The Nitsche-based Partition of Unity Method for coupled problems in linear elasticity

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Pablo Jiménez Recio

aus

Madrid, Spanien

Bonn, 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter/Betreuer: Prof. Dr. Marc Alexander Schweitzer

Gutachter: Prof. Dr. Jürgen Dölz

Tag der Promotion: 16.01.2026

Erscheinungsjahr: 2026

Sich zwar immer mehr und mehr in den aufregendsten und in den ungeheuerlichsten und in den epochemachendsten Gedanken zu schulen und sich solchen einzigen für ihn noch möglichen Gedanken mit einer noch immer größeren Entschlossenheit vollkommen auszuliefern, sei seine tagtägliche Disziplin, aber nur immer bis zu dem äußersten Grade vor *der absoluten Verrücktheit. Geht man so weit, wie Karrer, sagt Oehler, ist man plötzlich entschieden und absolut verrückt und mit einem Schlag wertlos geworden. Denken und immer mehr und immer mehr mit immer größerer Intensität und mit einer immer noch größeren Rücksichtslosigkeit und mit einem immer noch größeren Erkenntnisfanatismus, sagt Oehler, aber nicht einen Augenblick zu weit denken. Jeden Augenblick können wir zu weit denken, sagt Oehler, einfach zu weit gehen in unserem Denken, sagt Oehler, und alles ist wertlos.*

– Thomas Bernhard, *Gehen*

Acknowledgments

I would like to thank, first and foremost, Albert Ziegenhagel, a *rara avis* in this mathematics environment, whose non-academic approach to work felt like a breath of fresh air. It is from him that I have learned the most during the course of my doctoral studies: first directly, while he was part of the group (this wording does not do justice to the role that he had: he held the whole project on his shoulders), and then indirectly, since the whole PUMA library carries his indelible signature. All I have aimed for in the last few years is to honor his work and make the PUMA library as efficient, robust, clearly-written and well-designed as it can be, all of which I learned from Albert himself. Every one of my merits can be traced back to him, and for that I will remain forever indebted.

In the absence of Albert, Lukas Troska became the only reason I was able to remain part of the PUMA group. His passion for this project is on par with mine and Albert's, and together we have carried the project forward after Albert left, making the PUMA library what it is today. Many of my achievements would simply not have been possible without him, as well as the writing of this thesis, which I would have very gladly abandoned at multiple occasions during the last two years. To him I also remain forever grateful.

I would also like to thank Matthias Birner, Jannik Michels and Paul Kühner, who always put their best effort in contributing to the PUMA library. Sadly, sometimes one learns to appreciate these details only when the people are no longer there. I extend my appreciation to all members of the PUMA group, acknowledging the work that we have all carried out to bring the project forward.

Finally, I would like to thank my advisor Marc Alexander Schweitzer, who laid the first stone of the PUMA project, and without whose intervention I would not have met all the aforementioned people. Had he not offered me a PhD position in his group (and a SHK position before that), I would have likely ended up in some strictly academic group, and who knows how more miserable I would be right now. PUMA has now occupied the last eight years of my life and, for better or worse, I have been shaped by this project perhaps even more intensely than I have shaped it myself.

Por supuesto, mi más sincero agradecimiento, naturalmente no restringido a la realización de este trabajo, es para mi familia, incluyendo a aquellos que tristemente ya no están con nosotros.

Abstract

This thesis addresses the use of Nitsche's method within the Partition of Unity Method (PUM), and in particular within the PUMA software library developed both at Fraunhofer SCAI and at the Institute for Numerical Simulation, which has provided the practical ground for the mathematical developments. While the specific adaptation of Nitsche's method to the PUM has been already published in a paper attached to this thesis, the focus of this work lies on problems arising from linear elasticity, with the aim of handling interface constraints for the coupling of solid and shell models. These include not only single isotropic materials, but also laminated structures consisting of multiple orthotropic elastic materials, which are used to simulate fiber-reinforced composites. This work revealed the lack of efficient and robust smoothers for PUMA's multilevel solver, which should be able to handle anisotropic and higher-order problems. The development of such a smoother, based on the Factorized Sparse Approximate Inverse (FSAI) preconditioner and combined with the Chebyshev iteration, has been the subject of another paper, which is also attached to this thesis. Additionally, the coupling of shell structures parameterized by Non-Uniform Rational B-Splines (NURBS) required the development and implementation of projection algorithms capable of projecting a point into a NURBS curve or surface (also retrieving the corresponding parameters). All in all, this thesis and its attached papers involve the formulation and discretization of PDEs with the Nitsche-based PUM, including both geometrical and analytical complications, its numerical integration (handling all involved operations in an efficient but also rigorous manner), and finally the iterative solution of the resulting linear systems (again with specific emphasis on efficiency), thus encompassing all stages of the numerical solution of linear and elliptic PDEs. The developments are accompanied by several numerical tests, which also showcase the practical work carried out for a significant improvement of the PUMA library.

Contents

1 Introduction

The Partition of Unity Method (from now on, PUM), which provides the basis for this thesis, is a discretization approach for the numerical solution of partial differential equations (PDEs) on Lipschitz domains via Galerkin methods. Galerkin methods, roughly speaking, consist on replacing the infinite-dimensional function spaces (Sobolev spaces in particular) arising in the weak formulation of some PDE by finite-dimensional (discrete) ones, which lead to problems with a finite number of degrees of freedom whose solution should approximate that of the original problem, and which can in turn be found (at least approximately) with the help of a computer.

The Finite Element Method (FEM), whose description is available in many places, is the most popular method for the purpose of discretizing function spaces, mainly for historical reasons, but also for its intuitive description, particularly appreciated by the engineering community. Over the years, a myriad other methods have been proposed to overcome the FEM's intrinsic limitations, some of which are closer to the FEM (e.g. XFEM, GFEM, CutFEM, the Finite Cell Method), others more distant (e.g. the Reproducing Kernel Particle Method, the hp-clouds method, the PUM). The FEM's main limitation is the need to construct a mesh of the problem's domain (be it with triangles, squares, tetrahedra or hexahedra), and then to refine it and/or coarsen it, all of which carries a significant computational cost. So-called meshfree methods (and among them, the PUM) aim to overcome this limitation by operating without the need for a mesh. An additional limitation of the FEM is the discretization of higher-order problems, involving higher (than $H^1(\Omega)$) regularity of the function spaces, which the FEM can handle, but not trivially. The PUM can however easily fulfill such regularity requirements by definition. Another limitation is the incorporation of discontinuities in the function space: XFEM and the PUM achieve this through so-called enrichment functions, which are carefully introduced into the basis functions of the discrete function space.

All in all, the PUM overcomes all of the FEM's main limitations, but needless to say, at a certain cost: a more complicated mathematical description, and therefore a more involved computational implementation (mainly at the integration stage), combined with the loss of "nice-to-have" FEM properties like the nodal basis. A brief account of the PUM, including the necessary concepts for the rest of the thesis, is given in Chapter 2.

Nitsche's method, the second main component of this thesis, consists of a manipulation of the Galerkin equation so that essential boundary conditions (usually embedded in the function spaces themselves) can be imposed in a "weak" manner (i.e. approximately) through the incorporation of additional terms. Since the solution of the discretized problem is only an approximation of the exact solution, imposing the boundary conditions also in an approximate manner introduces no additional issue, provided that the order of the approximation error remains unaffected.

