random splittings of our entire data set into 80 % training and 20 % validation data; cf. also Section 8.1.3 for details on the hyper parameter search space. For three-dimensional problems, using a neural network with $[40, 30, 25]$ hidden layers, i.e., three hidden layers with 40, 30, and

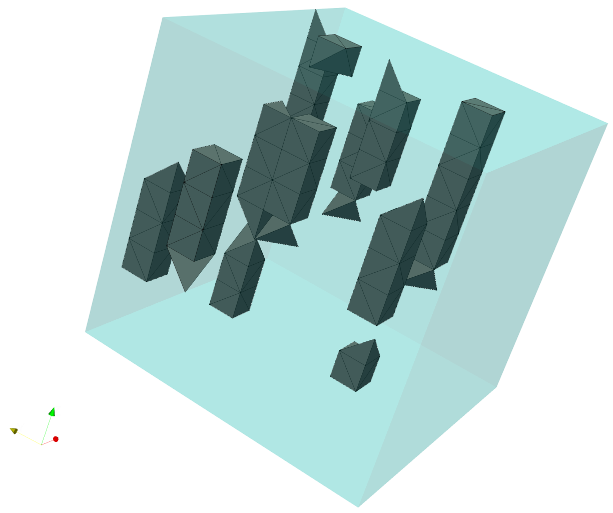


Figure 8.14: Example of a randomly distributed coefficient function in the unit cube obtained by using the same randomly generated coefficient for a horizontal or vertical beam of a maximum length of 4 finite element voxels. The grey voxels correspond to a high coefficient and we have a low coeffient of 1 otherwise. Visualization for $2 \times 2 \times 2$ subdomains and $H/h = 5$.

25 neurons, respectively, results in the highest accuracy values for both the training and validation data. The ROC curve and a precision-recall plot of the neural network with the optimal hyper parameters are presented in Fig. 8.15. Let us note that we use the same neural network for both, regular and METIS decompositions, in our numerical experiments in Section 8.3.

8.1.4.4 Results on the training data in three dimensions

On the complete set of training and validation data, we obtain the results listed in Table 8.3. As in Section 8.1.3, we use the classification thresholds $\tau \in \{0.45, 0.5\}$ for the two-class classification and $\tau \in \{0.4, 0.5\}$ for the three-class classification, respectively. For the two-class classification, we observe nearly the same accuracy values when using the classification threshold $\tau = 0.5$ or $\tau = 0.45$. For the three-class classification, however, lowering the threshold to $\tau = 0.4$ results in a lower accuracy value than for using the threshold of $\tau = 0.5$. In both cases, the number of false negative faces, which corresponds to the number of critical faces not detected by the algorithm and which are critical for the convergence of the iterative FETI-DP solver, can be reduced by decreasing the threshold τ. In our context, we denote this approach to improve the robustness as *overshooting*. In Section 8.3, we will always compare the results for the default threshold $\tau = 0.5$, and the overshooting threshold, i.e., $\tau = 0.45$ and $\tau = 0.4$ for the two-class and three-class model, respectively. More details on heuristic strategies for the choice of the threshold τ are given in Section 8.1.5.

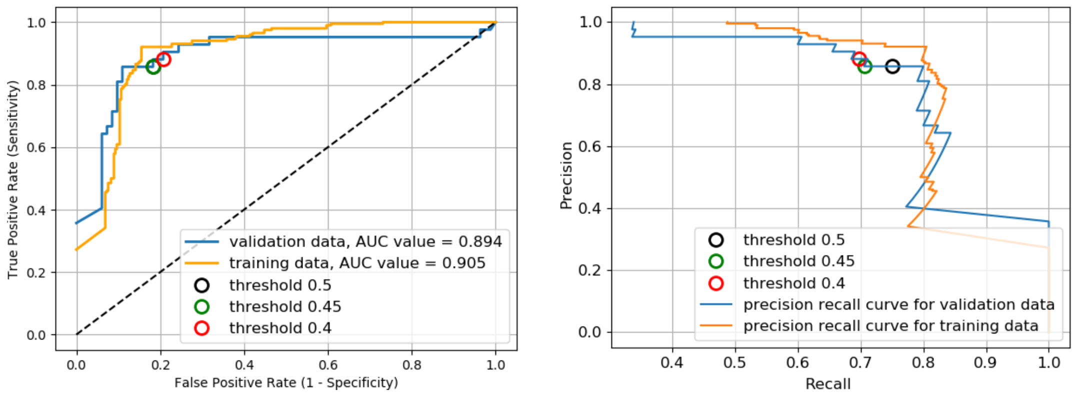


Figure 8.15: ROC curve and precision-recall plot for the optimal model for three-dimensional problems obtained by a grid search. See Section 8.1.5 for a definition of precision and recall. The thresholds used in Section 8.3 are indicated as circles. Taken from [75].

classification type	**threshold** τ	**fp**	**fn**	**acc**
two-class classification	0.45	2.76%	1.76%	95.5%
	0.5	1.70%	3.40%	94.9%
three-class classification	0.4	5.2%	1.7%	93.1%
	0.5	2.1%	2.3%	95.6%

Table 8.3: Results on the complete randomized training and validation data set in three spatial dimensions. We define the accuracy (acc) as the number of true positives and true negatives divided by the total number of training and validation configurations. Table already published in [75, Table 1].

8.1.5 Heuristic strategies for the choice of the ML threshold

As already stated above, we use the softmax function as the activation function for the output layer of our neural network. Thus, for a multi-class classification problem, the obtained output values of the neural network can be interpreted as the probability values that a given input vector belongs to each of the given classes; see also (8.2). Given an output vector of probability values, the machine learning algorithm has to make a decision into which of the different classes a given input vector should be categorized. Illustratively spoken, we have to define a decision boundary between the different classes. The most intuitive classification approach would be to always select the respective class with the highest corresponding probability value; cf. also [169]. However, in our specific application within adaptive coarse spaces, this may not always be the best choice.

For an illustrative analysis of the problem, let us consider an exemplary neural network which is trained to distinguish between critical and uncritical edges in a two-dimensional domain decomposition. Hence, we consider a two-class classification problem. In order to transfer the following discussion to the standard terminology within machine learning, we also refer to edges where (at least) one adaptive coarse basis function is necessary as *positive* or *positive edges* and to edges where the eigenvalue problem is unnecessary as *negative* or *negative edges*.

Of course, using our machine learning method, we are interested in a preferably high accuracy of the classification. On the one hand, expecting a perfect accuracy of 100% is a rather unrealistic expectation since the training of a neural network is a highly non-linear optimization process and also depends on stochastic variables, as, e.g., the choice of the parameter initialization and the choice of the batches in an SGD method. On the other hand, we can take advantage of available information regarding our specific ML problem to optimize the decision of the neural network specifically with respect to our purposes. Generally speaking, our predominant goal is to identify all critical edges, for which adaptive coarse constraints are necessary to obtain a robust FETI-DP algorithm. In particular, we preferably try to avoid edges, which are incorrectly identified as negative by the neural network since they might decrease the rate of convergence of the resulting FETI-DP algorithm. We refer to these edges as *false negatives* or *false negative edges*. On the contrary, edges which are incorrectly identified as positive by the neural network, are not critical for the convergence properties but only result in the solution of an additional eigenvalue problem which does not lead to any additional coarse constraints since the tolerance criterion will not be satisfied. We refer to the latter edges as *false positives* or *false positive edges*. Consequently, only false negative edges might negatively affect the rate of convergence of the iterative solver whereas false positive edges only correspond to some extra computational effort. Therefore, our predominant goal is to avoid any false negative edges, while false positive edges are - to a certain extent - acceptable. With this in mind, we try to define an optimal choice for the ML decision threshold τ for the decision boundary between the two classes. Given that - roughly speaking - we aim to avoid false negative edges at the cost of some false positive edges, using a value of $\tau = 0.5$, which results in an equal weighting between the two classes, might not always be the optimal choice.

For a two-class classification problem, we can use the ROC curve as well as a precision-recall graph of the training and validation data as an indicator for the selection of an appropriate decision threshold τ. For the sake of completeness, let us briefly present the definitions of two different performance metrics named *precision* and *recall*. In a two-class classification problem, the precision is defined by

$$\text{precision} := \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8.5}$$

and the recall by

$$\text{recall} := \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{8.6}$$

where TP denotes the number of true positives, FP the number of false positives, and FN the number of false negatives. Thus, the precision measures how many of the data predicted as positive are actually positive, whereas recall measures how many of the positive samples are categorized correctly by the positive predictions [135]. Consequently, recall is often used as a performance metric when we need to identify all positive examples, which is the case in our application where we aim to correctly identify all critical edges. Hence, we principally want the recall to reach a high value. Simultaneously, we also prefer the precision to be high as well, since a recall of one could also be achieved for a neural network that categorizes all edges simply to class 1. This, however, would simply result in the original adaptive FETI-DP coarse space and no eigenvalue problems or computational effort would be saved. As we can observe from the latter explanations, the optimal choice of the decision threshold τ is always a trade-off between precision and recall of the resulting ML algorithm. In our application, we aim to define the value of τ such that, predominantly, the number of false negative edges is reduced to an essential minimum or, if possible, to zero, whereas the number of false positive edges remains at a modest value. As we will observe in Section 8.2, for almost all tested problems, it is possible to completely eliminate all false negative edges for the decision threshold $\tau \in \{0.4, 0.45\}$ while also obtaining an acceptable number of false positive edges. This implies that the considered classification problem can be handled quite well by the trained neural networks.

For completeness, let us mention that the above described strategies are also equally valid for three-dimensional domain decompositions and the classification of critical faces.

8.2 Numerical results for adaptive FETI-DP and ML-FETI-DP for two-dimensional test problems

In this section, we provide comparative results for classic FETI-DP, the adaptive FETI-DP method introduced in Section 3.5, and our hybrid ML-FETI-DP algorithm for different test problems in two spatial dimensions. As already mentioned in the beginning of this chapter, we deliberately present the results for two- and three-dimensional problems in separate subsections. Although the idea of using deep learning for the identification of critical edges or faces is the same in both cases, the specific implementation differs in some aspects. In particular, the design and generation of training and validation data is more complex in three dimensions. Additionally, a slightly different network architecture as well as different values for the decision threshold τ are used.

In the following, we show numerical results for two types of different coefficient distributions which are used for stationary diffusion and linear elasticity problems; see Section 8.2.1 for a detailed description. We consider both regular domain decompositions of the unit square as well as irregular decompositions obtained by METIS [87]. For both cases, we will consider a two-class classification model as well as an extended three-class classification model using frugal constraints; cf. Section 8.1.2. The presented results in Sections 8.2.2 and 8.2.3 rely on the carefully selected set of *smart data* which are used for the training and validation of the respective neural networks. In Section 8.2.4, we extend these results by comparing the performance of ML-FETI-DP for different sets of training data used for the optimization of the neural network. Thus, we compare the smart training data with different sets of randomized training data and combinations of both types. In Section 8.2.5, we further examine the sampling procedure for two-dimensional problems in more detail by reducing the number of computed sampling points and investigate the effect on the generalization properties of the resulting neural network.

Parts of this section have already been published in modified or unmodified form in [72, 74, 76, 77].

8.2.1 Description of the tested coefficient functions

In the following subsections, we consider two types of discontinuous coefficient functions or discontinuous material distributions, respectively, which result in sharp jumps along and across the interface of the domain decomposition. First, we consider a synthetic coefficient distribution as visualized in Fig. 8.16 (left) to which we refer to as *circle problem*. We use this coefficient distribution to define a stationary diffusion problem where we consider a coefficient of $\rho = 1e6$ in the dark blue circles and $\rho = 1$ in the rest of Ω. Second, we consider a more realistic coefficient distribution as visualized in Fig. 8.16 (middle). We refer to the respective model problem as *microsection problem*. We use this coefficient distribution to define both a stationary diffusion as well as a linear elasticity problem. For the scalar diffusion case, we consider a coefficient of $\rho = 1e6$ in the black part of the microsection and $\rho = 1$ elsewhere. For linear elasticity, we set the Poisson ratio constantly to $\nu = 0.3$ and only consider jumps in the Young modulus E. In particular, we set $E = 1e3$ in the black part of the microsection and $E = 1$ elsewhere.

Let us note that the microsection presented in Fig. 8.16 (middle) is a subsection of the larger microsection in Fig. 8.16 (right) of a dual-phase steel. In our numerical results, we discuss our approach in more detail for a single subsection of the microsection whose size is suitable for our MATLAB [134] computations. Additionally, we consider a total of ten different subsections of the microsection in Fig. 8.16 (right) that cover the whole structure of the microsection to prove that our ML-FETI-DP algorithm is robust for different coefficient distributions.

Finally, let us remark that in the following experiments, we always choose the mesh resolution such that the coefficient function is constant on each finite element. Then, the predictions and the accuracy of our machine learning classification algorithm is independent of the mesh resolution of the finite element mesh.

8.2.2 Numerical results for the two-class model

Let us first discuss our experiments for the two-class classification model. Here, we train a neural network to distinguish between critical edges, where the eigenvalue problem results in additional adaptive constraints, and edges where the eigenvalue problem is unnecessary. For the remainder of this section, we will refer to edges where (at least) an adaptive coarse basis function is necessary for robustness as *positive* or *positive edges* and to edges where the eigenvalue problem is unnecessary as *negative* or *negative edges*. To obtain the corresponding classification for each edge which is used as output data for the neural network, we use a tolerance of $TOL = 100$ in the adaptive FETI-DP algorithm. For the classification of edges in ML-FETI-DP, we will consider two different values for the (ML) threshold τ, i.e., $\tau \in \{0.45, 0.5\}$ for the two-class model.

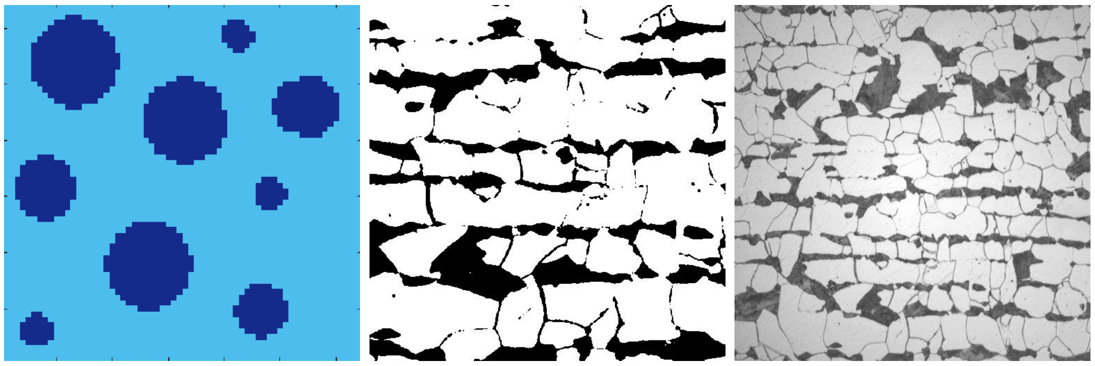


Figure 8.16: Two heterogeneous coefficient distributions used for the numerical comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP in two dimensions. **Left:** Coefficient function with randomly distributed circles of different radii. We have $\rho = 1e6$ in the dark blue circles and $\rho = 1$ elsewhere. **Middle:** Subsection of a microsection of a dual-phase steel obtained from the image on the right. We consider $\rho = 1e6$ or $E = 1e3$, respectively, in the black part of the microsection and $\rho = 1$ or $E = 1$ elsewhere. **Right:** Complete microsection of a dual-phase steel. Courtesy of Jörg Schröder, University of Duisburg-Essen, Germany, orginating from a cooperation with thyssenkrupp. Taken from [72].