In the standard FEM, essential boundary conditions are usually imposed at the boundary nodes, and thus Nitsche's method is not commonly used. In meshfree methods, and in particular in the PUM, as already mentioned, a nodal basis is not directly available, and thus alternative approaches have to be devised for the imposition of essential boundary conditions. Specifically for the PUM,

an algebraic approach was introduced by Schweitzer [20] which was derived as a limiting case of Nitsche's method. However, it has certain limitations, e.g. for interface constraints coupling two solutions from different function spaces, or for the simultaneous enforcement of different kinds of boundary conditions (which emerge in higher-order problems). Alternatively, Nitsche's method was already used in combination with the PUM in [7], where the stabilization function (the main component of the method) was chosen simply as a constant over the whole boundary. The first significant development for this thesis was to introduce a locally-defined stabilization function, something which was already common in the Nitsche-FEM, and which I generalized for the PUM in [14] (attached in Appendix C.1). The so-called flat-top property of the PUM, already stated in [8] but never really enforced for boundary patches in our implementation, revealed itself critical for the proper application of Nitsche's method in non-trivial domains. A patch-aggregation approach addressing this issue was also included in [14].

The implementation of Nitsche's method in the PUMA library allowed us to use it in a variety of interesting cases, in particular those involving Kirchhoff-Love plates and shells, which encompass multiple kinds of essential boundary conditions. For this kind of problems, interfacial constraints are also of particular interest, since real-world shell structures are composed of multiple different parts. An additional interesting topic is the treatment of certain shell parts as 3-dimensional volumes while coupling them to other parts considered as 2-dimensional surfaces, known as *solid-shell blending*. Chapter 3 is dedicated to all this. The extension to laminated composite materials, where each ply is modeled as a linearly orthotropic material with a given orientation, is covered in Chapter 4.

However, precisely the higher order of the 2-dimensional shell models and the laminated structure of 3-dimensional models for composite materials make the iterative solution of this kind of problems particularly difficult. Although a multilevel solver as the one described in [6] was already available in PUMA, only Jacobi and Gauss-Seidel preconditioners were available as smoothers, which were of little use for these cases. For that reason, the FSAI preconditioner was implemented following [12], with the extensions of nestedness and adaptivity. Additionally, FSAI's smoothing performance was improved by embedding it into a Chebyshev polynomial iteration. All this work on linear solvers for symmetric positive definite (s.p.d.) matrices has been published in [15] (attached in Appendix C.2) and is summarized in Chapter 5.

Finally, in Chapter 6 we present several numerical examples that put into practice some of the concepts and developments of this thesis, followed by an overview of the whole work in Chapter 7.

For the coupling of different shell parametrizations given as non-uniform rational B-spline (NURBS) surfaces, it was necessary to implement two algorithms for the projection of a point onto a NURBS curve or surface. This work (especially for the surface case) required a careful assessment of the available literature. For the interested reader, we provide detailed and systematic descriptions of both algorithms and their components in Appendix A.

In Appendix B we describe some relevant changes to the PUMA library that have been developed parallel to the thesis work, to make it the efficient, rigorous and clearly-written (although unfortunately still academic) library that it is today.

Finally, Appendix C gathers the two papers that we attach to this thesis, as already mentioned.

2 The Partition of Unity Method

In this chapter we will present a minimal set of PUM concepts that should help understand the background of our work. For more details, we refer to [14] (Appendix C.1) and the references therein, in particular to Schweitzer's seminal work [18], and also to Ziegenhagel's thesis [22].

Definition 2.1. Let $\Omega \in \mathbb{R}^d$ be some open domain. We say that a set of functions $\{\varphi_i\}_{i=0}^n$ with $\varphi_i : \bar{\Omega} \to \mathbb{R}$ is a *partition of unity* over Ω if

$$0 \leq \varphi_i(x) \leq 1 \quad \forall i \in \{0, \dots, n\}, \quad \text{and} \quad \sum_{i=0}^{n} \varphi_i(x) = 1, \quad \forall x \in \bar{\Omega}.$$

Given some Lipschitz domain $\Omega \in \mathbb{R}^d$, the PUM is based on the construction of a partition of unity $\{\varphi_i\}_{i=0}^n$ over Ω, and then of finite-dimensional function spaces of the form

$$V^{\mathrm{PU}}(\Omega) = \sum_{i=0}^{n} \varphi_i V_i(\omega_i), \quad \omega_i \coloneqq \operatorname{supp}(\varphi_i).$$

In other words, the partition of unity is used to glue together so-called *local spaces* $V_i(\omega_i)$, each of which is assumed to be defined on the corresponding ω_i. We refer to the ω_i as *patches*, and to the set $C \coloneqq \{\omega_i\}_{i=0}^n$ as a *cover* of Ω. Usually Ω and ω_i are dropped from the notation, as we will do here, writing simply $V^{\mathrm{PU}} = \sum_{i=0}^n \varphi_i V_i$.

In order to produce computationally-tractable problems for increasing n, it is important to ensure that each patch ω_i intersects with a limited number of the remaining patches. More precisely, the method relies on covers $C = \{\omega_i\}_{i=0}^n$ fulfilling

$$\lambda_C(x) \coloneqq \operatorname{card}\left(\{i : x \in \omega_i\}\right) \leq M, \quad \forall x \in \bar{\Omega}, \tag{2.1}$$

for some constant $M > 0$ independent of the cover size. A homogeneously refined cover for a square domain is represented in Figure 2.1.

With respect to the local spaces V_i, they are usually chosen simply as polynomial spaces of degree p_i, $V_i = \mathcal{P}^{p_i}$ (or their d-dimensional counterparts for vector problems, $[\mathcal{P}^{p_i}]^d$). However, a defining characteristic of the PUM is the possibility to *enrich* the local spaces with any additional functions that may be helpful for the problem at hand, i.e. to construct

$$V_i = \mathcal{P}^{p_i} + \mathcal{E}_i.$$

We refer to $\mathcal{E}_i$ as *enrichment spaces* and to their components as *enrichment functions*.

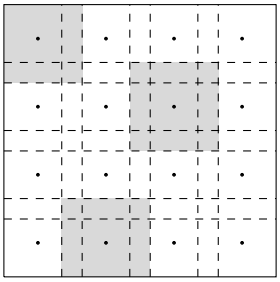


Figure 2.1: A homogeneously refined cover for a square domain. Each black dot corresponds to a patch center, patch boundaries are dashed, and for illustrative purposes three patches are colored in gray.

The regularity of any $u \in V^{\mathrm{PU}}$ is bounded by that of the partition of unity functions φ_i, which can actually be constructed to satisfy $\varphi_i \in C^k(\Omega)$ for any predefined $k \in \mathbb{N}$ (or simply piecewise constant), as illustrated in [14, Section 2.2]. As a result, the PUM can naturally produce function spaces fulfilling any continuity requirement on its derivatives.

If we denote $\{\vartheta_i^k\}_{k=1}^{m_i}$ a basis for each local space V_i, the evaluation of an element $u \in V^{\mathrm{PU}}$ can be written as

$$u(x) = \sum_{i=0}^{n} \varphi_i(x) \sum_{k=1}^{m_i} \bar{u}_{(i,k)} \vartheta_i^k(x), \quad x \in \Omega, \tag{2.2}$$

for a vector of coefficients $\bar{u} = (\bar{u}_{(i,k)}) \in \prod_{i=0}^{n} \mathbb{R}^{m_i}$, which uniquely identifies $u \in V^{\mathrm{PU}}$ if the global basis functions

$$\left\{\varphi_i \vartheta_i^k : \; i = 0, \dots, n; \; k = 1, \dots, m_i\right\} \tag{2.3}$$

are linearly independent. With respect to this aspect, let us introduce an important but sometimes overlooked property of the PUM.

Definition 2.2 (Flat-top property)**.** Let $\Omega \in \mathbb{R}^d$ be some open domain and $\{\varphi_i\}_{i=0}^{n}$ a partition of unity over Ω. Let us denote $\omega_i = \operatorname{supp}(\varphi_i)$. We call

$$\omega_{i,\mathrm{FT}} \coloneqq \omega_i \setminus \bigcup_{\substack{j=0 \\ j \neq i}}^{n} \omega_j$$

the *flat-top region* of φ_i, given that $\varphi_i(x) = 1$ for any $x \in \omega_{i,\mathrm{FT}}$.

Let $C > 0$ be some positive constant. We say that $\{\varphi_i\}$ satisfies the *flat-top property with constant* C if it holds that

$$\mu_d(\omega_{i,\mathrm{FT}}) \geq C \, \mu_d(\omega_i), \quad \forall i \in \{0, \dots, n\},$$

where μ_d is the d-dimensional Lebesgue measure.

The flat-top property was initially formulated to derive linear independence of the *global* basis functions based on linear independence of the *local* basis functions (within their flat-top regions) [19], for which it was a sufficient but not necessary condition. However, when working with Nitsche's method, it revealed itself as a necessary condition to prevent the stabilization function from taking arbitrarily large values.

Ideally, we would like to fix some $C > 0$ and produce only partitions of unity satisfying the flat-top property with constant C. However, this is not an easy task for arbitrary domains. While one can easily enforce the constraint for functions φ_i with $\omega_i \cap \partial\Omega = \emptyset$, the same cannot be said for functions with support at the boundary. In [14, Section 2.3] we presented a post-processing transformation of partitions of unity that, for a given $C > 0$, "aggregates" each φ_i violating the flat-top property with constant C to some neighboring φ_j not violating it.

Finally, let us now describe the PUM discretization of a generic classical elliptic problem, i.e.

$$\text{find} \quad u \in V \quad : \quad a(v,u) = \ell(v), \quad \forall v \in V,$$

with $a\colon V \times V \to \mathbb{R}$ some continuous V-elliptic bilinear form, and $\ell\colon V \to \mathbb{R}$ some continuous linear form. Then, the Galerkin discretization for a space $V^{\mathrm{PU}} \subset V$ reads simply

$$\text{find} \quad u \in V^{\mathrm{PU}} \quad : \quad a(v,u) = \ell(v), \quad \forall v \in V^{\mathrm{PU}}.$$

Relying on the basis functions (2.3), this problem can be represented by the linear system $A\bar{u} = b$, for the vector of coefficients $\bar{u}$ as in (2.2), and the stiffness matrix A and load vector b given by

$$A_{(i,k),(j,l)} = a(\varphi_i \vartheta_i^k,\, \varphi_j \vartheta_j^l), \quad b_{(i,k)} = \ell(\varphi_i \vartheta_i^k). \tag{2.4}$$

Linear systems arising from PUM discretizations thus possess a natural block structure, i.e. the stiffness matrix A can be seen as

$$A = \begin{pmatrix} A[0,0] & A[0,1] & \cdots & A[0,n] \\ A[1,0] & A[1,1] & \cdots & A[1,n] \\ \vdots & \vdots & \ddots & \vdots \\ A[n,0] & A[n,1] & \cdots & A[n,n] \end{pmatrix},$$

with each non-zero block $A[i,j]$ (such that $\operatorname{supp}(\varphi_i) \cap \operatorname{supp}(\varphi_j) \neq \emptyset$) given by

$$A[i,j] \coloneqq \begin{pmatrix} A_{(i,0),(j,0)} & A_{(i,0),(j,1)} & \cdots & A_{(i,0),(j,m_j)} \\ A_{(i,1),(j,0)} & A_{(i,1),(j,1)} & \cdots & A_{(i,1),(j,m_j)} \\ \vdots & \vdots & \ddots & \vdots \\ A_{(i,m_i),(j,0)} & A_{(i,m_i),(j,1)} & \cdots & A_{(i,m_i),(j,m_j)} \end{pmatrix} \in \mathbb{R}^{m_i \times m_j}, \quad i,j \in \{0,\dots,n\},$$

and the b vector analogously as

$$b = \begin{pmatrix} b[0] \\ b[1] \\ \vdots \\ b[n] \end{pmatrix}, \quad \text{with} \quad b[i] \coloneqq \begin{pmatrix} b_{(i,0)} \\ b_{(i,1)} \\ \vdots \\ b_{(i,m_i)} \end{pmatrix} \in \mathbb{R}^{m_i}, \quad i \in \{0,\dots,n\}.$$

The PUMA library

For a concise and practical illustration of the PUMA library, in Listing 2.1 we provide a `python` script to solve the Poisson equation

$$\text{find} \quad u \in V^{\text{PU}} \quad : \quad \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega f v \, \mathrm{d}\Omega, \quad \forall v \in V^{\text{PU}},$$

in the unit square domain $\Omega = (0,1)^2$, with r.h.s. $f(x) = -6$, and with Dirichlet boundary condition $u(x,y) = 1 + x^2 + 2y^2$ over the whole boundary.

```python
from puma import *

# Define the domain
omega = UnitSquareDomain()

# Create the PUM space for a certain refinement level and a certain polynomial degree
V = PumSpace(omega, level=5, polynomial_degree=3)

# Define the boundary data
x = Position()
g = 1 + x[0]**2 + 2*x[1]**2

bc = DirichletConstraint(V, None, g)

# Define Poisson's problem
u = ScalarTrialFunction(V)
v = ScalarTestFunction(V)

a = inner(grad(u), grad(v)) * dx

f = -6
L = f * v * dx

# Define the discrete solution
u = ScalarFunction(V)

# Solve into u
solver = solve(a == L, u, bc)
```

Listing 2.1: A `python` script for solving a simple Poisson problem with the PUMA library.

3 Nitsche's method

In this section we will introduce Nitsche's method for the equations of static linear elasticity. For a more detailed introduction based on Poisson's equation, we refer to our previous work [14] (included in Appendix C.1), and for a generic mathematical formulation, we refer to the work of Benzaken et al. [2].