8.2.2.1 Regular domain decompositions

As a first set of numerical experiments, we consider a regular domain decomposition of the circle problem in the unit cube and the microsection problem in Fig. 8.16 into $8 \times 8 = 64$ subdomains. For both problems, we use a discretization with 8 192 finite elements per subdomain. For the numerical solution of both problems we apply classic FETI-DP, adaptive FETI-DP and ML-FETI-DP. In Fig. 8.17 (left) and Fig. 8.18, we visualize the ML classification of critical edges as obtained by ML-FETI-DP for a stationary diffusion problem using the following color code: edges which ML-FETI-DP correctly identifies as positive are marked in green (true positive), edges which ML-FETI-DP incorrectly identifies as positive are marked in yellow (false positive), and edges which ML-FETI-DP incorrectly identifies as negative are marked in red (false negative). All edges which are correctly identified as negative (true negative) are not marked. Let us remark that the false positives, i.e., yellow edges are not critical for the robustness and convergence of the algorithm. In fact, for each false positive edge only a single unnecessary eigenvalue problem is solved which - in principle - could be omitted. In contrast, false negatives, i.e., red edges might decrease the rate of convergence of the resulting adaptive FETI-DP algorithm. Therefore, we also consider the approach of *overshooting* and lowering the ML-threshold τ in some cases.

In Fig. 8.17 (left), we see that only two yellow edges (false positives) but no critical red edges (false negatives) occur for the circle problem. Here, we potentially save 92% of the eigenvalue problems. For the microsection problem used to define a stationary

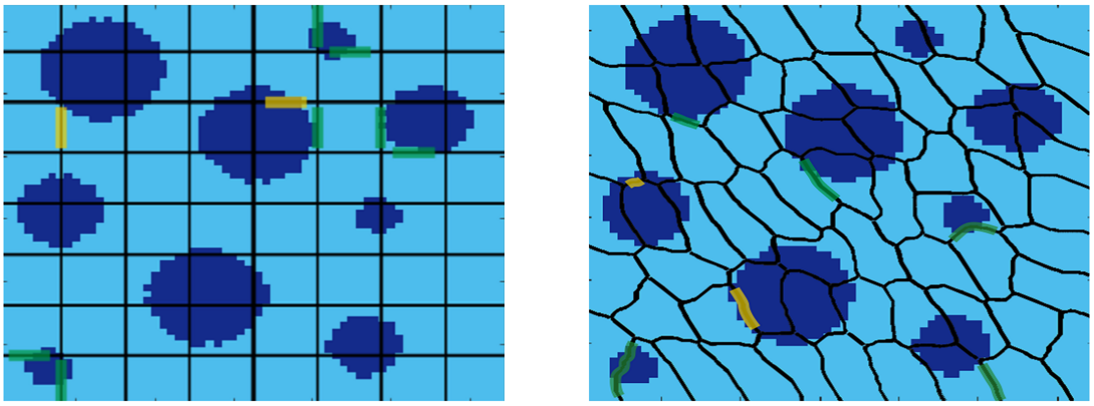


Figure 8.17: Circle problem for stationary diffusion with marked edges as obtained by ML-FETI-DP for the two-class model: true positives are marked in green, false positives are marked in yellow, and false negatives are marked in red. **Left:** Regular domain decomposition into 8×8 subdomains; cf. also Table 8.4. **Right:** METIS domain decomposition into 64 subdomains; cf. also Table 8.7. Figure in modified form in [72].

diffusion problem, we save 65% of the eigenvalue problems using an ML threshold of $\tau = 0.5$ and 60% using an ML threshold of $\tau = 0.45$ for the decision boundary between the two classes; cf. also Table 8.4. In the latter case, we have no false negative edges and two false negative edges in the first case. See Fig. 8.18 for a graphical representation of both results. Additionally, we provide the condition numbers and iteration counts for all discussed examples in Table 8.4. As we can observe from Table 8.4, the two false negative edges for the microsection problem and $\tau = 0.5$ increase the condition number, but the convergence of ML-FETI-DP is still fast. In particular, the iteration number is still satisfactory which implies that the relatively high condition number estimate is caused by only a few high eigenvalues of the preconditioned system. To obtain also a low condition number, which is comparable to adaptive FETI-DP, an overshooting with the ML threshold of $\tau = 0.45$ works as expected. In this case, we obtain no false negative edges for ML-FETI-DP and the same condition number as for adaptive FETI-DP.

To prove the robustness of ML-FETI-DP independently of the specific subsection of the microstructure shown in Fig. 8.16 (right), we additionally summarize numerical results for ten different subsections in an aggregated form. We therefore present averages and maximum values of the condition number and iteration counts aggregated for all subsections in Table 8.5. Here, we can observe that for all subsections, we are able to obtain no false negative edges when using the ML threshold $\tau = 0.45$. In particular, the application of ML-FETI-DP results in almost the same condition numbers and iteration counts as the adaptive FETI-DP method, whereas a high number of the eigenvalue problems can be omitted.

We further provide numerical results for linear elasticity problems in two dimensions for the same ten different subsections of the microsection problem in Table 8.6. As for the

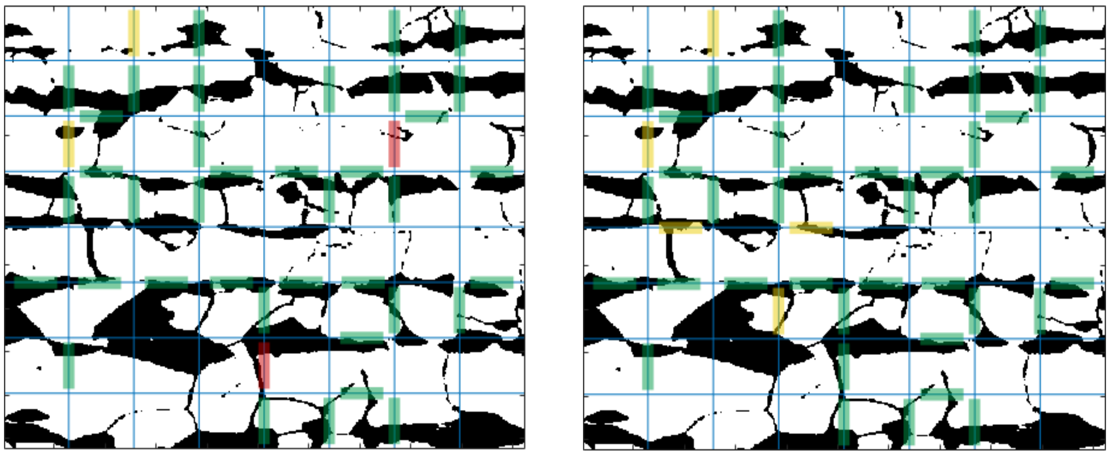


Figure 8.18: Microsection problem for stationary diffusion with marked edges as obtained by ML-FETI-DP for the two-class model for a regular domain decomposition into 8×8 subdomains: true positives are marked in green, false positives are marked in yellow, and false negatives are marked in red; cf. also Table 8.4. **Left:** ML threshold $\tau = 0.5$. **Right:** ML threshold $\tau = 0.45$. Figure in modified form in [72].

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
	classic	-	1.04e6	56	0	-	-	-
Circle Problem	adaptive	-	8.82	35	112	-	-	-
	ML	0.5	8.83	35	9	2	0	0.98
	classic	-	-	>300	0	-	-	-
Microsection	adaptive	-	15.86	36	112	-	-	-
Problem	ML	0.5	9.64e4	45	39	2	2	0.96
	ML	0.45	15.86	36	44	5	0	0.95

Table 8.4: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **regular domain decomposition** for the **two-class model** for **stationary diffusion**; cf. also Fig. 8.17 (left) and Fig. 8.18. We show the ML threshold (τ), the condition number (cond), the number of CG iterations (it), the number of solved eigenvalue problems (evp), the number of false positives (fp), the number of false negatives (fn), and the accuracy in the classification (acc). Table already published in [72, Table 3].

Ten Different Microsection Problems							
Algorithm	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
adaptive	-	11.04	34.6	112.0	-	-	-
		(15.87)	(38)	(112)	-	-	-
ML	0.5	8.61e4	39.5	45.0	1.6	1.9	0.97
		(9.73e4)	(52)	(57)	(2)	(3)	(0.96)
ML	0.45	11.04	34.6	46.9	4.4	0	0.96
		(15.87)	(38)	(59)	(6)	(0)	(0.94)

Table 8.5: Results for 10 different subsections of a microsection of a dual-phase steel for the **two-class model** for **stationary diffusion**. Comparison of adaptive FETI-DP and ML-FETI-DP for **regular domain decompositions**. We show the average values as well as the maximum values (in brackets). See Table 8.4 for the column labeling. Table already published in [72, Table 4].

stationary diffusion problem, we use the smart data set for the training and validation of the neural network. Here, we consider a domain decomposition into 8×8 subdomains and use a discretization of 1 800 finite elements per subdomain. We present average and maximum values of the condition numbers and iteration counts for all considered subsections of the microsection in Table 8.6. Analogously to the diffusion case in Table 8.5, we are able to eliminate all false negative edges and thus obtain a robust algorithm when using the ML threshold of $\tau = 0.45$. Consequently, for $\tau = 0.45$, we obtain almost the same condition numbers for ML-FETI-DP as for adaptive FETI-DP.

8.2.2.2 METIS domain decompositions

As a second set of experiments, we consider METIS domain decompositions into 64 subdomains of the circle problem as well as the microsection problem from Fig. 8.16. In analogy to Section 8.2.2.1, we apply classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for the numerical solution of both model problems. In Fig. 8.17 (right) and Fig. 8.19, we visualize the ML classification of critical edges as obtained by ML-FETI-DP for a stationary diffusion problem using the same color code as before. Let us recall that only false negative, i.e., red edges are critical for the robustness and the convergence of the iterative solver. As explained in the beginning of this chapter, we can make an effort to improve the robustness of ML-FETI-DP by using an overshooting approach. This usually results in some false positive, i.e., yellow marked edges, which are not critical for the convergence of the algorithm but correspond to additional computational effort.

In Fig. 8.17 (right), we observe that, as for a regular domain decomposition, only two yellow edges (false positives) but no critical red edges (false negatives) occur for the circle problem. Here, we potentially save 96% of the eigenvalue problems compared to adaptive FETI-DP. For the microsection problem, we save the computation of 61% of the eigenvalue problems considering an ML threshold of $\tau = 0.5$ as well as $\tau = 0.45$. In the first case, ML-FETI-DP misses three critical edges (red edges), whereas in the second

Ten Different Microsection Problems							
Algorithm	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
adaptive	-	79.07	87.4	112.0	-	-	-
		(92.83)	(91)	(112)	-	-	-
ML	0.5	9.32e4	92.2	44.0	2.2	2.4	0.96
		(10.32e4)	(95)	(56)	(3)	(3)	(0.95)
ML	0.45	79.07	87.4	48.2	4.8	0	0.95
		(92.83)	(91)	(61)	(7)	(0)	(0.93)

Table 8.6: Results for 10 different subsections of a microsection of a dual-phase steel for the **two-class model** for **linear elasticity**. Comparison of adaptive FETI-DP and ML-FETI-DP for **regular domain decompositions**. We show the average values as well as the maximum values (in brackets). See Table 8.4 for the column labeling. Table already published in [72, Table 5].

case, we are able to eliminate all red edges. See Fig. 8.19 for a visualization of both results. We also provide condition numbers and iteration counts for all discussed examples in Table 8.7. Although ML-FETI-DP misses three critical edges for the microsection problem for an ML threshold of $\tau = 0.5$, the number of iterations remains moderate. Again, this effect is likely to be caused by a small number of high eigenvalues of the preconditioned system. Nonetheless, using a lower threshold of $\tau = 0.45$, we obtain the same robustness as adaptive FETI-DP, i.e., a condition number which is independent of the coefficient contrast. Also for METIS decompositions, we again provide numerical results for ten different subsections of the microsection shown in Fig. 8.16 (right). We present average and maximum values of the condition numbers and iteration counts in Table 8.8. Here, we can observe that for all ten subsections we are able to achieve almost the same condition number and iteration number (on average) for ML-FETI-DP and $\tau = 0.45$ as for adaptive FETI-DP, whereas we can omit a high number of the edge eigenvalue problems.

Furthermore, we additionally test the two-class model of ML-FETI-DP for linear elasticity problems and a METIS domain decomposition into 64 subdomains of the microsection problem. The respective aggregated results for ten different subsections are summarized in Table 8.9. Here, as in the diffusion case, we also obtain a robust algorithm with no false negative edges when using the ML threshold of $\tau = 0.45$, leading to condition number estimates which are independet of the contrast of the Young modulus.

8.2.3 Numerical results for the three-class model

In this section, we discuss numerical results for the extended three-class model of ML-FETI-DP; cf. Section 8.1.2. Here, we slightly modify the definition of edges of class 1 and class 2 and use the frugal constraints as introduced in Chapter 6 to approximate the first eigenmodes for edges assigned to class 1. Consequently, we only set up and solve the eigenvalue problem for edges which are assigned to class 2 by the neural network. This

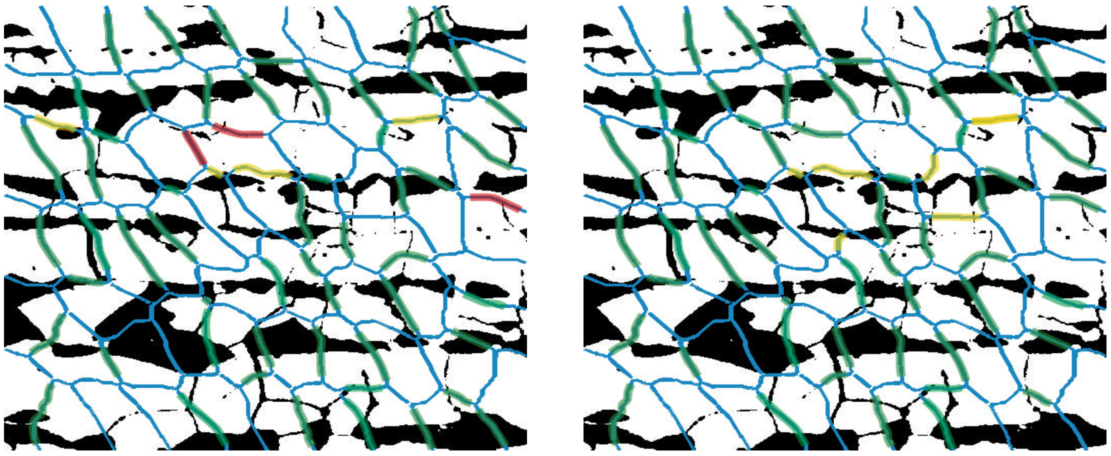


Figure 8.19: Microsection problem for stationary diffusion with marked edges as obtained by ML-FETI-DP for the two-class model for a METIS domain decomposition into 64 subdomains: true positives are marked in green, false positives are marked in yellow, and false negatives are marked in red; cf. also Table 8.7. **Left:** ML threshold $\tau = 0.5$. **Right:** ML threshold $\tau = 0.45$. Figure in modified form in [72].

Model Problem	Algorithm	τ	cond	it	evp	fp	fn	acc
	classic	-	9.18e5	75	0	-	-	-
Circle Problem	adaptive	-	13.56	37	160	-	-	-
	ML	0.5	13.56	37	7	2	0	0.99
	classic	-	-	>350	-	-	-	-
Microsection	adaptive	-	16.52	35	160	-	-	-
Problem	ML	0.5	1.78e4	51	62	3	3	0.96
	ML	0.45	16.52	35	68	6	0	0.96

Table 8.7: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **METIS domain decomposition** for the **two-class model** for **stationary diffusion**; cf. also Fig. 8.17 (right) and Fig. 8.19. See Table 8.4 for the column labeling. Table already published in [72, Table 6].

Ten Different Microsection Problems							
Algorithm	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
adaptive	-	14.81	35.6	160.0	-	-	-
		(22.58)	(38)	(160)	-	-	-
ML	0.5	1.38e4	51.0	63.2	2.0	2.0	0.97
		(2.07e4)	(52)	(73)	(3)	(3)	(0.96)
ML	0.45	14.81	35.6	65.4	6.2	0	0.96
		(22.58)	(38)	(75)	(7)	(0)	(0.95)

Table 8.8: Results for 10 different subsections of a microsection of a dual-phase steel for the **two-class model** for **stationary diffusion**. Comparison of adaptive FETI-DP and ML-FETI-DP for **METIS domain decompositions**. We show the average values as well as the maximum values (in brackets). See Table 8.4 for the column labeling. Table already published in [72, Table 7].

Ten Different Microsection Problems							
Algorithm	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
adaptive	-	85.73	89.4	160.0	-	-	-
		(99.05)	(97)	(160)	-	-	-
ML	0.5	2.74e4	92.8	65.0	2.4	2.6	0.96
		(2.89e4)	(102)	(74)	(3)	(3)	(0.96)
ML	0.45	85.73	89.4	67.2	7.4	0	0.96
		(99.05)	(97)	(77)	(8)	(0)	(0.95)

Table 8.9: Results for 10 different subsections of a microsection of a dual-phase steel for the **two-class model** for **linear elasticity**. Comparison of adaptive FETI-DP and ML-FETI-DP for **METIS domain decompositions**. We show the average values as well as the maximum values (in brackets). See Table 8.4 for the column labeling. Table already published in [72, Table 8].

potentially further decreases the number of necessary eigenvalue problems.

As before, we always use a tolerance of $TOL = 100$ in the adaptive FETI-DP algorithm and, if not stated otherwise, an ML threshold of $\tau = 0.5$ for ML-FETI-DP. Let us remark that we only consider the microsection problem for the following experiments in this section, since for the circle problem, only edges from class 0 and class 1 occur. Thus, using a three-class model for the circle problem does not result in different accuracy values and false positive or false negative rates as for the two-class model.

8.2.3.1 Regular domain decompositions

In the following numerical experiments, we consider the same discretization and domain decomposition as in Section 8.2.2.1. Here, besides classic FETI-DP with primal vertices, adaptive FETI-DP, and ML-FETI-DP, we additionally compare FETI-DP with primal vertex constraints and frugal edge constraints on all edges as described in Chapter 6. Let us recall that the construction of a frugal coarse space does not require the solution of any eigenvalue problems but can be interpreted as a low-dimensional approximation of the adaptive FETI-DP coarse space described in Section 3.5.

In Fig. 8.20, we visualize the ML classification of critical edges using the three-class model of ML-FETI-DP for a stationary diffusion problem. As we can observe from Fig. 8.20 (left), for $\tau = 0.5$, we obtain a single red edge, where ML-FETI-DP assigns this edge falsely to class 1 instead of class 2. Nonetheless, ML-FETI-DP is still robust, since this single edge is not critical for the convergence of the iterative solver; see also Table 8.10. Here, ML-FETI-DP using the three-class model saves 93% of the eigenvalue problems compared to adaptive FETI-DP. Let us note that this is a significant improvement compared to ML-FETI-DP using the two-class model as investigated in Section 8.2.2.1, where we have saved 65% for exactly the same model problem. This is due to the fact that, here, we additionally omit the eigenvalue problem for edges assigned to class 1 where we impose a frugal constraint. Again, considering a lower ML threshold of $\tau = 0.4$ as described in Section 8.1.2, we can eliminate all false negative edges for the prize of additional false positives; see Fig. 8.20 (right). As already mentioned, the latter ones do not negatively affect the robustness of ML-FETI-DP but result in additional computational effort. Also for the three-class model, we test ML-FETI-DP for the same ten subsections of the complete microsection as considered in Section 8.2.2. We present the respective average and maximum values of the condition numbers and iteration counts in an aggregated form in Table 8.11.

8.2.3.2 METIS domain decompositions

To complete the set of numerical examples for ML-FETI-DP based on the smart training data, we consider the same test problem as in Section 8.2.3.1, but using irregular domain decompositions obtained by METIS. In particular, we use the same discretization and number of subdomains as in Section 8.2.2.2 for the two-class model to enable a direct comparison. The obtained classifications of critical edges using the three-class model for the microsection problem and a stationary diffusion problem are visualized in Fig. 8.21.

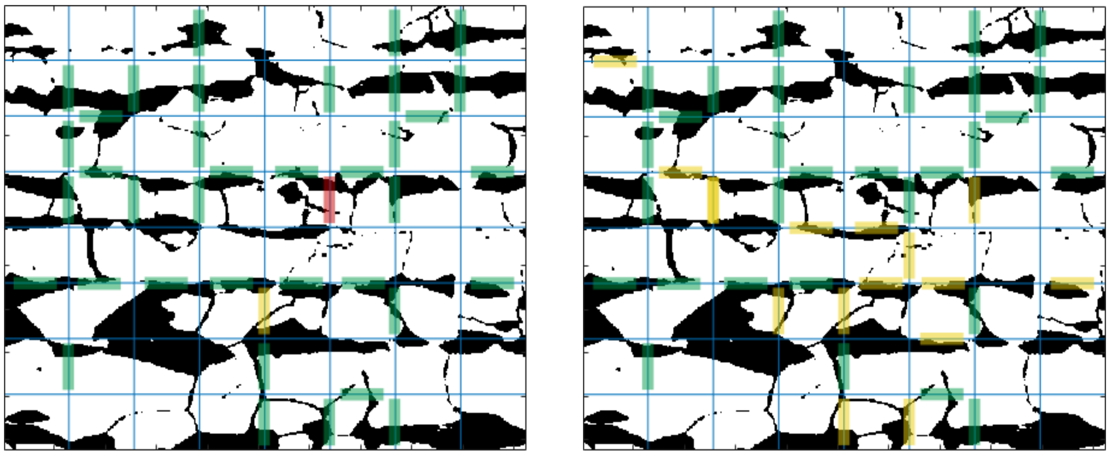


Figure 8.20: Microsection problem for stationary diffusion with marked edges as obtained by ML-FETI-DP for the three-class model for a regular domain decomposition into 8×8 subdomains. Correctly assigned edges from class 1 and class 2 are marked in green. Edges from class 1 which are falsely assigned to class 0, and edges from class 2 which are falsely assigned to class 0, are marked in red. Edges from class 0, which are falsely assigned to class 1, and edges from class 1, which are falsely assigned to class 2, are marked in yellow; cf. also Table 8.10. **Left:** ML threshold of $\tau = 0.5$. **Right:** ML threshold $\tau = 0.4$. Figure in modified form in [72].

Model Problem	Algorithm	τ	cond	it	evp	e-avg	fp	fn	acc
	classic	-	-	>300	0	-	-	-	-
Microsection	frugal	-	8.21e4	127	0	112	-	-	-
Problem	adaptive	-	15.86	36	112	-	-	-	-
	ML	0.5	231.37	56	8	30	1	1	0.98
	ML	0.4	16.21	37	24	18	16	0	0.85

Table 8.10: Comparison of classic FETI-DP, frugal FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** for the **three-class model** for **stationary diffusion**; cf. also Fig. 8.20. See Table 8.4 for the column labeling. Additionally, we report the number of edges on which we impose a frugal constraint (e-avg). Table already published in [72, Table 9].

Ten Different Microsection Problems								
Algorithm	τ	**cond**	**it**	**evp**	**e-avg**	**fp**	**fn**	**acc**
adaptive	-	11.04	34.6	112.0	-	-	-	-
		(15.87)	(38)	(112)	-	-	-	-
ML	0.5	147.41	48.8	4.1	43.6	1.7	1.3	0.97
		(271.38)	(58)	(10)	(46)	(3)	(3)	(0.95)
ML	0.4	12.37	34.8	16.0	24.2	10.5	0.0	0.90
		(16.41)	(39)	(24)	(28)	(16)	(0)	(0.85)

Table 8.11: Results for 10 different subsections of a microsection of a dual-phase steel for the **three-class model** for **stationary diffusion**. Comparison of adaptive FETI-DP and ML-FETI-DP for **regular domain decompositions**. We show the average values as well as the maximum values (in brackets). See Table 8.4 for the column labeling. Additionally, we report the number of edges on which we impose a frugal constraint (e-avg). Table already published in [72, Table 10].

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**	**e-avg**	**fp**	**fn**	**acc**
	classic	-	-	>350	0	-	-	-	-
Microsection	frugal	-	-	>350	0	160	-	-	-
Problem	adaptive	-	16.52	35	160	-	-	-	-
	ML	0.5	245.71	56	9	42	2	1	0.98
	ML	0.4	17.82	36	26	31	16	0	0.90

Table 8.12: Comparison of classic FETI-DP, frugal FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **METIS domain decompositions** for the **three-class** model for **stationary diffusion**; cf. also Fig. 8.21. See Table 8.4 for the column labeling. Additionally, we report the number of edges on which we impose a frugal constraint (e-avg). Table already published in [72, Table 11].

Using the ML threshold $\tau = 0.5$, we obtain again only one single false negative edge, which is marked in red; see Fig. 8.21 (left). As for the regular domain decomposition, the false negative edge can be removed by choosing $\tau = 0.4$; see Fig. 8.21 (right). In both cases, ML-FETI-DP is robust, converges fast, and up to 94% of the eigenvalue problems can be saved; see Table 8.12. The behavior does not change for the ten different microsections and we again provide average and maximum values in an aggregated form in Table 8.13.

8.2.4 Comparative results for different sets of training data

So far, all previously presented results for ML-FETI-DP were based on a neural network which has been trained using a specific and carefully designed set of training data. In particular, this training set is based on the coefficient distributions shown in Fig. 8.6

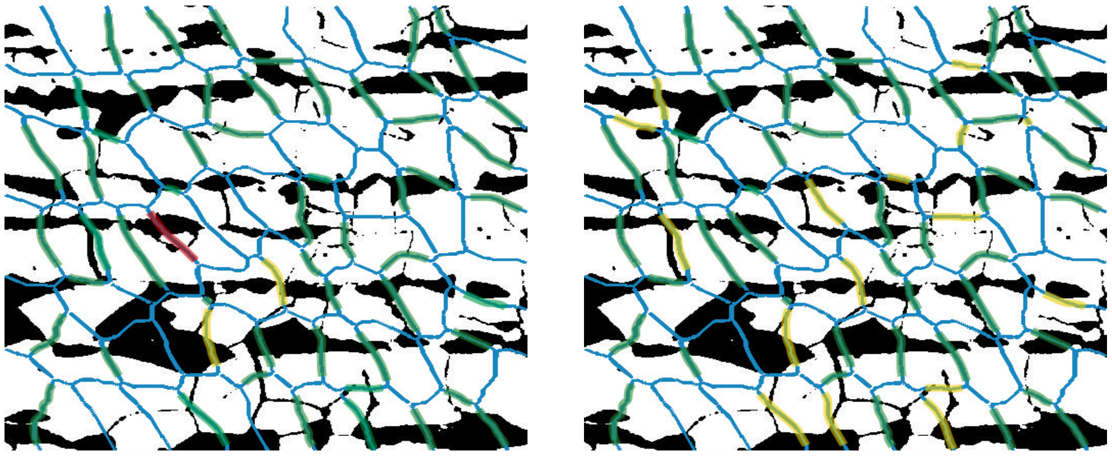


Figure 8.21: Microsection problem for stationary diffusion with marked edges as obtained by ML-FETI-DP for the three-class model for a METIS domain decomposition into 64 subdomains. Correctly assigned edges from class 1 and class 2 are marked in green. Edges from class 1 which are falsely assigned to class 0, and edges from class 2 which are falsely assigned to class 0, are marked in red. Edges from class 0 which are falsely assigned to class 1, and edges from class 1 which are falsely assigned to class 2, are marked in yellow; cf. also Table 8.12. **Left:** ML threshold $\tau = 0.5$. **Right:** ML threshold $\tau = 0.4$. Figure in modified form in [72].

Ten Different Microsection Problems								
Algorithm	τ	**cond**	**it**	**evp**	**e-avg**	**fp**	**fn**	**acc**
adaptive	-	14.81	35.6	160.0	-	-	-	-
		(22.58)	(38)	(160)	-	-	-	-
ML	0.5	233.55	53.2	7.5	46.2	1.6	2.0	0.97
		(256.51)	(57)	(10)	49	(2)	(3)	(0.96)
ML	0.4	15.73	36.8	23.8	28.8	15.8	0.0	0.89
		(24.04)	(40)	(26)	(31)	(21)	(0)	(0.86)

Table 8.13: Results for 10 different subsections of a microsection of a dual-phase steel for the **three-class model** for **stationary diffusion**. Comparison of adaptive FETI-DP and ML-FETI-DP for **METIS domain decompositions**. We show the average values as well as the maximum values (in brackets). See Table 8.4 for the column labeling. Additionally, we report the number of edges on which we impose a frugal constraint (e-avg). Table already published in [72, Table 12].

and is referred to as smart data. However, for three-dimensional problems, the manual construction of training data which ensure satisfactory generalization properties is more complex, especially for irregular domain decompositions. Thus, we additionally test the feasibility of randomly generated training data for the neural network in this section. In particular, we aim to design training data which can easily be generated without any a priori knowledge of the considered model problem. In the following, we investigate and compare the robustness of the resulting hybrid algorithms for four different sets of training and validation data. For all considered choices of training data sets, we provide numerical results for both, stationary diffusion and linear elasticity problems.

Parts of this section have already been published in modified or unmodified form in [74, 76, 77].

First, let us note that for the numerical results presented in this section, we only train the neural network on two regular subdomains sharing a straight edge. Regarding the coefficient functions, we use different sets of coefficient distributions to generate different sets of training data. Despite the varying sets of training data, we use the same setup and the same hyper parameters for the training of the neural network as described in Section 8.1.3.

Besides the already introduced and tested set of smart data, here, we consider random data to train the neural network. Let us note that a completely random coefficient distribution is not appropriate since in this case, coefficient jumps appear at almost all edges. Thus, for almost every edge an eigenvalue problem has to be solved. This yields a neural network which strongly overestimates the number of eigenvalue problems needed and thus leads to a large number of false positive edges in the test data.

Consequently, as a second set of training data, we use a slightly more structured set of randomly generated coefficients with a varying ratio of high and low coefficient values. For the first part of this training set, we randomly generate the coefficient for each pixel, consisting of two triangular finite elements, independently and only control the ratio of high and low coefficient values. Here, we use 30%, 20%, 10%, and 5% of high coefficient values. For the second part, we also control the distribution of the coefficients to a certain degree by randomly generating either horizontal or vertical stripes of a maximum length of four or eight pixels, respectively; see Fig. 8.22 for an exemplary coefficient distribution. Additionally, we generate new coefficient distributions by superimposing pairs of horizontal and vertical coefficient distributions. We refer to this second set of training data as *random data*. Please note that this set of training data can be generated without any a priori knowledge of the underlying model problem.