3.1 Linear elasticity

Although it is an ubiquitous equation in numerical analysis, let us present the equations of static linear elasticity. For a more mathematically-detailed description, we refer to [4]. Let $\Omega \in \mathbb{R}^d$ (with $d \in \{2, 3\}$) be an open Lipschitz domain, $\Gamma_u \subset \partial\Omega$ the Dirichlet boundary, which we assume to have positive measure, i.e. $\mu_{d-1}(\Gamma_u) > 0$, and $\Gamma_t \coloneqq \partial\Omega \setminus \Gamma_u$ the Neumann boundary. For a displacement field $\mathbf{u} \in \left[C^1(\bar{\Omega})\right]^d$, the infinitesimal strain tensor is $\boldsymbol{\varepsilon}(\mathbf{u}) \coloneqq \operatorname{sym}(\nabla\mathbf{u})$, and the resulting Cauchy stress tensor follows Hooke's law $\boldsymbol{\sigma}(\mathbf{u}) \coloneqq \mathbb{C} : \boldsymbol{\varepsilon}(\mathbf{u})$ with the 4th-order stiffness tensor $\mathbb{C} \in \mathbb{R}^{d\times d\times d\times d}$. The symmetry of the strain and stress tensors enforces the following symmetry relationships on the stiffness tensor

$$\mathbb{C}_{ijkl} = \mathbb{C}_{jikl}, \quad \mathbb{C}_{ijkl} = \mathbb{C}_{klij}, \tag{3.1}$$

and we additionally assume the ellipticity condition

$$\exists C_0 > 0 \quad : \quad \sum_{i,j,k,l} \mathbb{C}_{ijkl} A_{ij} A_{kl} \geq C_0 \sum_{i,j} A_{ij}^2 = C_0 \operatorname{tr}(A^2), \quad \forall A \in \operatorname{Sym}_d(\mathbb{R}), \tag{3.2}$$

where $\operatorname{Sym}_d(\mathbb{R})$ denotes the set of real symmetric matrices of size d. The equations of linear elasticity (in strong form) are then

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{f}, & \text{in} \quad \Omega, \\ \mathbf{u} = \mathbf{g}, & \text{on} \quad \Gamma_u, \\ \boldsymbol{\sigma}(\mathbf{u})\mathbf{n} = \mathbf{t}, & \text{on} \quad \Gamma_t. \end{cases} \tag{3.3}$$

where $\mathbf{f} \in [C(\Omega)]^d$ is the body force per unit volume, $\mathbf{g} \in [C(\Gamma_u)]^d$ is the prescribed displacement at the Dirichlet boundary, and $\mathbf{t} \in [C(\Gamma_t)]^d$ is the prescribed traction at the Neumann boundary.

For the particular case of an isotropic homogeneous material, the simplest model of linear elasticity, the stress tensor can be written as

$$\boldsymbol{\sigma}(\mathbf{u}) = \lambda \operatorname{tr}(\boldsymbol{\varepsilon}(\mathbf{u}))\, \mathbb{I}_d + 2\mu\, \boldsymbol{\varepsilon}(\mathbf{u}) \,. \tag{3.4}$$

with the Lamé parameters $\mu > 0$, $\lambda > 0$, which can be related to Young's modulus $E > 0$ and Poisson's ratio $\nu \in \left(0, \frac{1}{2}\right)$ as

$$\lambda = \frac{E\,\nu}{(1+\nu)\,(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}, \qquad E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}, \quad \nu = \frac{\lambda}{2(\lambda+\mu)}.$$

To derive the weak formulation of (3.3), we first multiply the differential equation by a test function $\mathbf{v} \in V \coloneqq \left[H^1(\Omega)\right]^d$ and integrate by parts over Ω, leading to

$$\int_\Omega \boldsymbol{\sigma}(\mathbf{u}) \colon \boldsymbol{\varepsilon}(\mathbf{v}) \,\mathrm{d}\Omega - \int_{\partial\Omega} (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n}) \cdot \mathbf{v} \,\mathrm{d}\Gamma = \int_\Omega \mathbf{f} \cdot \mathbf{v} \,\mathrm{d}\Omega\,.$$

Splitting the boundary integral into the Dirichlet and Neumann boundaries, and incorporating the prescribed traction, we arrive at

$$\int_\Omega \boldsymbol{\sigma}(\mathbf{u}) \colon \boldsymbol{\varepsilon}(\mathbf{v}) \,\mathrm{d}\Omega - \int_{\Gamma_u} (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n}) \cdot \mathbf{v} \,\mathrm{d}\Gamma = \int_\Omega \mathbf{f} \cdot \mathbf{v} \,\mathrm{d}\Omega + \int_{\Gamma_t} \mathbf{t} \cdot \mathbf{v} \,\mathrm{d}\Gamma, \quad \forall \mathbf{v} \in V. \tag{3.5}$$

At this point, the usual procedure is to reduce the test space to $V^0 \coloneqq \{\mathbf{w} \in V : \ \mathbf{w}|_{\Gamma_u} = 0\}$ and the solution space to $V^g \coloneqq \{\mathbf{w} \in V : \ \mathbf{w}|_{\Gamma_u} = g\}$, leading to the problem

$$\text{find} \quad \mathbf{u} \in V^g \quad : \quad a(\mathbf{v}, \mathbf{u}) = \ell(\mathbf{v}), \quad \forall \mathbf{v} \in V^0,$$

where

$$a(\mathbf{v}, \mathbf{u}) \coloneqq \int_\Omega \boldsymbol{\sigma}(\mathbf{u}) \colon \boldsymbol{\varepsilon}(\mathbf{v}) \,\mathrm{d}\Omega\,, \quad \ell(\mathbf{v}) \coloneqq \int_\Omega \mathbf{f} \cdot \mathbf{v} \,\mathrm{d}\Omega + \int_{\Gamma_t} \mathbf{t} \cdot \mathbf{v} \,\mathrm{d}\Gamma,$$

and the regularity of the data can be weakened to $\mathbf{f} \in \left[L^2(\Omega)\right]^d$, $\mathbf{g} \in \left[L^2(\Gamma_u)\right]^d$ and $\mathbf{t} \in \left[L^2(\Gamma_t)\right]^d$. Notably, this problem is equivalent to the minimization of the energy functional

$$\mathcal{J}(\mathbf{w}) \coloneqq \frac{1}{2} a(\mathbf{w}, \mathbf{w}) - \ell(\mathbf{w}) = \frac{1}{2} \int_\Omega \boldsymbol{\sigma}(\mathbf{w}) \colon \boldsymbol{\varepsilon}(\mathbf{w}) \,\mathrm{d}\Omega - \int_\Omega \mathbf{f} \cdot \mathbf{w} \,\mathrm{d}\Omega - \int_{\Gamma_t} \mathbf{t} \cdot \mathbf{w} \,\mathrm{d}\Gamma, \quad \mathbf{w} \in V^g.$$

Furthermore, given some $\mathbf{u}_g \in V^g$, the problem can be reformulated as

$$\text{find} \quad \mathbf{u}_0 \in V^0 \quad : \quad a(\mathbf{v}, \mathbf{u}_0) = \ell(\mathbf{v}) - a(\mathbf{v}, \mathbf{u}_g), \quad \forall \mathbf{v} \in V^0, \tag{3.6}$$

and finally setting $\mathbf{u} = \mathbf{u}_0 + \mathbf{u}_g \in V^g$.

It is important to notice that the bilinear form $a(\cdot, \cdot)$ is symmetric, which follows from the symmetries of the stiffness tensor (3.1), that it is V-continuous, and that it is also V^0-coercive, which follows from (3.2) and Korn's inequalities. These three properties guarantee, following the Lax-Milgram theorem, that (3.6) is well-posed and its solution is unique (see [4, Section 6.3]).

Given a discrete subspace $V_n \subset V^0$, the standard Galerkin discretization of Equation (3.6) is then

$$\text{find} \quad \mathbf{u}_0 \in V_n \quad : \quad a(\mathbf{v}, \mathbf{u}_0) = \ell(\mathbf{v}) - a(\mathbf{v}, \mathbf{u}_g), \quad \forall \mathbf{v} \in V_n,$$

which also has a unique solution. Nitsche's method, however, avoids embedding the boundary condition into the function space, considering instead $V_n \subset V$ as the space for the solution and the test functions, and proceeding from (3.5) as follows. First, we add a *penalty term* $\int_{\Gamma_u} \gamma(\mathbf{u}-\mathbf{g})\cdot\mathbf{v}\ \mathrm{d}\Gamma$ to the equation, with a stabilization function $\gamma \in L^2(\Gamma_u)$, $\gamma > 0$, resulting in

$$\int_\Omega \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\ \mathrm{d}\Omega + \int_{\Gamma_u} [\gamma\,\mathbf{u}\cdot\mathbf{v} - (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n})\cdot\mathbf{v}]\ \mathrm{d}\Gamma = \int_\Omega \mathbf{f}\cdot\mathbf{v}\ \mathrm{d}\Omega + \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{v}\ \mathrm{d}\Gamma + \int_{\Gamma_u} \gamma\,\mathbf{g}\cdot\mathbf{v}\ \mathrm{d}\Gamma\,,$$

and then a *symmetrization term* $\int_{\Gamma_u} (\boldsymbol{\sigma}(\mathbf{v})\mathbf{n})\cdot(\mathbf{g}-\mathbf{u})\ \mathrm{d}\Gamma$ to make the l.h.s. of the equation symmetric with respect to the pair $(\mathbf{u},\mathbf{v})$:

$$\int_\Omega \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\ \mathrm{d}\Omega + \int_{\Gamma_u} [\gamma\,\mathbf{u}\cdot\mathbf{v} - (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n})\cdot\mathbf{v} - (\boldsymbol{\sigma}(\mathbf{v})\mathbf{n})\cdot\mathbf{u}]\ \mathrm{d}\Gamma = \\ = \int_\Omega \mathbf{f}\cdot\mathbf{v}\ \mathrm{d}\Omega + \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{v}\ \mathrm{d}\Gamma + \int_{\Gamma_u} \mathbf{g}\cdot[\gamma\,\mathbf{v} - \boldsymbol{\sigma}(\mathbf{v})\mathbf{n}]\ \mathrm{d}\Gamma\,. \tag{3.7}$$

This symmetrization term may also be multiplied by some factor $\theta \in [-1, 1)$, which would produce a non-symmetric bilinear form. We will not consider this possibility here, since we actively seek symmetry with the purpose of producing symmetric positive definite (s.p.d.) matrices.

After defining the bilinear form $a_\gamma : V_n \times V_n \to \mathbb{R}$ and the linear form $\ell_\gamma : V_n \to \mathbb{R}$ respectively from the l.h.s. and the r.h.s. of (3.7) as

$$a_\gamma(\mathbf{v},\mathbf{u}) \coloneqq \int_\Omega \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\ \mathrm{d}\Omega + \int_{\Gamma_u} [\gamma\,\mathbf{u}\cdot\mathbf{v} - (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n})\cdot\mathbf{v} - (\boldsymbol{\sigma}(\mathbf{v})\mathbf{n})\cdot\mathbf{u}]\ \mathrm{d}\Gamma\,,$$
$$\ell_\gamma(\mathbf{v}) \coloneqq \int_\Omega \mathbf{f}\cdot\mathbf{v}\ \mathrm{d}\Omega + \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{v}\ \mathrm{d}\Gamma + \int_{\Gamma_u} \mathbf{g}\cdot[\gamma\,\mathbf{v} - \boldsymbol{\sigma}(\mathbf{v})\mathbf{n}]\ \mathrm{d}\Gamma\,,$$

the discretization of (3.6) following Nitsche's method can be written as

$$\text{find} \quad \mathbf{u} \in V_n \quad : \quad a_\gamma(\mathbf{v},\mathbf{u}) = \ell_\gamma(\mathbf{v}), \quad \forall \mathbf{v} \in V_n.$$

In this scenario, if a_γ is both continuous and coercive, then the Lax-Milgram theorem guarantees that a solution to the problem exists and is unique. Continuity is trivial to show, and relying on Cauchy-Schwarz' and Young's inequalities, it can also be shown that a_γ is V_n-coercive provided that

$$\exists \rho \in (0,1) \quad : \quad \int_{\Gamma_u} \gamma^{-1} |\boldsymbol{\sigma}(\mathbf{v})\mathbf{n}|^2\ \mathrm{d}\Gamma \le \rho^2 \int_\Omega \boldsymbol{\sigma}(\mathbf{v}) : \boldsymbol{\varepsilon}(\mathbf{v})\ \mathrm{d}\Omega, \quad \forall \mathbf{v} \in V_n \tag{3.8}$$

(see [14, Section 3.1] for a detailed argument in the case of the Poisson equation). Note that the finite-dimensionality of V_n makes the presence of ρ superfluous in the above inequality, since it is equivalent to a strict inequality without ρ^2. Nevertheless, we keep the ρ^2 factor in the notation, since that allows us to define the stabilization function γ based on a fixed value of ρ, independent of the discrete function space V_n.

3.2 Linear elasticity with interfacial constraints

Let us now formulate the linear elasticity equations for a domain formed by two distinct components $\Omega^{(1)}, \Omega^{(2)} \subset \mathbb{R}^3$, both open Lipschitz domains, with $\Omega^{(1)} \cap \Omega^{(2)} = \emptyset$ and $\bar{\Omega}^{(1)} \cap \bar{\Omega}^{(2)} = \Gamma_I$. In this case, the equations of linear elasticity can be written as

$$
\begin{cases}
-\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}^{(k)}) = \mathbf{f}^{(k)}, & \text{in} \quad \Omega^{(k)}, \quad k \in \{1,2\}, \\
\mathbf{u}^{(k)} = \mathbf{g}^{(k)}, & \text{on} \quad \Gamma_u^{(k)}, \quad k \in \{1,2\}, \\
\boldsymbol{\sigma}(\mathbf{u}^{(k)})\mathbf{n} = \mathbf{t}^{(k)}, & \text{on} \quad \Gamma_t^{(k)}, \quad k \in \{1,2\}, \\
[[\mathbf{u}]] = \mathbf{0}, & \text{on} \quad \Gamma_I, \\
[[\boldsymbol{\sigma}(\mathbf{u})\,\mathbf{n}_I]] = \mathbf{0}, & \text{on} \quad \Gamma_I.
\end{cases} \tag{3.9}
$$

where $[[\cdot]]$ represents the jump of a (vector) quantity at the interface, i.e. $[[\mathbf{u}]] \coloneqq \mathbf{u}^{(2)} - \mathbf{u}^{(1)}$ is the displacement jump at the interface, and $\mathbf{n}_I$ is the normal to the interface pointing from $\Omega^{(1)}$ to $\Omega^{(2)}$ (note however that both objects, the jump and the interface normal, can be simultaneously defined in the opposite sense).