To generate the output data that are necessary to train the neural network, we solve the eigenvalue problems as described in Section 3.5 for all the aforementioned training and validation configurations. In analogy to the results presented in Sections 8.2.2 and 8.2.3 for the smart training data, we again consider both a two-class classification as well as a three-class classification variant of ML-FETI-DP using the random training data. For all our training and validation data sets, we use a tolerance of $TOL = 100$ to generate the output data for each edge.

In the following, we compare the performance of the ML-FETI-DP algorithm using the different choices of sets of training and validation data to train our neural network.

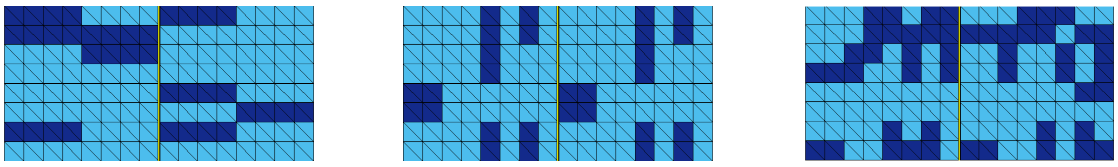


Figure 8.22: Examples of three different randomly distributed coefficient functions in two dimensions obtained by using the same randomly generated coefficient for a horizontal (**left**) or vertical (**middle**) stripe of a maximum length of four finite element pixels, as well as by a pairwise superimposing (**right**). Taken from [74].

In particular, we use the set of 4,500 *smart data* configurations (denoted by *'S'*) and sets of 4,500 and 9,000 *random data* configurations (denoted by *'R1'* and *'R2'*, respectively) each individually as well as a combination of 4,500 *smart* and 4,500 *random data* configurations, which will be denoted by *'SR'*. Let us note that we did not observe a significant improvement for the larger number of 18,000 random data configurations.

First, we present results for the whole set of training data using cross-validation and a fixed ratio of 20% as validation data to test the generalization properties of our neural network. Please note that due to a different heterogeneity of the various training data sets, the accuracy values in Table 8.14 and Table 8.16 are not directly comparable with each other. To mathematically illustrate the varying heterogeneities of the different training data sets, we additionally provide the distribution among the two or three, respectively, different classes for each data set in Table 8.14. However, the provided results in both tables serve as a sanity check to prove that the trained model is able to generate appropriate predictions. With respect to the training data for stationary diffusion problems, the results in terms of accuracy in Table 8.14 show that, besides training the neural network with the set of smart data, also the training with randomly generated coefficient functions as well as with a combination of both training sets leads to a reliable machine learning model. Similarly, the same observation can be made with respect to the training data for linear elasticity problems in Table 8.16. Thus, in both cases, it is reasonable to apply all of the trained models to our test problem in form of microsection subsections.

Second, we use ten different randomly chosen subsections of a microsection of a dual-phase steel as shown in Fig. 8.16 (right) as a test problem for the trained neural networks. In particular, we use exactly the same subsections of the microsection as before. In all presented computations, we consider $\rho_1 = 1e6$ in the black part of the microsection and $\rho_2 = 1$ elsewhere in case of a stationary diffusion problem, and $E_1 = 1e6, E_2 = 1$, and a constant value of $\nu = 0.3$ in case of a linear elasticity problem, respectively. For the discretization of both test problems, we use a regular decomposition of the domain Ω into 8×8 square subdomains and a subdomain size defined by $H/h = 64$. Please note that also other mesh resolutions of the finite element mesh can be used without affecting the accuracy of our classification algorithm as long as the coefficient function is constant on each finite element; cf. also Section 8.1.1.

					class		
training configuration	τ	**fp**	**fn**	**acc**	**0**	**1**	**2**
S, two-class	0.45	8.8%	1.9%	89.2%	67%	33%	-
	0.5	5.4%	5.1%	89.5%			
S, three-class	0.4	5.1%	1.0%	93.9%	67%	20%	13%
	0.5	3.2%	2.3%	94.5%			
R1, two-class	0.45	11.4%	6.7%	81.9%	49%	51%	-
	0.5	8.8%	9.0%	82.2%			
R1, three-class	0.4	9.1%	7.1%	83.8%	49%	39%	12%
	0.5	8.9%	7.0%	84.1%			
R2, two-class	0.45	9.6%	5.3%	85.1%	53%	47%	-
	0.5	7.2%	7.5%	85.3%			
R2, three-class	0.4	10.7%	4.4%	84.9%	53%	28%	19%
	0.5	7.4%	6.9%	85.7%			
SR, two-class	0.45	5.1%	2.1%	92.8%	58%	42%	-
	0.5	3.4%	3.5%	93.1%			
SR, three-class	0.4	5.2%	2.0%	92.8%	58%	29.5%	12.5%
	0.5	4.3%	2.2%	93.5%			

Table 8.14: Comparative results on the different sets of training data for stationary diffusion problems in two dimensions. The numbers are averages over all respective training configurations. We show the ML threshold (τ), the number of false positives (**fp**), the number of false negatives (**fn**), and the accuracy in the classification (**acc**). Table in a slightly modified form already published in [74, Table 1].

For the ten mircosections and the stationary diffusion problem, all four different training data sets result in a robust algorithm when using an ML threshold $\tau = 0.45$; see Table 8.15. For all these approaches, we obtain no false negative edges, which are critical for the convergence of the algorithm. However, the use of 4,500 and 9,000 random data (see R1 and R2) results in a higher number of false positive edges compared to the sole use of 4,500 smart data, which results in a larger number of computed eigenvalue problems. For linear elasticity problems and the ten microsections, the results are fairly comparable; see Table 8.17. Again, the use of the smart training data set provides the best trade-off between robustness and computation cost since we obtain no false negative edges and only 4.8 false positive edges on average. Additionally, also using 9,000 random data (R2) as well as the combined training data set (SR) provides no false negative edges when choosing the ML threshold $\tau = 0.45$. However, for both cases, we have a slightly increased number of false positive edges compared to the smart training data set which results in a slightly higher computational effort.

As a conclusion, we observe that we are able to achieve comparable results when using randomly generated coefficient distributions as training data compared to the manually selected smart data. However, we need a higher number of random data and a slight

Alg.	**T-Data**	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
classic	-	-	-	>300	0	-	-	-
adaptive	-	-	11.0 (15.9)	34.6 (38)	112.0 (112)	-	-	-
ML	S	0.5	8.6e4 (9.7e4)	39.5 (52)	45.0 (57)	1.6 (2)	1.9 (3)	0.97 (0.96)
	S	0.45	11.0 (15.9)	34.6 (38)	46.9 (59)	**4.4 (6)**	**0 (0)**	0.96 (0.94)
	R1	0.5	1.3e5 (1.6e5)	49.8 (52)	43.2 (44)	7.4 (8)	3.8 (4)	0.88 (0.87)
	R1	0.45	11.0 (15.9)	34.6 (38)	53.8 (58)	14.6 (16)	0 (0)	0.86 (0.84)
	R2	0.5	1.5e5 (1.6e5)	50.2 (51)	40.4 (41)	5.6 (6)	3.4 (4)	0.91 (0.89)
	R2	0.45	11.0 (15.9)	34.6 (38)	50.4 (52)	11.2 (12)	0 (0)	0.90 (0.87)
	SR	0.5	9.6e4 (9.8e4)	45.8 (48)	38.2 (39)	1.8 (2)	1.6 (2)	0.96 (0.95)
	SR	0.45	11.0 (15.9)	34.6 (38)	43.4 (44)	4.8 (5)	0 (0)	0.96 (0.94)

Table 8.15: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** for the **two-class model** with different sets of training data, for 10 different subsections of the microsection in Fig. 8.16 (right) for a **stationary diffusion** problem, with $TOL = 100$. Here, training data is denoted as **T-Data**. See Table 8.4 for the column labeling. Table already published in [74, Table 2].

structure in the randomized coefficient distributions to achieve the same accuracy and generalization properties as for the smart data. On the other hand, the advantage of the randomized training data lies in the random generation of the coefficient distributions which is possible without any specific problem-related knowledge. This is especially relevant for the training of a neural network which can be evaluated for three-dimensional domains. Let us also remark that setting up a smart data set in three dimensions is very complicated and we thus prefer a randomly generated set of training data in this case; cf. also Section 8.1.4. Given the relatively low ratio of additional false positive edges compared to the number of saved eigenvalue problems, we can conclude that we can achieve comparable results using the randomized training data when generating a sufficient amount of coefficient configurations.

8.2.5 Experiments on reducing the computational effort for the generation of training data

As already explained in detail in Section 8.1.1, the design of appropriate input and output data is fundamental for the training of a neural network which approximates our specific classification model. In our application within adaptive coarse spaces, we use samples of the coefficient function as input data of the neural network and in Section 8.2.4, we have observed that the choice of the underlying coefficient distributions can have a significant impact on the generalization properties of the trained network. For the different sets of training data presented in Section 8.2.4, in case of regular decompositions of the domain, the resulting sampling points usually cover the complete neighboring subdomains for a specific edge. Even though the training of the neural network as well as the generation of the training and validation data can be performed in an a priori offline-phase, we aim to further reduce the complexity of our approach. In particular, in this section,

	two-class				three-class			
training configuration	τ	**fp**	**fn**	**acc**	τ	**fp**	**fn**	**acc**
S, full sampling	0.45	8.9%	2.7%	88.4%	0.4	5.2%	2.0%	92.8%
	0.5	5.5%	5.6%	88.9%	0.5	3.3%	3.3%	93.4%
R1, full sampling	0.45	12.9%	6.4%	80.7%	0.4	10.7%	8.0%	81.3%
	0.5	8.6%	9.1%	82.3%	0.5	8.9%	9.3%	81.8%
R2, full sampling	0.45	9.9%	5.5%	84.6%	0.4	9.8%	4.8%	85.4%
	0.5	7.0%	7.2%	85.8%	0.5	7.2%	6.4%	86.4%
SR, full sampling	0.45	8.7%	3.1%	88.2%	0.4	6.7%	2.9%	90.4%
	0.5	5.3%	5.4%	89.3%	0.5	4.5%	4.4%	91.1%

Table 8.16: Comparative results on the different sets of training data for linear elasticity problems in two dimensions. The numbers are averages over all respective training configurations. See Table 8.14 for the column labeling. Table in modified form already published in [76, Table 6].

we aim to reduce the size of the input data by using fewer sampling points, i.e., by computing sampling points only in slabs of varying width around a given edge between two subdomains; see also Fig. 8.1 for an exemplary visualization. Let us note that, in principle, our machine learning problem is an image classification task, and thus the approach of reducing the number of sampling points corresponds to the idea of using only a fraction of pixels of the original image as input data for the neural network. In the following, we investigate and compare the accuracy of the resulting machine learning classification model for different widths of the slabs for the sampling points. We provide numerical results for linear elasticity problems and both the two-class and the three-class model.

Parts of this section have already been published in modified or unmodified form in [76, 77].

Let us note that for the numerical results presented in this section, we only train the neural network on two regular subdomains sharing a straight edge. Since the smart training data as well as an increased number of randomized training data showed a comparable performance in Section 8.2.4, here, we focus on the smart training data. Additionally, we focus on the ML thresholds $\tau = 0.45$ and $\tau = 0.4$ for the two-class and the three-class approach, respectively, which led to the most robust results for the full sampling approach. Moreover, as already mentioned, we focus on linear elasticity problems for the following discussion. Despite the different sets of training data, we use the same setup and the same hyper parameters for the training of the neural network as described in Section 8.1.3.

Let us now describe in more detail how we reduce the number of sampling points used as input data for the neural network. In general, for all computed sampling points, we need to determine the corresponding finite element as well as to evaluate the coefficient function for the same finite element. Therefore, there is clearly potential to save resources and compute time in the training as well as in the evaluation phase by reducing the num-

Alg.	**T-Data**	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
classic	-	-	-	>300	0	-	-	-
adaptive	-	-	79.1 (92.8)	87.4 (91)	112.0 (112)	-	-	-
ML	S	0.5	9.3e4 (1.3e5)	92.2 (95)	44.0 (56)	2.2 (3)	2.4 (3)	0.96 (0.95)
	S	0.45	79.1 (92.8)	87.4 (91)	48.2 (61)	**4.8 (7)**	**0 (0)**	0.95 (0.93)
	R1	0.5	1.4e5 (1.6e5)	96.6 (98)	47.2 (48)	7.6 (8)	4.0 (5)	0.90 (0.88)
	R1	0.45	1.7e3 (2.1e4)	90.4 (91)	53.6 (57)	13.4 (16)	0.8 (1)	0.87 (0.86)
	R2	0.5	1.1e5 (1.3e5)	96.2 (97)	46.8 (48)	7.0 (8)	3.6 (5)	0.91 (0.89)
	R2	0.45	79.1 (92.8)	87.4 (91)	52.8 (57)	11.6 (12)	0 (0)	0.90 (0.87)
	SR	0.5	9.7e4 (9.9e4)	94.8 (97)	45.8 (47)	5.8 (6)	3.0 (4)	0.92 (0.93)
	SR	0.45	79.1 (92.8)	87.4 (91)	50.6 (61)	8.8 (10)	0 (0)	0.92 (0.90)

Table 8.17: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** for the **two-class model** with different sets of training data, for 10 different subsections of the microsection in Fig. 8.16 (right) for a **linear elasticity** problem, with $TOL = 100$. See Table 8.4 for the column labeling. Table already published in [77, Table 1].

ber of sampling points used as input data for the neural network. Usually, the coefficient variations close to the edge are the most relevant, i.e., the most critical for the condition number bound of FETI-DP. Therefore, to reduce the total number of sampling points in the sampling grid, reducing the density of the grid points with increasing distance to the edge is a natural approach. More drastically, one could exclusively consider sampling points in a neighborhood of the edge, i.e., on slabs next to the edge. We consider the latter approach here; see also Fig. 8.1 for an illustration of the sampling points inside slabs around an edge $\mathcal{E}_{ij}$. In particular, for subdomains of width H, we consider the cases of sampling in one half and one quarter, i.e., $H/2$ and $H/4$, as well as the extreme case of sampling only inside minimal slabs of the width of one finite element, i.e., h.

For the training data, both sampling in slabs of width $H/2$ and $H/4$ leads to accuracy values which are only slightly lower than for the original full sampling approach; see Table 8.18. In particular, we get slightly higher false positive values, especially for the three-class classification. For the extreme case of sampling only in slabs of width h, i.e., using slabs with the minimal possible width in terms of finite elements, the accuracy value drops from 92.8% for the three-class model to only 68.4% for the threshold $\tau = 0.4$. Note that we did not observe a significant improvement for this sampling strategy for more complex network architectures. Hence, it is questionable if the latter sampling approach still provides a reliable machine learning model.

Subsequently, we apply the different machine learning models to the microsection problem in Fig. 8.16 (middle) using a regular domain decomposition into 8×8 subdomains and $H/h = 64$. We consider $E = 1e3$ in the black part of the microsection and $E = 1$ elsewhere, and set the Poisson ratio to the constant value $\nu = 0.3$. For this test problem, sampling in slabs of width $H/2$ and $H/4$ results in robust algorithms for both the two-class and the three-class model when using the ML threshold $\tau = 0.45$ or $\tau = 0.4$, respectively; see Tables 8.19 and 8.20. For all these approaches, we obtain no false negative edges, which are critical for the convergence of the algorithm. However, the use of fewer

	two-class				three-class			
training configuration	τ	**fp**	**fn**	**acc**	τ	**fp**	**fn**	**acc**
S, full sampling	0.45	8.9%	2.7%	88.4%	0.4	5.2%	2.0%	92.8%
	0.5	5.5%	5.6%	88.9%	0.5	3.3%	3.3%	93.4%
S, sampling in $H/2$	0.45	8.0%	2.6%	89.4%	0.4	9.6%	4.3%	86.1%
	0.5	5.9%	4.0%	90.1%	0.5	7.4%	5.0%	87.6%
S, sampling in $H/4$	0.45	8.2%	2.7%	89.1%	0.4	10.4%	3.9%	85.7%
	0.5	5.7%	4.5%	89.8%	0.5	8.1%	4.8%	87.1%
S, sampling in h	0.45	20.8%	7.5%	71.7%	0.4	22.4%	9.2%	68.4%
	0.5	15.4%	12.9%	72.3%	0.5	15.0%	15.3%	69.7%

Table 8.18: Comparative results on the smart training data set for sampling points in slabs of different widths around the edge for linear elasticity problems in two dimensions. The numbers are averages over all respective training configurations. See Table 8.14 for the column labeling. Table already published in [77, Table 2].

sampling points results in more false positive edges and therefore in a larger number of computed eigenvalue problems. When sampling only in slabs of minimal width h, we do not obtain a robust algorithm for the microsection problem for neither the two-class nor the three-class classification model. This is caused by the existence of a relatively high number of false negative edges.

With respect to our model problems, we can summarize that reducing the effort in the training and evaluation of the neural network by reducing the size of the sampling grid still leads to a robust algorithm. This is also observed during the training and validation of the neural network in terms of high accuracy values. However, we can also conclude that the slab width for the sampling procedure cannot be chosen too small and a sufficient number of finite elements close to the edge have to be covered by the sampling grid.

8.3 Numerical results for adaptive FETI-DP and ML-FETI-DP for three-dimensional test problems

In this section, we provide comparative results for the classic FETI-DP, adaptive FETI-DP, and our hybrid ML-FETI-DP method for three-dimensional problems. Thus, we verify that our proposed approach can also be extended to three-dimensional domains, although this requires some additional effort in the training phase of the neural network; cf. Section 8.1.4. We present numerical results for stationary diffusion problems with different heterogeneous coefficient functions ρ and linear elasticity problems with different distributions for the Young modulus E, respectively. In analogy to the experiments in Section 8.2 for two spatial dimensions, we provide results for both the two-class as well as the three-class model of ML-FETI-DP. For the three-class model in three spatial dimensions, we omit the eigenvalue problem also for faces assigned to class 1 and replace the respective eigenmodes by frugal constraints. Let us recall that for ML-FETI-DP in

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**	**fp**	**fn**	**acc**
	classic	-	-	>300	0	-	-	-
Microsection	adaptive	-	84.72	89	112	-	-	-
Problem	ML, full sampling	0.5	9.46e4	91	41	2	2	0.96
	ML, full sampling	0.45	84.72	89	46	5	0	0.95
	ML, sampling in $H/2$	0.45	84.72	89	47	6	0	0.95
	ML, sampling in $H/4$	0.45	85.31	90	48	7	0	0.94
	ML, sampling in h	0.45	10.9e5	137	50	19	10	0.74

Table 8.19: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** for the **two-class model** with different slab widths for the computation of sampling points, for the microsection subsection in Fig. 8.16 (middle) for a **linear elasticity** problem, with $TOL = 100$. See Table 8.4 for the column labeling. Table already published in [77, Table 3].

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**	**e-avg**	**fp**	**fn**	**acc**
	classic	-	-	>300	0	-	-	-	-
Microsection	adaptive	-	84.72	89	112	-	-	-	-
Problem	ML, full sampling	0.5	274.73	101	15	31	3	2	0.96
	ML, full sampling	0.4	86.17	90	22	24	6	0	0.95
	ML, sampling in $H/2$	0.4	85.29	90	25	26	9	0	0.92
	ML, sampling in $H/4$	0.4	85.37	90	25	27	10	0	0.92
	ML, sampling in h	0.4	2.43e4	111	27	52	29	7	0.68

Table 8.20: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for **regular domain decompositions** for the **three-class model** with different slab widths for the computation of sampling points, for the microsection subsection in Fig. 8.16 (middle) for a **linear elasticity** problem, with $TOL = 100$. See Table 8.4 for the column labeling. Additionally, we report the number of edges on which we impose a frugal constraint (e-avg). Table already published in [77, Table 4].

three spatial dimensions, we apply the neural networks exclusively for the identification of critical faces. Additionally, we solve all eigenvalue problems for edges which belong to more than three subdomains; see also the related explanations in Section 8.1.4 for more details.

In the following experiments, we always use structured tetrahedral meshes of the unit cube constructed from discretizing each voxel of a regular voxel mesh by five piecewise linear tetrahedral finite elements. Let us note that all considered coefficient distributions are chosen to be constant on each voxel. For all our numerical computations, we use the PCG algorithm as the iterative solver. As the stopping criterion for PCG, we use a relative reduction of the preconditioned residual by a factor of $1e$-8. For adaptive FETI-DP, we use the tolerance $TOL = 100$ for the selection of adaptive coarse constraints. In our comparison, we consider both domain decompositions into regular, cubic subdomains as well as irregular domain decompositions obtained by METIS [87]. Please note that the configurations appearing in the numerical experiments in this section are generally not part of our training and validation data set. In particular, we have chosen both different coefficient distributions as well as different combinations of the numbers of finite elements and numbers of subdomains.

Parts of this section have already been published in modified or unmodified form in [75, 76].

8.3.1 Description of the tested coefficient functions

In this section, we briefly describe the two types of discontinuous coefficient distributions which we use for a comparison of the different FETI-DP coarse spaces. First, we consider a stationary diffusion problem with a coefficient function ρ based on five randomly distributed spherical inclusions of different radii in the unit cube; see Fig. 6.12. All five spherical inclusions share the same high coefficient. In particular, we consider the same coefficient distribution which we have already used in the numerical tests for the frugal coarse space in Chapter 6. As indicated in Fig. 6.13, the spherical inclusions cut or touch edges and faces of the domain decomposition interface, which makes this synthetic coefficient distribution relatively hard for an iterative solver, i.e., it leads to a deteriorating rate of convergence for classic coarse spaces. In our computations, all voxels within the five spheres have an identical high coefficient $\rho = 1e6$, whereas the remaining voxels all have a small coefficient $\rho = 1$.

As the second model problem, we consider a linear elastic RVE of a dual-phase steel representing the microstructure of a DP600 steel and obtained by an electronic backscatter diffraction (EBDS) measurement; see Fig. 8.23. This dual-phase steel consists of a martensitic phase and a ferritic phase. In our computations, we use a high Young's modulus in the martensitic phase and a low Young's modulus in the ferritic phase of the material. The most realistic model problem considered here is the case of a coefficient contrast of $1e3$ for the Young modulus E and a constant value for the Poisson ratio $\nu = 0.3$. Let us note that the RVE is part of a larger microstructure which was presented in [23].

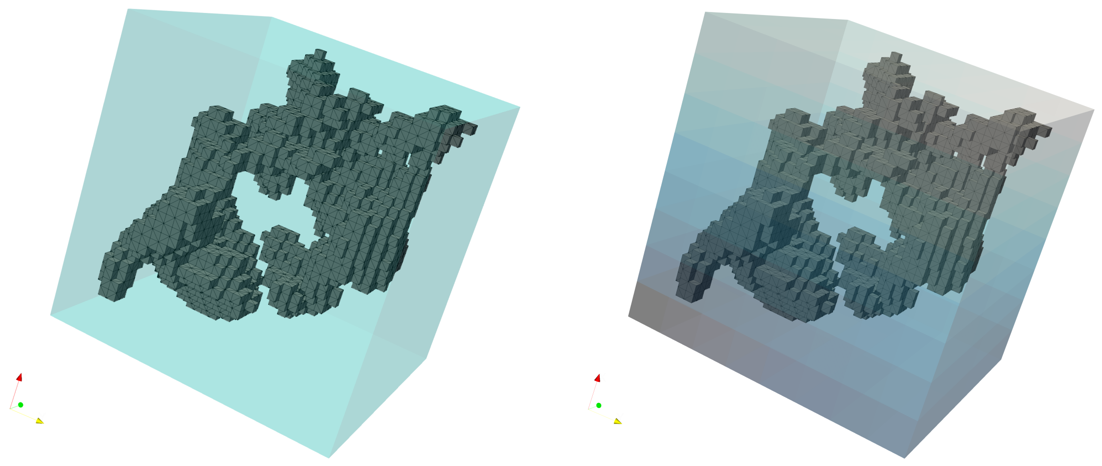


Figure 8.23: Coefficient distribution of an RVE of a dual-phase steel in the unit cube, used for the numerical comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP in three dimensions. The RVE is part of a larger structure presented in [23]. The higher coefficient is $E_1 = 1e3$ (shown in black) and the lower coefficient is $E_2 = 1$, with $\nu = 0.3$ everywhere. **Left:** Discretization with 512 subdomains and $H/h = 4$. **Right:** Visualization for a domain decomposition into $8 \times 8 \times 8$ cubic subdomains, visualized by the different shades of blue, and $H/h = 4$.

8.3.2 Numerical results for the two-class model

Let us first discuss our experiments for the two-class model of ML-FETI-DP. Here, the neural network distinguishes between faces, where the eigenvalue problem results in at least one (in case of stationary diffusion) or six (in case of linear elasticity) additional adaptive constraints and faces, where the eigenvalue is unnecessary. In analogy to Section 8.2, we will refer to the latter case as *negative* or *negative faces* and to the first case as *positive* or *positive faces*. For the adaptive FETI-DP algorithm as well as to obtain the output data for the training of the neural network, we always use a tolerance of $TOL = 100$. For the classification of faces in ML-FETI-DP, we will consider two different values for the ML threshold τ, i.e., $\tau \in \{0.45, 0.5\}$.

8.3.2.1 Regular domain decompositions

Let us first consider the stationary diffusion problem, where the coefficient distribution is given by the spherical inclusions shown in Fig. 6.12. We partition the unit cube into $4\times4\times4$ regular subdomains with the subdomain size defined by $H/h = 10$. In Table 8.21, we compare the iteration counts and condition number estimates for the classic FETI-DP, adaptive FETI-DP, and our hybrid ML-FETI-DP method. Analogously to Section 8.2,

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp$_{\mathcal{F}}$**	**fp**	**fn**	**acc**
	classic	-	-	>350	0	-	-	-
Five	adaptive	-	44.97	63	144	-	-	-
Spheres	ML	0.5	2.73e4	67	7	2	2	0.97
	ML	0.45	44.97	63	12	5	0	0.96

Table 8.21: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **regular domain decomposition** for the **two-class model** for **stationary diffusion** and five spherical inclusions; cf. Fig. 6.12. See Table 8.4 for the column labeling. In particular, here, we denote by evp$_{\mathcal{F}}$ the number of solved eigenvalue problems on faces. Table already published in [75, Table 2].

besides the accuracy of the ML classification using ML-FETI-DP, we also report the number of false negative and false positive faces obtained for two different ML thresholds τ; cf. also the related discussion in Section 8.1 and Section 8.2. Let us note that only false negative faces are critical for the convergence of the ML-FETI-DP method. On the other hand, false positive faces correspond to eigenvalue problems which are solved even though they are not necessary for the robustness of the algorithm since they do not result in any adaptive constraints. Thus, they result in additional computational effort but do not negatively affect the rate of convergence. We can observe from Table 8.21, that we obtain two false negative faces for ML-FETI-DP when using the ML threshold of $\tau = 0.5$. This leads to a worse condition number estimate compared to adaptive FETI-DP, while the iteration number of the algorithm is still satisfactory. By lowering the ML decision threshold to $\tau = 0.45$, we are able to eliminate all false negative faces and thus to correctly identify all critical faces where the eigenvalue problem is necessary. In particular, using our ML-FETI-DP method with an overshooting strategy, we solve only 12 eigenvalue problems on faces in contrast to 144 face eigenvalue problems for the fully adaptive FETI-DP algorithm. Nonetheless, we are still able to retain the same condition number estimate and iteration count as for adaptive FETI-DP. This is indeed a major saving in the number of eigenvalue problems on faces and thus computation time.

We further provide numerical results for the linear elasticity problem using the RVE visualized in Fig. 8.23 as material distribution. Here, we decompose our domain into $8 \times 8 \times 8$ regular subdomains with a reduced subdomain size defined by $H/h = 4$. For completeness, let us recall that for linear elasticity problems, we use a different neural network than for stationary diffusion problems since the correct machine learning labels may differ; cf. Section 8.1. We summarize the comparative results for this model problem in Table 8.22. As for the stationary diffusion problem in Table 8.21, we are able to obtain zero false negative faces and thus can retain a robust algorithm when using the ML threshold of $\tau = 0.45$. Furthermore, we observe that only 66 eigenvalue problems on faces have to be solved for ML-FETI-DP in comparison to 1 344 eigenvalue problems solved for the fully adaptive FETI-DP method.

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**$_{\mathcal{F}}$	**fp**	**fn**	**acc**
	classic	-	-	>350	0	-	-	-
RVE	adaptive	-	16.89	39	1344	-	-	-
Problem	ML	0.5	3.76e4	45	52	10	5	0.98
	ML	0.45	16.89	40	66	19	0	0.98

Table 8.22: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **regular domain decomposition** for the **two-class model** for **linear elasticity** and an RVE; cf. Fig. 8.23. See Table 8.21 for the column labeling. Table already published in [75, Table 3].

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**$_{\mathcal{F}}$	**fp**	**fn**	**acc**
	classic	-	-	>350	0	-	-	-
Five	adaptive	-	30.24	49	288	-	-	-
Spheres	ML	0.5	3.17e4	55	27	5	4	0.97
	ML	0.45	30.25	50	38	12	0	0.96

Table 8.23: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **METIS domain decomposition** for the **two-class model** for **stationary diffusion** and five spherical inclusions; cf. Fig. 6.12. See Table 8.21 for the column labeling. Table already published in [75, Table 4].

8.3.2.2 METIS domain decompositions

As a next set of test problems, we consider irregular domain decompositions obtained by METIS for the same stationary diffusion and linear elasticity problems as in Section 8.3.2.1. Here, we decompose the unit cube into 64 subdomains for the stationary diffusion problem and into 512 subdomains for the linear elasticity problem.

The corresponding results are summarized in Table 8.23 and Table 8.24, respectively. For the stationary diffusion problem, the ML algorithm misses 4 critical faces when using the ML threshold $\tau = 0.5$. However, when lowering the ML threshold to $\tau = 0.45$, we are again able to identify all critical faces. Consequently, we can retain nearly the same convergence behavior as for the adaptive FETI-DP method, while solving only 38 instead of 288 eigenvalue problems; see Table 8.23. For the linear elasticity problem, the results are fairly comparable; see Table 8.24. Again, when using the ML threshold of $\tau = 0.45$, we are able to identify all faces which are critical for the convergence of the algorithm and where the local eigenvalue problem needs to be set up and solved. In this specific case, we only have to solve 92 eigenvalue problems on faces instead of 1 547 for the adaptive FETI-DP approach.