The corresponding weak formulation derived following Nitsche's method is

$$
\begin{aligned}
&\sum_{k=1}^{2} \int_{\Omega^{(k)}} \boldsymbol{\sigma}(\mathbf{u}^{(k)}) : \boldsymbol{\varepsilon}(\mathbf{v}^{(k)}) \, \mathrm{d}\Omega \\
&+ \sum_{k=1}^{2} \int_{\Gamma_u^{(k)}} \left[\gamma^{(k)} \, \mathbf{u}^{(k)} \cdot \mathbf{v}^{(k)} - \left(\boldsymbol{\sigma}(\mathbf{u}^{(k)})\mathbf{n} \right) \cdot \mathbf{v}^{(k)} - \left(\boldsymbol{\sigma}(\mathbf{v}^{(k)})\mathbf{n} \right) \cdot \mathbf{u}^{(k)} \right] \mathrm{d}\Gamma \\
&+ \int_{\Gamma_I} \left[\{\gamma\}_\alpha \, [[\mathbf{u}]] \cdot [[\mathbf{v}]] - \{\boldsymbol{\sigma}(\mathbf{u})\mathbf{n}_I\}_\alpha \cdot [[\mathbf{v}]] - \{\boldsymbol{\sigma}(\mathbf{v})\mathbf{n}_I\}_\alpha \cdot [[\mathbf{u}]] \right] \mathrm{d}\Gamma \\
&= \sum_{k=1}^{2} \int_{\Omega^{(k)}} \mathbf{f}^{(k)} \cdot \mathbf{v}^{(k)} \, \mathrm{d}\Omega + \sum_{k=1}^{2} \int_{\Gamma_t^{(k)}} \mathbf{t}^{(k)} \cdot \mathbf{v}^{(k)} \, \mathrm{d}\Gamma \\
&\qquad\qquad + \sum_{k=1}^{2} \int_{\Gamma_u^{(k)}} \mathbf{g}^{(k)} \cdot \left[\gamma^{(k)} \, \mathbf{v}^{(k)} - \boldsymbol{\sigma}(\mathbf{v}^{(k)})\mathbf{n} \right] \mathrm{d}\Gamma \, .
\end{aligned}
$$

where $\gamma^{(k)} \in L^2(\Gamma_u^{(k)} \cup \Gamma_I)$, $\gamma^{(k)} > 0$, $k \in \{1,2\}$. For a given $\alpha : \Gamma_I \to [0,1]$, the averaging operator $\{\cdot\}_\alpha$ is defined as $\{q\}_\alpha = \alpha q^{(1)} + (1-\alpha) q^{(2)}$, so that in particular

$$
\{\boldsymbol{\sigma}(\mathbf{w})\mathbf{n}_I\}_\alpha = \left[\alpha \boldsymbol{\sigma}(\mathbf{w}^{(1)}) + (1-\alpha)\boldsymbol{\sigma}(\mathbf{w}^{(2)}) \right] \mathbf{n}_I .
$$

For discretization spaces $V_n^{(k)} \subset H^1(\Omega^{(k)})$, $k \in \{1,2\}$, it can be shown (see [14, Section 3.4.2] for a sketch in the case of the Poisson equation) that, independently of the choice of α, the bilinear form is coercive in $V_n^{(1)} \times V_n^{(2)}$ provided that there exists $\rho \in (0,1)$ such that

$$
\int_{\Gamma_u^{(k)} \cup \Gamma_I} \left(\gamma^{(k)} \right)^{-1} |\boldsymbol{\sigma}(\mathbf{v})\mathbf{n}|^2 \, \mathrm{d}\Gamma \le \rho^2 \int_{\Omega^{(k)}} \boldsymbol{\sigma}(\mathbf{v}) : \boldsymbol{\varepsilon}(\mathbf{v}) \, \mathrm{d}\Omega, \quad \forall \mathbf{v} \in V_n^{(k)}, \quad k \in \{1,2\},
$$

i.e. that each $\gamma^{(k)}$ satisfies the analogous constraint to (3.8) in its corresponding subdomain and for the corresponding function space. Different choices of α include the constant values 0, 0.5 and 1, and a further definition makes use of $\gamma^{(1)}$ and $\gamma^{(2)}$ as

$$\alpha^\star \coloneqq \frac{\gamma^{(2)}}{\gamma^{(1)} + \gamma^{(2)}}, \tag{3.10}$$

so that, in particular,

$$\{\gamma\}_{\alpha^\star} = \frac{2\,\gamma^{(1)}\gamma^{(2)}}{\gamma^{(1)} + \gamma^{(2)}},$$

and thus $\{\gamma\}_{\alpha^\star} \to 2\min_{k\in\{1,2\}} \gamma^{(k)}$ if $\max_{k\in\{1,2\}} \gamma^{(k)} \to \infty$.

Note that, in order to extend the formulation to different material properties for each domain, it suffices to replace $\boldsymbol{\sigma}\left(\mathbf{w}^{(k)}\right)$ by $\boldsymbol{\sigma}^{(k)}\left(\mathbf{w}^{(k)}\right)$, where $\mathbf{w}^{(k)}$ may be a test or trial function.

3.3 The Kirchhoff-Love shell model

A shell is a 3-dimensional solid volume with a thickness dimension that is significantly smaller than the other two. In our work, we will consider only shells with constant thickness, which we can describe via a parametrization of the midsurface, i.e. an open Lipschitz domain $\Xi \subset \mathbb{R}^2$ and a sufficiently regular physical map $\mathcal{F}\colon \Xi \to \mathbb{R}^3$ representing the midsurface of the shell. For more detailed derivations, we refer to [1, 3, 5, 17].

We begin by defining the covariant basis vectors $\mathbf{a}_i \coloneqq \partial_i \mathcal{F}$ for $i \in \{1, 2\}$ (in other words, the columns of $\nabla\mathcal{F}$), and the contravariant basis vectors $\mathbf{a}^i$ by the relationship $\mathbf{a}^i \cdot \mathbf{a}_j = \delta^i_j$. With that, we may define the unit normal vector to the midsurface as

$$\mathbf{a}_3 = \mathbf{a}^3 \coloneqq \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\|\mathbf{a}_1 \times \mathbf{a}_2\|},$$

and then the physical domain of the volumetric shell of thickness $h > 0$ as

$$\mathcal{B} \coloneqq \left\{ \mathcal{F}(\xi_1, \xi_2) + \zeta\, \mathbf{a}_3(\xi_1, \xi_2) : \ (\xi_1, \xi_2) \in \Xi,\ \zeta \in \left(-\frac{h}{2},\ \frac{h}{2}\right) \right\}.$$