8.3.3 Numerical results for the three-class model

Let us now discuss the results for our extended three-class model in three spatial dimensions. Let us briefly recall from Section 8.1.2, that in the three-class approach of

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp**$_{\mathcal{F}}$	**fp**	**fn**	**acc**
	classic	-	-	>350	0	-	-	-
RVE	adaptive	-	20.13	41	1547	-	-	-
Problem	ML	0.5	3.57e4	47	77	10	6	0.98
	ML	0.45	20.13	41	91	18	0	0.98

Table 8.24: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **METIS domain decomposition** for the **two-class model** for **linear elasticity** and an RVE; cf. Fig. 8.23. See Table 8.21 for the column labeling. Table already published in [75, Table 5].

ML-FETI-DP, we construct frugal face constraints instead of solving an eigenvalue problem for all faces which are classified in class 1 by our neural network; see also Chapter 6 for a detailed description of the respective frugal constraints. Thus, we do not need to solve any eigenvalue problems for the corresponding faces and can omit the respective eigenvalue problems.

As for the two-class model, we always use a tolerance of $TOL = 100$ in the adaptive FETI-DP algorithm as well as to determine the output label for our ML classification of the training data. To provide a direct and fair comparison between the two-class and the three-class model, we consider the same coefficient functions and material distributions, respectively, as in Section 8.3.2. In the following, we consider two different values of the ML threshold τ, i.e., $\tau \in \{0.4, 0.5\}$.

8.3.3.1 Regular domain decompositions

First, we summarize the results for the stationary diffusion problem and five spherical inclusions of a high coefficient as in Fig. 6.12 for classic FETI-DP, adaptive FETI-DP and ML-FETI-DP in Table 8.25. Analogously to the two-class model, we are able to eliminate all false negative faces for the three-class model, when using the ML threshold $\tau = 0.4$. However, in comparison to the two-class model in Table 8.23, we now only need to solve 9, instead of 12, of the original 144 face eigenvalue problems. Thus, due to the computation of the frugal constraints for faces of class 1, we are able to further reduce the number of necessary eigenvalue problems, while retaining a robust algorithm.

Second, we present the respective results for the linear elasticity problem and an RVE in Table 8.26. All in all, the observations with respect to robustness and efficiency of the coarse space are again fairly similar. In this case, we are able to further reduce the number of necessary eigenvalue problems from 66 in Table 8.25 to 32 by using frugal face constraints for all faces classified to class 1.

8.3.3.2 METIS domain decompositions

For the sake of completeness, we finally present numerical results for the three-class model and irregular decompositions of the unit cube obtained by METIS. In this case, we obtain similar results compared to those for regular domain decompositions in Sec-

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
	classic	-	-	>350	0	-	-	-
Five	adaptive	-	44.97	63	144	-	-	-
Spheres	ML	0.5	1.36e4	66	5	4	3	0.95
	ML	0.4	46.77	64	9	13	0	0.91

Table 8.25: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **regular domain decomposition** for the **three-class model** for **stationary diffusion** and five spherical inclusions; cf. Fig. 6.12. See Table 8.21 for the column labeling. Table already published in [75, Table 6].

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
	classic	-	-	>350	0	-	-	-
RVE	adaptive	-	16.89	39	1344	-	-	-
Problem	ML	0.5	4.27e3	44	27	11	5	0.98
	ML	0.4	18.49	40	32	26	0	0.98

Table 8.26: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **regular domain decomposition** for the **three-class model** for **linear elasticity** and an RVE; cf. Fig. 8.23. See Table 8.21 for the column labeling. Table already published in [75, Table 7].

tion 8.3.3.1. In Table 8.27, we present the results for the stationary diffusion problem and the heterogeneous coefficient distribution defined by five spheres with a high coefficient. We observe that, for an appropriate choice of the ML threshold τ, the number of necessary face eigenvalue problems can be further reduced to 19 for $\tau = 0.4$, compared to the respective value of 38 for the two-class model in Table 8.23 and the ML threshold $\tau = 0.45$. Considering the results for the linear elasticity problem and the chosen RVE in Table 8.28 delivers very similar observations. Using the three-class classification model of ML-FETI-DP and $\tau = 0.45$, we only need to solve 45 out of originally 1 547 eigenvalue problems on faces for adaptive FETI-DP, while retaining the robustness and fast convergence rate of the algorithm.

Model Problem	Algorithm	τ	cond	it	evp$_{\mathcal{F}}$	fp	fn	acc
	classic	-	-	>350	0	-	-	-
Five	adaptive	-	30.24	49	288	-	-	-
Spheres	ML	0.5	3.75e4	56	12	8	4	0.96
	ML	0.4	33.52	50	19	15	0	0.95

Table 8.27: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **METIS domain decomposition** for the **three-class model** for **stationary diffusion** and five spherical inclusions; cf. Fig. 6.12. See Table 8.21 for the column labeling. Table already published in [75, Table 8].

Model Problem	**Algorithm**	τ	**cond**	**it**	**evp$_\mathcal{F}$**	**fp**	**fn**	**acc**
	classic	-	-	>350	0	-	-	-
RVE	adaptive	-	20.13	41	1547	-	-	-
Problem	ML	0.5	4.12e4	47	28	11	5	0.98
	ML	0.4	24.22	42	45	28	0	0.98

Table 8.28: Comparison of classic FETI-DP, adaptive FETI-DP, and ML-FETI-DP for a **METIS domain decomposition** for the **three-class model** for **linear elasticity** and an RVE; cf. Fig. 8.23. See Table 8.21 for the column labeling. Table already published in [75, Table 9].

8.4 Some summarizing remarks regarding the computational effort

In this chapter, we have introduced a hybrid approach which combines adaptive FETI-DP methods with techniques from machine learning to design robust and efficient coarse spaces. The aim of the proposed technique was to identify critical edges or faces of the domain decomposition which are responsible for outliers of the spectrum of the preconditioned system. In general, for these edges, problem-dependent constraints are necessary to design a robust algorithm. We have numerically tested this approach both in two and three spatial dimensions for a range of different synthetic and practically inspired problems. In all considered cases, the number of necessary eigenvalue problems in an adaptive FETI-DP method could be reduced significantly, while almost the same iteration counts and condition numbers could be maintained. Additionally, we have proposed and tested an extended three-class classification method, which unites the idea of ML-FETI-DP with the heuristic frugal coarse space from Chapter 6. In particular, by using frugal constraints as a low-dimensional approximation of adaptive constraints for certain selected edges or faces, we are able to further reduce the number of necessary eigenvalue problems in an adaptive FETI-DP method. Let us recall that usually, in a parallel implementation, the setup and the solution of the eigenvalue problems as well as the computation of the required local Schur complements take up a significant part of the total time to solution. Taking this observation into account, the proposed ML-FETI-DP method can potentially reduce the computational effort and the expected overall time to solution. Furthermore, computing only adaptive constraints on selected equivalence classes instead of categorically for all edges or faces of a domain decomposition, fulfills the desire to construct a preferably small but robust coarse space which also increases the parallel potential of the obtained iterative solver.

Of course, to enable a fair evaluation of the proposed ML-FETI-DP method compared to classic, frugal, or completely adaptive coarse spaces, we cannot neglect the time and the computational effort which is needed for the setup, i.e., the training, and the evaluation of the neural networks. First, let us recall that the training and validation of the neural network can be completely conducted in an a priori offline-phase. This includes the generation of training and validation data as well as the iterative process of training the neural network, i.e., the iterative optimization of the network parameters. Here, the main idea is to train a network once for a class of model problems and subsequently apply it to numerous computations based on this model problem, e.g., stationary diffusion problems with many different diffusion coefficient functions. Note that we explicitly designed our procedure such that it is independent of the underlying finite element discretization. Even though the approach is not advocated for the solution of a single problem, we have to consider that our neural networks are relatively small. Thus, the required time to train the neural networks is expected to be low when using optimized code and appropriate hardware, e.g., GPUs. Hence, we believe that the proposed method already amortizes for a small number of different problems.

Second, we have to consider the time which is needed for the evaluation of the already trained and saved network for previously unseen test problems. Let us recapitulate that as input data, i.e., features of the neural network, we use function evaluations of the coefficient function of the underlying PDE. In our context, we refer to these input data as sampling points. Consequently, for the generation of the input data for a given test problem, we have to compute the geometric location of the sampling points as well as to determine the associated finite elements. The computation of the geometric location of the sampling points only requires scalar operations and is thus negligible. To determine the corresponding finite element index for all sampling points, efficient search algorithms can be used, as, e.g., a binary search which results in a logarithmic runtime depending on the number of computed sampling points. Moreover, we have shown in Section 8.2.5 that for two-dimensional problems, it is possible to reduce the number of sampling points to a relatively close surrounding of an edge without severly impairing the accuracy of the ML classification, especially for the two-class model. The same concept can analogously be applied to three-dimensional test problems and further reduces the required computational effort needed for the classification of the considered equivalence class. On top of that, as already mentioned, our applied networks are rather shallow which makes their evaluation cheap once the sampling points have been computed.

Finally, let us formulate a general conclusion with respect to the different modifications or variants of the described ML-FETI-DP method. Given that our predominant goal is to design a coarse space which shows a stable rate of convergence for highly heterogeneous coefficient distributions and is - at the same time - preferably small, we highly recommend the three-class model for most practical applications. In this case, a potentially higher number of unnecessary eigenvalue problems can be saved due to the introduction of an additional, third class of edges or faces, respectively, for which we replace the eigenvalue problem by the corresponding frugal constraints. Furthermore, we would generally use a training and validation data set based on randomized coefficient distributions. In particular, these data can be generated without any a priori knowledge of the considered class of test problems and are applicable both for regular and irregular decompositions in two and three spatial dimensions.

To conclude this chapter, let us note again that our present implementation uses Python and Tensorflow [1] for the training of the different neural networks while the adaptive FETI-DP method is implemented in MATLAB [134]. Thus, it is hard or unreliable, respectively, to compare the actual times to solution for ML-FETI-DP and adaptive

FETI-DP based on the current implementations. To be able to obtain a final evaluation of this question, we plan to integrate the described hybrid approach, using adaptive constraints only for certain equivalence classes as predicted by the machine learning algorithm, into our parallel software of BDDC; cf. also Chapter 9.

9 Conclusion and Future Work

In this thesis, we have presented and numerically investigated several approximate and adaptive coarse spaces for the FETI-DP and the BDDC domain decomposition methods. For the BDDC method, we have considered three different approximate coarse spaces in a common framework, using our parallel BDDC software based on PETSc [8, 9]. With respect to parallel scalability, the considered three-level BDDC method as well as the approximate BDDC preconditioner based on AMG showed the largest parallel potential. In particular, due to their multilevel structure, these methods only require the exact solution of a coarse problem on the coarsest level, which usually is of a much smaller size than the original coarse problem. Of course, as a drawback, the approximate solution of the coarse problem results in higher condition number estimates compared to classic BDDC methods. This also supports the theoretical upper bounds of the condition number of the different methods which we have compared within a common framework.

Furthermore, we have presented a frugal coarse space for both the BDDC and the FETI-DP method. The frugal coarse space is a heuristic approach and can be interpreted as a low-dimensional approximation of a certain adaptive method. The computation of the frugal coarse space is computationally cheap and does not require the solution of any eigenvalue problems on parts of the domain decomposition interface. In our MATLAB experiments, the resulting frugal FETI-DP coarse space has shown very promising results for different heterogeneous model problems in both two and three spatial dimensions. In particular, we have presented a heuristic strategy to further reduce the size of the frugal FETI-DP coarse space, based on estimates of the energy related to the computed frugal constraints. Thus, it is possible to further increase the parallel potential of the respective method while simultaneously maintaining its robust rate of convergence. Additionally, we have implemented and compared the frugal coarse space in the same parallel BDDC implementation as the previously mentioned approximate BDDC preconditioners. In the related parallel experiments, we have observed that the frugal coarse space is clearly more general and thus more robust than classic BDDC coarse spaces, especially for realistic coefficient distributions. Hence, we suggest to establish the presented frugal coarse space as a new default approach given its superiority compared to classic coarse spaces and its relatively low computational effort.

On top of that, we have discussed a hybrid approach, which combines the concept of adaptive coarse spaces with techniques from deep learning. Here, we have trained neural networks to make an automated prediction in a preprocessing phase, for which edges or faces of a domain decomposition, adaptive constraints are necessary to retain a robust solver. We have applied this approach to a specific adaptive FETI-DP method which relies on the solution of certain localized eigenvalue problems on the respective equivalence classes between two neighboring subdomains. Let us recall that in our experiments with regard to frugal FETI-DP and BDDC, we have observed that also the frugal coarse space might deteriorate in convergence for completely arbitrary or highly complex coefficient distributions, even though the iteration numbers are still satisfactory. In this case, adaptive, i.e., problem-dependent coarse spaces are necessary for which we can prove a condition number estimate which is independent of the coefficient contrast. However, as a drawback, the setup and the solution of the respective eigenvalue problems are usually computationally expensive. Additionally, for many real-world problems, a high number of the eigenvalue problems are actually unnecessary since they do not result in any additional constraints. In this thesis, we have presented a modified approach, where we only set up and solve the eigenvalue problems for edges or faces, which are classified as critical by the neural network. We have tested the proposed approach for stationary diffusion and linear elasticity problems in both, two and three spatial dimensions. In both cases, we have numerically shown the stability and the robustness of the resulting ML-FETI-DP coarse space. In particular, in the two-dimensional case, we were able to save up to 94 % of the eigenvalue problems for realistic coefficient functions obtained from a microsection of a dual-phase steel. For the three-dimensional case, we were able to save up to 97 % of the eigenvalue problems on faces in the best case. Let us note that the presented results are based on our MATLAB [134] implementations of the considered adaptive FETI-DP method. For the training and subsequent evaluation of the neural networks, we have used Python and Tensorflow [1].

Moreover, we have also proposed an extended three-class variant of ML-FETI-DP which combines the described method with our frugal constraints. Here, we further distinguish between a third class of edges or faces, respectively, where we also omit the eigenvalue problem and instead implement the respective frugal constraints. As a result, the number of necessary eigenvalue problems can be further reduced and thus the related computational effort of the adaptive FETI-DP method. Hence, we would recommend the three-class classification for most practical model problems compared to the two-class approach.

Similarly as we were able to combine our frugal constraints with the concept of ML-FETI-DP, it is also possible to unite other combinations of the presented approximate and adaptive coarse spaces. For instance, we have shown that it is possible to use an AMG method for the approximate solution of a three-dimensional frugal coarse problem for BDDC. We have observed that the resulting algorithm is again robust and thus suggest that combinations of the presented approximate coarse spaces can be used to further increase the parallel scalability of FETI-DP and BDDC.

As a further approximate coarse space, we propose to combine the frugal coarse space with the presented three-level BDDC preconditioner, which are both integrated into our parallel BDDC implementation. This would include the construction of frugal constraints both on the level of the subdomains as well as on the level of the subregions. Alternatively, it is also possible to combine frugal constraints on the second level with adaptive constraints on the third level. This would lead to smaller eigenvalue problems which have to be set up and solved on the coarsest level, compared to a standard adaptive BDDC method. A numerical investigation and comparison of both variants with respect to robustness and parallel scalability is an interesting topic of future research.

Given that our results for ML-FETI-DP are, so far, based on our MATLAB implementation of the respective FETI-DP coarse space, we plan to implement the related adaptive BDDC method in our parallel software framework as well as to integrate the considered deep learning-based classifications, i.e., computing adaptive constraints only for certain equivalence classes as predicted by the neural network. Especially, this would enable us to finally evaluate the question, how much time to solution and computational effort is actually saved for the hybrid ML approach compared to adaptive FETI-DP or BDDC, respectively.

Finally, we also plan to predict the adaptive constraints themselves for a given heterogeneous coefficient distribution in the near future. Here, we could benefit from our work involving ML-FETI-DP in order to provide an approach which is independent of the underlying finite element discretization and is applicable both to regular as well as irregular domain decompositions.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from `https://www.tensorflow.org/`.

[2] S. Badia, A. F. Martín, and H. Nguyen. Physics-based balancing domain decomposition by constraints for multi-material problems. *SIAM J. Sci. Comput.*, 79(2):718–747, 2019.

[3] S. Badia, A. F. Martín, and J. Principe. On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections. *Parallel Comput.*, 50:1–24, 2015.

[4] S. Badia, A. F. Martín, and J. Principe. Multilevel Balancing Domain Decomposition at Extreme Scales. *SIAM J. Sci. Comput.*, 38(1):C22–C52, 2016.

[5] A. H. Baker, A. Klawonn, T. Kolev, M. Lanser, O. Rheinbach, and U. M. Yang. Scalability of Classical Algebraic Multigrid for Elasticity to half a Million Parallel Tasks. In *Software for Exascale Computing—SPPEXA 2013–2015*, volume 113 of *Lect. Notes Comput. Sci. Eng.*, pages 113–140. Springer, 2016.

[6] A. H. Baker, T. V. Kolev, and U. M. Yang. Improving algebraic multigrid interpolation operators for linear elasticity problems. *Numer. Linear Algebra Appl.*, 17(2-3):495–517, 2010.

[7] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, and K. Willcox. Brochure on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence. USDOE Office of Science (SC) (United States). 2018.

[8] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018.

[9] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2018.

[10] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries. In *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[11] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini. Isogeometric BDDC preconditioners with deluxe scaling. *SIAM J. Sci. Comput.*, 36(3):A1118–A1139, 2014.

[12] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini. Adaptive selection of primal constraints for isogeometric BDDC deluxe preconditioners. *SIAM J. Sci. Comput.*, 39(1):A281–A302, 2017.

[13] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini. Parallel sum primal spaces for isogeometric deluxe BDDC preconditioners. In *Domain decomposition methods in science and engineering XXIII*, volume 116 of *Lect. Notes Comput. Sci. Eng.*, pages 17–29. Springer, Cham, 2017.

[14] A. Ben-Israel and T. N. Greville. *Generalized inverses: theory and applications*, volume 15. Springer Science & Business Media, 2003. 2nd edition.

[15] A. S. Berahas and M. Takáč. A robust multi-batch L-BFGS method for machine learning. *Optimization Methods and Software*, 35(1):191–219, 2020.

[16] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] P. E. Bjørstad, J. Koster, and P. Krzyżanowski. Domain decomposition solvers for large scale industrial finite element problems. In *PARA2000 Workshop on Applied Parallel Computing*, pages 373–383. Lecture Notes in Computer Science 1947, Springer-Verlag, 2000.

[18] R. Bollapragada, J. Nocedal, D. Mudigere, H.-J. Shi, and P. T. P. Tang. A progressive batching L-BFGS method for machine learning. In *International Conference on Machine Learning*, pages 620–629. PMLR, 2018.

[19] L. Bottou and O. Bousquet. 13 The Tradeoffs of Large-Scale Learning. *Optimization for machine learning*, page 351, 2011.

[20] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[21] J.-F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu. Variational Formulation and Algorithm for Trace Operator in Domain Decomposition Calculations.

In *Domain Decomposition Methods (Los Angeles, CA, 1988)*, pages 3–16, Philadelphia, PA, 1989. SIAM.

[22] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* Springer-Lehrbuch Masterclass. Springer, Berlin-Heidelberg-New York, 5. edition, 2013.

[23] D. Brands, D. Balzani, L. Scheunemann, J. Schröder, H. Richter, and D. Raabe. Computational Modeling of Dual-Phase Steels Based on Representative Three-Dimensional Microstructures Obtained from EBSD Data. *Arch. Appl. Mech.*, 86(3):575–598, 2016.

[24] S. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods.* Texts in Applied Mathematics. Springer New York, 3. edition, 2008.

[25] S. C. Brenner and L.-Y. Sung. BDDC and FETI-DP without matrices or vectors. *Computer methods in applied mechanics and engineering*, 196(8):1429–1435, 2007.

[26] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A Stochastic Quasi-Newton Method for Large-Scale Optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[27] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[28] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156, 1994.

[29] J. G. Calvo and O. B. Widlund. An adaptive choice of primal constraints for BDDC domain decomposition algorithms. *Electron. Trans. Numer. Anal.*, 45:524–544, 2016.

[30] F. Chollet. *Deep learning with Python.* Simon and Schuster, 2017.

[31] P. G. Ciarlet. *Mathematical Elasticity, Volume I: Three-Dimensional Elasticity.* Elsevier Science, 1988.

[32] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

[33] J.-M. Cros. A preconditioner for the Schur complement domain decomposition method. In *Proceedings of the 14th International Conference on Domain Decomposition Methods in Science and Engineering*, pages 373–380. National Autonomous University of Mexico (UNAM), Mexico City, Mexico, 2003.

[34] H. De Sterck, R. D. Falgout, J. W. Nolting, and U. M. Yang. Distance-two interpolation for parallel algebraic multigrid. *Numerical Linear Algebra with Applications*, 15(2-3):115–139, 2008.

[35] H. De Sterck, U. M. Yang, and J. J. Heys. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1019–1039, 2006.

[36] F. dell'Isola, M. Sofonea, and D. Steigmann. *Mathematical Modelling in Solid Mechanics*. Advanced Structured Materials. Springer, Singapore, 2017.

[37] C. Dohrmann and C. Pechstein. Modern domain decomposition solvers - BDDC, deluxe scaling, and an algebraic approach. 2013. Slides for a talk at the NuMa Seminar, JKU Linz, December 10th, 2013. Online available at https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[38] C. R. Dohrmann. A preconditioner for substructuring based on constrained energy minimization. *SIAM J. Sci. Comput.*, 25(1):246–258, 2003.

[39] C. R. Dohrmann. An approximate BDDC preconditioner. *Numer. Linear Algebra Appl.*, 14(2):149–168, 2007.

[40] C. R. Dohrmann, K. H. Pierson, and O. B. Widlund. On Inexact Solvers for the Coarse Problem of BDDC. In *Domain decomposition methods in science and engineering XXV*, volume 138 of *Lect. Notes Comput. Sci. Eng.*, pages 117–124. Springer, Cham, 2018.

[41] C. R. Dohrmann, K. H. Pierson, and O. B. Widlund. Vertex-based preconditioners for the coarse problems of BDDC. *SIAM Journal on Scientific Computing*, 41(5):A3021–A3044, 2019.

[42] C. R. Dohrmann and O. B. Widlund. Some recent tools and a BDDC algorithm for 3D problems in H(curl). In *Domain decomposition methods in science and engineering XX*, volume 91 of *Lect. Notes Comput. Sci. Eng.*, pages 15–25. Springer, Heidelberg, 2013.

[43] C. R. Dohrmann and O. B. Widlund. A BDDC Algorithm with Deluxe Scaling for Three-Dimensional H(curl) Problems. *Communications on Pure and Applied Mathematics*, 69(4):745–770, 2016.

[44] V. Dolean, F. Nataf, R. Scheichl, and N. Spillane. Analysis of a two-level Schwarz method with coarse spaces based on local Dirichlet-to-Neumann maps. *Comput. Methods Appl. Math.*, 12(4):391–414, 2012.

[45] Z. Dostál. Conjugate gradient method with preconditioning by projector. *International Journal of Computer Mathematics*, 23(3-4):315–323, 1988.

[46] M. Dryja and O. B. Widlund. Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems. *Communications on pure and applied mathematics*, 48(2):121–155, 1995.

[47] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7):2121–2159, 2011.

[48] Y. Efendiev, J. Galvis, R. Lazarov, and J. Willems. Robust domain decomposition preconditioners for abstract symmetric positive definite bilinear forms. *ESAIM Math. Model. Numer. Anal.*, 46(5):1175–1199, 2012.

[49] L. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2nd edition, 2010.

[50] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. 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(7):1523–1544, 2001.

[51] C. Farhat, M. Lesoinne, and K. 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).

[52] C. Farhat and F.-X. Roux. A method of Finite Element Tearing and Interconnecting and its parallel solution algorithm. *Int. J. Numer. Meth. Engrg.*, 32:1205–1227, 1991.

[53] J. Galvis and Y. Efendiev. Domain decomposition preconditioners for multiscale flows in high-contrast media. *Multiscale Modeling & Simulation*, 8(4):1461–1483, 2010.

[54] J. Galvis and Y. Efendiev. Domain decomposition preconditioners for multiscale flows in high contrast media: reduced dimension coarse spaces. *Multiscale Modeling & Simulation*, 8(5):1621–1644, 2010.

[55] M. J. Gander, A. Loneland, and T. Rahman. Analysis of a new harmonically enriched multiscale coarse space for domain decomposition methods. *arXiv preprint arXiv:1512.05285*, pages 1–21, 2015.

[56] R. V. Garimella, M. J. Shashkov, and P. M. Knupp. Triangular and quadrilateral surface mesh quality optimization using local parametrization. *Computer Methods in Applied Mechanics and Engineering*, 193(9-11):913–928, 2004.

[57] S. Gippert. *Domain decomposition methods for elastic materials with compressible and almost incompressible components*. PhD thesis, Universität Duisburg-Essen, 2013. Online available at https://duepublico2.uni-due.de/receive/duepublico_mods_00030084.

[58] S. Gippert, A. Klawonn, and O. 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.

[59] S. Gippert, A. Klawonn, and O. Rheinbach. A deflation based coarse space in dual-primal FETI methods for almost incompressible elasticity. In *Numerical mathematics and advanced applications—ENUMATH 2013*, volume 103 of *Lect. Notes Comput. Sci. Eng.*, pages 573–581. Springer, Cham, 2015.

[60] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth int. conf. on artificial intelligence and statistics*, pages 315–323, 2011.

[61] P. Goldfeld, L. F. Pavarino, and O. B. Widlund. Balancing Neumann-Neumann preconditioners for mixed approximations of heterogeneous problems in linear elasticity. *Numerische Mathematik*, 95(2):283–324, 2003.

[62] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press Cambridge, 2016.

[63] P. Gosselet, C. Rey, and D. J. Rixen. On the initial estimate of interface forces in FETI methods. *Computer Methods in Applied Mechanics and Engineering*, 192(25):2749–2764, 2003.

[64] A. Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

[65] W. D. Gropp, S. Huss-Lederman, and A. Lumsdaine. *MPI-The Complete Reference, Volume 2: The MPI Extensions*. MIT Press, Cambridge, MA, USA, 1998.

[66] S. Haykin. *Neural networks and learning machines*. Pearson Education India, 3. edition, 2010.

[67] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[68] A. Heinlein. *Parallel Overlapping Schwarz Preconditioners and Multiscale Discretizations with Applications to Fluid-Structure Interaction and Highly Heterogeneous Problems*. Phd thesis, University of Cologne, 2016. Online available at http://kups.ub.uni-koeln.de/id/eprint/6841.

[69] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach. Multiscale coarse spaces for overlapping Schwarz methods based on the ACMS space in 2D. *Electronic Transactions on Numerical Analysis (ETNA)*, 48:156–182, 2018.

[70] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach. Adaptive GDSW coarse spaces for overlapping Schwarz methods in Three Dimensions. *SIAM Journal on Scientific Computing*, 41(5):A3045–A3072, 2019.

[71] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Predicting the geometric location of critical edges in adaptive GDSW overlapping domain decomposition methods using deep learning. TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. 2021-2. https://kups.ub.uni-koeln.de/id/eprint/36257. Accepted for publication in the proceedings of the International Conference on Domain Decomposition Methods 26, Springer LNCSE, August 2021.

[72] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Machine Learning in Adaptive Domain Decomposition Methods - Predicting the Geometric Location of Constraints. *SIAM J. Sci. Comput.*, 41(6):A3887–A3912, 2019.

[73] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. A Frugal FETI-DP and BDDC Coarse Space for Heterogeneous Problems. *Electr. Trans. Numer. Anal.*, 53:562–591, 2020.

[74] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Machine Learning in Adaptive FETI-DP - A Comparison of Smart and Random Training Data. In *Domain decomposition methods in science and engineering XXV*, volume 138 of *Lect. Notes Comput. Sci. Eng.*, pages 218–226. Springer, 2020.

[75] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Combining Machine Learning and Adaptive Coarse Spaces - A Hybrid Approach for Robust FETI-DP Methods in Three Dimensions. *SIAM J. Sci. Comput.*, 43(5):S816–S838, 2021. Special Section Copper Mountain 2020.

[76] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Combining Machine Learning and Domain Decomposition Methods for the Solution of Partial Differential Equations–A Review. *GAMM-Mitt.*, 44(1):e202100001, 2021.

[77] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Machine Learning in Adaptive FETI-DP - Reducing the Effort in Sampling. In *Numerical Mathematics and Advanced Applications ENUMATH 2019*, volume 139 of *Lect. Notes Comput. Sci. Eng.*, pages 593–603. Springer, 2021.

[78] A. Heinlein, M. J. Kühn, and A. Klawonn. Local spectra of adaptive domain decomposition methods. In *Domain decomposition methods in science and engineering XXV*, volume 138 of *Lect. Notes Comput. Sci. Eng.*, pages 167–175. Springer, Cham, 2020.

[79] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[80] V. E. Henson and U. M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41:155–177, 2002.

[81] M. R. Hestenes. The conjugate gradient method for solving linear systems. In *Proc. Symp. Appl. Math VI, American Mathematical Society*, pages 83–102, 1956.

[82] G. Hinton, N. Srivastava, and K. Swersky. Lecture 6a: Overview of mini-batch gradient descent. Neural Networks for Machine Learning (Coursera Lecture Slides). 2012. Online available at https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[83] G. Hinton, N. Srivastava, and K. Swersky. Lecture 6e: RMSProp - Divide the gradient by a running average of its recent magnitude. Neural Networks for Machine Learning (Coursera Lecture Slides). 2012. Online available at https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[84] G. A. Holzapfel. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. Wiley, 2000.

[85] M. Jarošová, A. Klawonn, and O. Rheinbach. Projector preconditioning and transformation of basis in FETI-DP algorithms for contact problems. *Math. Comput. Simulation*, 82(10):1894–1907, 2012.

[86] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proceedings of the IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[87] G. Karypis and V. Kumar. METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. http://www.cs.umn.edu/~metis, 2009.

[88] H. H. Kim, E. Chung, and J. Wang. BDDC and FETI-DP preconditioners with adaptive coarse spaces for three-dimensional elliptic problems with oscillatory and high contrast coefficients. *Journal of Computational Physics*, 349:191–214, 2017.

[89] H. H. Kim and E. T. Chung. A BDDC algorithm with enriched coarse spaces for two-dimensional elliptic problems with oscillatory and high contrast coefficients. *Multiscale Model. Simul.*, 13(2):571–593, 2015.