We denote the midsurface $\Omega \equiv \mathcal{F}(\Xi)$. The Kirchhoff-Love shell model corresponds to a linear elastic model in $\mathcal{B}$ with certain assumptions on the displacement through the thickness, which are summarized in the ansatz

$$\mathbf{U}(\mathbf{x}) = \mathbf{u}(\xi_1, \xi_2) - \zeta\, \mathbf{a}_3(\xi_1, \xi_2) \cdot \underline{\nabla}\mathbf{u}(\xi_1, \xi_2), \quad \mathbf{x} = \mathcal{F}(\xi_1, \xi_2) + \zeta\, \mathbf{a}_3(\xi_1, \xi_2) \in \mathcal{B}, \tag{3.11}$$

with $\mathbf{u}\colon \Xi \to \mathbb{R}^3$ the displacement at the midsurface, and $\underline{\nabla}$ the surface gradient operator, which acts on a vector field $\mathbf{w}\colon \Xi \to \mathbb{R}^3$ as

$$\underline{\nabla}\mathbf{w} = \frac{\partial \mathbf{w}}{\partial \xi_\lambda} \otimes \mathbf{a}^\lambda.$$

The ansatz (3.11), when introduced in the load-free energy functional of linear elasticity, and after discarding terms of order higher than h^3, leads to

$$\frac{1}{2}\int_{\mathcal{B}} \boldsymbol{\sigma}(\mathbf{U}) : \boldsymbol{\varepsilon}(\mathbf{U}) \, \mathrm{d}\mathcal{B} \approx \frac{1}{2}\int_{\Omega} A(\mathbf{u}) : \alpha(\mathbf{u}) \, \mathrm{d}\Omega + \frac{1}{2}\int_{\Omega} B(\mathbf{u}) : \beta(\mathbf{u}) \, \mathrm{d}\Omega,$$

where α and β are, respectively, the membrane and bending strains, defined by

$$\alpha(\mathbf{u}) \coloneqq \mathrm{sym}(P \underline{\nabla}\mathbf{u}), \quad \beta(\mathbf{u}) \coloneqq -\mathrm{sym}(\mathbf{a}_3 \cdot \underline{\nabla}\,\underline{\nabla}\mathbf{u}), \tag{3.12}$$

with $P = \mathbb{I} - \mathbf{a}_3 \otimes \mathbf{a}_3$ the projector onto the midsurface's tangent plane. A and B are, respectively, the membrane and bending stresses, defined from the corresponding strains as

$$A(\mathbf{u}) \coloneqq h\,\mathbb{C} : \alpha(\mathbf{u}), \quad B(\mathbf{u}) \coloneqq \frac{h^3}{12}\,\mathbb{C} : \beta(\mathbf{u}),$$

where the stiffness tensor $\mathbb{C}$ is inherited from the original solid model. It is usual to incorporate the plane stress assumption, which for isotropic linear elasticity leads to $\mathbb{C}$ acting on the strains as

$$\mathbb{C} : \mathbb{S} = \frac{E}{1-\nu^2}\left((1-\nu)\,\mathbb{S} + \nu\ \mathrm{tr}(\mathbb{S})\,\mathbb{I}\right). \tag{3.13}$$

The thickness-related factors in the membrane and bending stresses arise respectively from

$$h = \int_{-h/2}^{h/2} \mathrm{d}\zeta, \quad \frac{h^3}{12} = \int_{-h/2}^{h/2} \zeta^2 \, \mathrm{d}\zeta.$$

Nitsche's method for Kirchhoff-Love shell problems has been derived already as rigorously as possible by Benzaken et al. [1], from whom we borrow most of the notation, and to whom we refer for a more detailed derivation. Notably, for shell problems there are three different kinds of essential boundary conditions (contrary to the single kind of Dirichlet boundary conditions in linear elasticity), as revealed by the bilinear form with consistency terms:

$$\begin{aligned}\int_{\Omega} A(\mathbf{u}) : \alpha(\mathbf{v}) \, \mathrm{d}\Omega + \int_{\Omega} B(\mathbf{u}) : \beta(\mathbf{v}) \, \mathrm{d}\Omega \\ - \int_{\partial\Omega} \mathbf{T}(\mathbf{u}) \cdot \mathbf{v} \, \mathrm{d}\Gamma - \int_{\partial\Omega} B_{nn}(\mathbf{u})\theta_n(\mathbf{v}) \, \mathrm{d}\Gamma - \sum_{p\in\chi} [\![B_{nt}(\mathbf{u})]\!](\mathbf{v}\cdot\mathbf{a}_3)\Big|_p,\end{aligned} \tag{3.14}$$

for $\mathbf{t}$ and $\mathbf{n}$ the boundary tangent and outside normal, and where $B_{nn}(\mathbf{v}) = \mathbf{n}\cdot(B(\mathbf{v})\mathbf{n})$ and $B_{nt}(\mathbf{v}) = \mathbf{n}\cdot(B(\mathbf{v})\mathbf{t})$ are the so-called *bending* and *twisting* moments, $\theta_n(\mathbf{v}) = -(\mathbf{a}_3\cdot\underline{\nabla}\mathbf{v})\cdot\mathbf{n}$ is the *normal rotation*, χ is the set of all corners in $\partial\Omega$, and

$$\mathbf{T}(\mathbf{v}) = A(\mathbf{v})\mathbf{n} - b\,(B(\mathbf{v})\mathbf{n} + \mathbf{t}B_{nt}(\mathbf{v})) + \left[\left(\underline{\nabla}\cdot B(\mathbf{v})\right)\cdot\mathbf{n} + (\underline{\nabla}B_{nt}(\mathbf{v}))\cdot\mathbf{t}\right]\mathbf{a}_3 \tag{3.15}$$

is the so-called *ersatz force*, with $b = -\underline{\nabla}\mathbf{a}_3$ the second fundamental form, c.f. [1, (3.7)]. Note that $[\![B_{nt}(\mathbf{v})]\!]$ denotes a jump in the twisting moment, i.e. its value for $\mathbf{t}$ and $\mathbf{n}$ *after* the corner minus its value for $\mathbf{t}$ and $\mathbf{n}$ *before* it (with an ordering given precisely by the boundary tangent $\mathbf{t}$).

Thus, Nitsche's method for the Kirchhoff-Love shell model consists in finding $\mathbf{u} \in V_n$ such that

$$a^S(\mathbf{u},\mathbf{v}) = \ell^S(\mathbf{v}) \quad \forall \mathbf{v} \in V_n,$$

with the bilinear form

$$
\begin{aligned}
a^S(\mathbf{u},\mathbf{v}) \coloneqq & \int_\Omega A(\mathbf{u}) \colon \alpha(\mathbf{v}) \, \mathrm{d}\Omega + \int_\Omega B(\mathbf{u}) \colon \beta(\mathbf{v}) \, \mathrm{d}\Omega \\
& + \int_{\Gamma_u} [\gamma_u \mathbf{u} \cdot \mathbf{v} - \mathbf{T}(\mathbf{u}) \cdot \mathbf{v} - \mathbf{T}(\mathbf{v}) \cdot \mathbf{u}] \, \mathrm{d}\Gamma \\
& + \int_{\Gamma_\theta} [\gamma_\theta \theta_n(\mathbf{u}) \theta_n(\mathbf{v}) - B_{nn}(\mathbf{u}) \theta_n(\mathbf{v}) - B_{nn}(\mathbf{v}) \theta_n(\mathbf{u})] \, \mathrm{d}\Gamma \\
& + \sum_{p \in \chi_u} [\gamma_C (\mathbf{u} \cdot \mathbf{a}_3)(\mathbf{v} \cdot \mathbf{a}_3) - [\![B_{nt}(\mathbf{u})]\!](\mathbf{v} \cdot \mathbf{a}_3) - [\![B_{nt}(\mathbf{v})]\!](\mathbf{u} \cdot \mathbf{a}_3)] \Big|_p ,
\end{aligned} \tag{3.16}
$$