[90] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[91] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[92] A. Klawonn, M. Kühn, and O. Rheinbach. Adaptive coarse spaces for FETI-DP in three dimensions. *SIAM J. Sci. Comput.*, 38(5):A2880–A2911, 2016.

[93] A. Klawonn, M. Kühn, and O. Rheinbach. Adaptive coarse spaces for FETI-DP in three dimensions with applications to heterogeneous diffusion problems. In *Domain decomposition methods in science and engineering XXIII*, volume 116 of *Lect. Notes Comput. Sci. Eng.*, pages 187–196. Springer, Cham, 2017.

[94] A. Klawonn, M. Kühn, and O. Rheinbach. Adaptive FETI-DP and BDDC methods with a generalized transformation of basis for heterogeneous problems. *Electron. Trans. Numer. Anal.*, 49:1–27, 2018.

[95] A. Klawonn, M. Kühn, and O. Rheinbach. Coarse spaces for FETI-DP and BDDC methods for heterogeneous problems: connections of deflation and a generalized transformation-of-basis approach. *Electron. Trans. Numer. Anal.*, 52:43–76, 2020.

[96] A. Klawonn, M. J. Kühn, and O. Rheinbach. Parallel adaptive FETI-DP using lightweight asynchronous dynamic load balancing. *International Journal for Numerical Methods in Engineering*, 121(4):621–643.

[97] A. Klawonn, M. Lanser, P. Radtke, and O. Rheinbach. On an adaptive coarse space and on nonlinear domain decomposition. In *Domain decomposition methods in science and engineering XXI*, volume 98 of *Lect. Notes Comput. Sci. Eng.*, pages 71–83. Springer, 2014.

[98] A. Klawonn, M. Lanser, and O. Rheinbach. An efficient approach to implement arbitrary coarse spaces in nonlinear and linear FETI-DP and BDDC methods. In Preparation.

[99] A. Klawonn, M. Lanser, and O. Rheinbach. Toward Extremely Scalable Nonlinear Domain Decomposition Methods for Elliptic Partial Differential Equations. *SIAM J. Sci. Comput.*, 37(6):C667–C696, 2015.

[100] A. Klawonn, M. Lanser, and O. Rheinbach. A highly scalable implementation of inexact nonlinear FETI-DP without sparse direct solvers. In *Numerical mathematics and advanced applications ENUMATH 2015*, volume 112 of *Lect. Notes Comput. Sci. Eng.*, pages 255–264. Springer, Cham, 2016.

[101] A. Klawonn, M. Lanser, and O. Rheinbach. Nonlinear BDDC Methods with Approximate Solvers. *Electron. Trans. Numer. Anal.*, 49:pp. 244–273, 2018.

[102] A. Klawonn, M. Lanser, O. Rheinbach, and M. Uran. Nonlinear FETI-DP and BDDC methods: a unified framework and parallel results. *SIAM J. Sci. Comput.*, 39(6):C417–C451, 2017.

[103] A. Klawonn, M. Lanser, O. Rheinbach, and J. Weber. Preconditioning the coarse problem of BDDC methods - Three-level, algebraic multigrid, and vertex-based preconditioners. *Electron. Trans. Numer. Anal.*, 51:432–450, 2019.

[104] A. Klawonn, M. Lanser, O. Rheinbach, G. Wellein, and M. Wittmann. Energy efficiency of nonlinear domain decomposition methods. *The International Journal of High Performance Computing Applications*, 35(3):237–253, 2021.

[105] A. Klawonn, L. F. Pavarino, and O. Rheinbach. Spectral element FETI-DP and BDDC preconditioners with multi-element subdomains. *Computer methods in applied mechanics and engineering*, 198(3-4):511–523, 2008.

[106] A. Klawonn, P. Radtke, and O. Rheinbach. FETI-DP with different scalings for adaptive coarse spaces. *PAMM*, 14(1):835–836, 2014.

[107] A. Klawonn, P. Radtke, and O. Rheinbach. FETI-DP methods with an adaptive coarse space. *SIAM J. Numer. Anal.*, 53(1):297–320, 2015.

[108] A. Klawonn, P. Radtke, and O. Rheinbach. A comparison of adaptive coarse spaces for iterative substructuring in two dimensions. *Electron. Trans. Numer. Anal.*, 45:75–106, 2016.

[109] A. Klawonn and O. Rheinbach. A parallel Implementation of Dual-Primal FETI Methods for Three-Dimensional Linear Elasticity Using a Transformation of Basis. *SIAM J. Sci. Comput.*, 28(5):1886–1906, 2006.

[110] A. Klawonn and O. Rheinbach. Inexact FETI-DP methods. *Internat. J. Numer. Methods Engrg.*, 69(2):284–307, 2007.

[111] A. Klawonn and O. Rheinbach. Robust FETI-DP methods for heterogeneous three dimensional elasticity problems. *Comput. Methods Appl. Mech. Engrg.*, 196(8):1400–1414, 2007.

[112] A. Klawonn and O. Rheinbach. Highly scalable parallel domain decomposition methods with an application to biomechanics. *ZAMM Z. Angew. Math. Mech.*, 90(1):5–32, 2010.

[113] A. Klawonn and O. Rheinbach. Deflation, projector preconditioning, and balancing in iterative substructuring methods: connections and new results. *SIAM J. Sci. Comput.*, 34(1):A459–A484, 2012.

[114] A. Klawonn, O. Rheinbach, and O. B. Widlund. An analysis of a FETI-DP algorithm on irregular subdomains in the plane. *SIAM J. Numer. Anal.*, 46(5):2484–2504, 2008.

[115] A. Klawonn and O. Widlund. FETI and Neumann-Neumann iterative substructuring methods: connections and new results. *Communications on Pure and Applied Mathematics*, 54(1):57–90, 2001.

[116] A. Klawonn and O. B. Widlund. Dual-primal FETI methods for linear elasticity. *Comm. Pure Appl. Math.*, 59(11):1523–1572, 2006.

[117] A. Klawonn, O. B. Widlund, and M. Dryja. Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM J. Numer. Anal.*, 40(1):159–179, 2002.

[118] J. Knepper. Multiskalen-Grobgitterräume für überlappende Schwarz- Gebietszerlegungsverfahren, 2016. Master thesis, University of Cologne.

[119] M. J. Kühn. *Adaptive FETI-DP and BDDC methods for highly heterogeneous elliptic finite element problems in three dimensions.* PhD thesis, Universität zu Köln, Cologne, 2018. Online available at http://kups.ub.uni-koeln.de/id/eprint/8234.

[120] M. H. Lanser. *Nonlinear FETI-DP and BDDC methods.* PhD thesis, Universität zu Köln, 2015. Online available at http://kups.ub.uni-koeln.de/id/eprint/6304.

[121] P. Le Tallec, Y. H. De Roeck, and M. Vidrascu. Domain decomposition methods for large linearly elliptic three-dimensional problems. *J. Comput. Appl. Math.*, 34(1):93–117, 1991.

[122] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[123] J. Li. A dual-primal FETI method for incompressible Stokes equations. *Numerische Mathematik*, 102(2):257–275, 2005.

[124] J. Li and O. Widlund. BDDC algorithms for incompressible Stokes equations. *SIAM journal on numerical analysis*, 44(6):2432–2455, 2006.

[125] J. Li and O. B. Widlund. FETI-DP, BDDC, and block Cholesky methods. *Internat. J. Numer. Methods Engrg.*, 66(2):250–271, 2006.

[126] J. Li and O. B. Widlund. On the use of inexact subdomain solvers for BDDC algorithms. *Comput. Methods Appl. Mech. Engrg.*, 196(8):1415–1428, 2007.

[127] J. Mandel. Balancing Domain Decomposition. *Comm. Numer. Meth. Engrg.*, 9:233–241, 1993.

[128] J. Mandel and C. 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.

[129] J. Mandel, C. R. Dohrmann, and R. Tezaur. An algebraic theory for primal and dual substructuring methods by constraints. *Appl. Numer. Math.*, 54(2):167–193, 2005.

[130] J. Mandel and B. 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.

[131] J. Mandel, B. Sousedík, and C. R. Dohrmann. Multispace and multilevel BDDC. *Computing*, 83(2-3):55–85, 2008.

[132] J. Mandel, B. Sousedík, and J. Sístek. Adaptive BDDC in three dimensions. *Math. Comput. Simulation*, 82(10):1812–1831, 2012.

[133] J. Mandel and R. Tezaur. On the convergence of a dual-primal substructuring method. *Numer. Math.*, 88(3):543–558, 2001.

[134] MATLAB. *version 8.1.0.604 (R2013a).* The MathWorks Inc., Natick, Massachusetts, 2013.

[135] A. Müller and S. Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists.* O'Reilly Media, 2016.

[136] R. Nabben and C. Vuik. A comparison of deflation and the balancing preconditioner. *SIAM Journal on Scientific Computing*, 27(5):1742–1759, 2006.

[137] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th int. conf. on machine learning (ICML-10)*, pages 807–814, 2010.

[138] R. Neugebauer. *Digitalisierung.* Springer, 2018.

[139] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2):355–365, 1987.

[140] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[141] D.-S. Oh, O. B. Widlund, S. Zampini, and C. R. Dohrmann. BDDC algorithms with deluxe scaling and adaptive selection of primal constraints for Raviart-Thomas vector fields. *Math. Comp.*, 87(310):659–692, 2018.

[142] L. F. Pavarino, O. B. Widlund, and S. Zampini. BDDC preconditioners for spectral element discretizations of almost incompressible elasticity in three dimensions. *SIAM J. Sci. Comput.*, 32(6):3604–3626, 2010.

[143] C. Pechstein and C. R. Dohrmann. A unified framework for adaptive BDDC. *Electron. Trans. Numer. Anal.*, 46:273–336, 2017.

[144] C. Pechstein and R. Scheichl. Analysis of FETI methods for multiscale PDEs. *Numer. Math.*, 111(2):293–333, 2008.

[145] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[146] K. H. Pierson. *A family of domain decomposition methods for the massively parallel solution of computational mechanics problems.* PhD thesis, University of Colorado at Boulder, 2000.

[147] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations.* Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2nd edition, 2008.

[148] P. Radtke. *Adaptive Coarse Spaces for FETI-DP and BDDC Methods.* PhD thesis, Universität zu Köln, 2015. Online available at http://kups.ub.uni-koeln.de/id/eprint/6426.

[149] O. Rheinbach. *Parallel scalable iterative substructuring: Robust exact and inexact FETI-DP methods with applications to elasticity.* PhD thesis, Universität Duisburg-Essen, 2006. Online available at https://duepublico2.uni-due.de/receive/duepublico_mods_00013718.

[150] O. Rheinbach. Parallel iterative substructuring in structural mechanics. *Arch. Comput. Methods Eng.*, 16(4):425–463, 2009.

[151] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[152] J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.

[153] Y. Saad. *Iterative methods for sparse linear systems.* SIAM, 2nd edition, 2003.

[154] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, 7:856–869, 1986.

[155] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning.* Cambridge University Press, 2014.

[156] J. C. Simo and T. J. R. Hughes. *Computational Inelasticity.* Interdisciplinary applied mathematics. Springer, 1998.

[157] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain decomposition: Parallel multilevel methods for elliptic partial differential equations.* Cambridge University Press, Cambridge, 1996.

[158] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core.* MIT Press, Cambridge, MA, USA, 2nd edition, 1998.

[159] B. Sousedík. *Comparison of some domain decomposition methods.* PhD thesis, Czech Technical University in Prague, 2008.

[160] B. Sousedík and J. Mandel. On adaptive-multilevel BDDC. In *Domain decomposition methods in science and engineering XIX*, volume 78 of *Lect. Notes Comput. Sci. Eng.*, pages 39–50. Springer, 2011.

[161] B. Sousedík, J. Šístek, and J. Mandel. Adaptive-multilevel BDDC and its parallel implementation. *Computing*, 95(12):1087–1119, 2013.

[162] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numer. Math.*, 126(4):741–770, 2014.

[163] N. Spillane and D. 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.

[164] K. Stüben. An introduction to algebraic multigrid. *Multigrid*, pages 413–532, 2001.

[165] A. Toselli and O. Widlund. *Domain Decomposition Methods—Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2005.

[166] X. Tu. *BDDC Domain Decomposition Algorithms: Methods with Three Levels and for Flow in Porous Media*. PhD thesis, Courant Institute of Mathematical Science, New York University, NY, 2007.

[167] X. Tu. Three-level BDDC in three dimensions. *SIAM J. Sci. Comput.*, 29(4):1759–1780, 2007.

[168] X. Tu. Three-level BDDC in two dimensions. *Internat. J. Numer. Methods Engrg.*, 69(1):33–59, 2007.

[169] J. Watt, R. Borhani, and A. K. Katsaggelos. *Machine learning refined: foundations, algorithms, and applications*. Cambridge University Press, 2nd edition, 2020.

[170] J. Weber. Drei-Level BDDC-Gebietszerlegungsverfahren: Theorie und parallele Implementierung in zwei und drei Dimensionen, 2017. Master thesis, University of Cologne.

[171] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[172] U. M. Yang. On long-range interpolation operators for aggressive coarsening. *Numerical linear algebra with applications*, 17(2-3):453–472, 2010.

[173] S. Zampini. PCBDDC: a class of robust dual-primal methods in PETSc. *SIAM J. Sci. Comput.*, 38(5):S282–S306, 2016.

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:

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Predicting the geometric location of critical edges in adaptive GDSW overlapping domain decomposition methods using deep learning. TR series, Center for Data and Simulation Science, University of Cologne, Germany, Vol. 2021-2. https://kups.ub.uni-koeln.de/id/eprint/36257. Accepted for publication in the proceedings of the International Conference on Domain Decomposition Methods 26, Springer LNCSE, August 2021.

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Machine Learning in Adaptive Domain Decomposition Methods - Predicting the Geometric Location of Constraints. *SIAM J. Sci. Comput.*, 41(6):A3887–A3912, 2019.

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. A Frugal FETI-DP and BDDC Coarse Space for Heterogeneous Problems. *Electr. Trans. Numer. Anal.*, 53:562–591, 2020.

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Machine Learning in Adaptive FETI-DP - A Comparison of Smart and Random Training Data. In *Domain decomposition methods in science and engineering XXV*, volume 138 of *Lect. Notes Comput. Sci. Eng.*, pages 218–226. Springer, 2020.

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Combining Machine Learning and Adaptive Coarse Spaces - A Hybrid Approach for Robust FETI-DP Methods in Three Dimensions. *SIAM J. Sci. Comput.*, 43(5):S816–S838, 2021. Special Section Copper Mountain 2020.

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Combining Machine Learning and Domain Decomposition Methods for the Solution of Partial Differential Equations–A Review. *GAMM-Mitt.*, 44(1):e202100001, 2021.

- A. Heinlein, A. Klawonn, M. Lanser, and J. Weber. Machine Learning in Adaptive FETI-DP - Reducing the Effort in Sampling. In *Numerical Mathematics and*

Advanced Applications ENUMATH 2019, volume 139 of *Lect. Notes Comput. Sci. Eng.*, pages 593–603. Springer, 2021.

- A. Klawonn, M. Lanser, O. Rheinbach, and J. Weber. Preconditioning the coarse problem of BDDC methods - Three-level, algebraic multigrid, and vertex-based preconditioners. *Electron. Trans. Numer. Anal.*, 51:432-450, 2019.

Ort, Datum Unterschrift (J. Weber)