and the linear form

$$
\begin{aligned}
\ell^S(\mathbf{v}) \coloneqq & \int_\Omega \mathbf{f} \cdot \mathbf{v} \, \mathrm{d}\Omega + \int_{\partial\Omega \setminus \Gamma_u} \hat{\mathbf{T}} \cdot \mathbf{v} \, \mathrm{d}\Gamma + \int_{\partial\Omega \setminus \Gamma_\theta} \theta_n(\mathbf{v}) \hat{B}_{nn} \, \mathrm{d}\Gamma + \sum_{p \in \mathcal{X} \setminus \chi_u} (\mathbf{v} \cdot \mathbf{a}_3) \Upsilon \Big|_p \\
& + \int_{\Gamma_u} \hat{\mathbf{u}} \cdot [\gamma_u \mathbf{v} - \mathbf{T}(\mathbf{v})] \, \mathrm{d}\Gamma \\
& + \int_{\Gamma_\theta} \hat{\theta}_n \, [\gamma_\theta \theta_n(\mathbf{v}) - B_{nn}(\mathbf{v})] \, \mathrm{d}\Gamma \\
& + \sum_{p \in \chi_u} \hat{u}_n \, [\gamma_C (\mathbf{v} \cdot \mathbf{a}_3) - [\![B_{nt}(\mathbf{v})]\!]] \Big|_p ,
\end{aligned}
$$

where Γ_u is the boundary part where the displacement is constrained (to $\hat{\mathbf{u}}$), Γ_θ that where the normal rotation is (to $\hat{\theta}_n$), and χ_u the set of corners where the normal displacement is (to $\hat{u}_n$). Finally, $\hat{\mathbf{T}}$ is a prescribed ersatz force on $\partial\Omega \setminus \Gamma_u$, $\hat{B}_{nn}$ a prescribed bending moment on $\partial\Omega \setminus \Gamma_\theta$, and Υ a prescribed jump of the twisting moment at each unconstrained corner.

After extracting the volume term from the bilinear form,

$$
\hat{a}^S(\mathbf{u},\mathbf{v}) \coloneqq \int_\Omega A(\mathbf{u}) \colon \alpha(\mathbf{v}) \, \mathrm{d}\Omega + \int_\Omega B(\mathbf{u}) \colon \beta(\mathbf{v}) \, \mathrm{d}\Omega,
$$

the condition for coercivity of $a^S \cdot V_n \times V_n \to \mathbb{R}$ in this case reads

$$
\begin{aligned}
\exists \rho \in (0,1) \quad : \quad & \left\| \gamma_u^{-\frac{1}{2}} |\mathbf{T}(\mathbf{v})| \right\|^2_{L^2(\Gamma_u)} + \left\| \gamma_\theta^{-\frac{1}{2}} B_{nn}(\mathbf{v}) \right\|^2_{L^2(\Gamma_\theta)} + \sum_{p \in \chi_u} \gamma_C^{-1} [\![B_{nt}(\mathbf{v})]\!]^2 \\
& \leq \rho^2 \hat{a}^S(\mathbf{v},\mathbf{v}), \quad \forall \mathbf{v} \in V_n,
\end{aligned} \tag{3.17}
$$

which can be made independent for each stabilization function for example by bounding each of the three terms in the l.h.s. by $\frac{\rho^2}{3} \hat{a}^S(\mathbf{v},\mathbf{v})$. Note further that γ_u can be split into *in-plane* (surface-tangential) and *out-of-plane* (surface-normal) components, each of which would exhibit a different scaling with respect to refinement (following an analogous splitting of the ersatz force, see [1, (3.7)]). A rough but simple alternative is to use a single stabilization function $\gamma_u = \gamma_\theta = \gamma_C$.

Remark 3.1. Note that the boundary integrals in (3.14) can be rewritten through integration by parts to remove the corner terms, leading to

$$
\begin{aligned}
(3.18) \qquad & -\int_{\partial\Omega} \mathbf{T}(\mathbf{u}) \cdot \mathbf{v} \, \mathrm{d}\Gamma - \int_{\partial\Omega} B_{nn}(\mathbf{u})\theta_n(\mathbf{v}) \, \mathrm{d}\Gamma - \sum_{p\in\chi} [\![B_{nt}(\mathbf{u})]\!](\mathbf{v}\cdot\mathbf{a}_3)\Big|_p = \\
& = -\int_{\partial\Omega} \tilde{\mathbf{T}}(\mathbf{u}) \cdot \mathbf{v} \, \mathrm{d}\Gamma - \int_{\partial\Omega} B_{nn}(\mathbf{u})\theta_n(\mathbf{v}) \, \mathrm{d}\Gamma - \int_{\partial\Omega} B_{nt}(\mathbf{u})\theta_t(\mathbf{v}) \, \mathrm{d}\Gamma = \\
& \qquad = -\int_{\partial\Omega} \tilde{\mathbf{T}}(\mathbf{u}) \cdot \mathbf{v} \, \mathrm{d}\Gamma - \int_{\partial\Omega} (B(\mathbf{u})\mathbf{n}) \cdot \boldsymbol{\theta}(\mathbf{v}) \, \mathrm{d}\Gamma
\end{aligned}
$$

where $\boldsymbol{\theta}(\mathbf{v}) = -\mathbf{a}_3 \cdot \underline{\nabla}\mathbf{v}$, with $\theta_t(\mathbf{v}) = \boldsymbol{\theta}(\mathbf{v}) \cdot \mathbf{t}$ the *tangential rotation*, and

$$
\tilde{\mathbf{T}}(\mathbf{u}) \coloneqq A(\mathbf{u})\mathbf{n} - b\, B(\mathbf{u})\mathbf{n} + \left[\left(\underline{\nabla} \cdot B(\mathbf{u})\right) \cdot \mathbf{n}\right] \mathbf{a}_3,
$$

which we may call *reduced ersatz force*, since it is part of the original ersatz force (3.15). Note that $\mathbf{u}$ and $\theta_t(\mathbf{u})$ are not independent, and any simultaneous boundary constraint on both quantities should be consistent.

3.4 Solid-shell blending

A case of particular interest consists of coupling a Kirchhoff-Love shell with another shell considered as a solid linear elastic body, as in the simple example of Figure 3.1. In doing so, we can use the more expensive linear elasticity model in certain shell parts of particular interest, where the displacement field may not satisfy the Kirchoff-Love assumptions encapsulated in the ansatz (3.11).

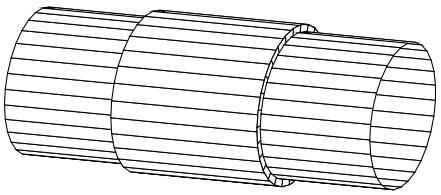


Figure 3.1: Cylindrical structure composed of a solid part and two shell parts.

For simplicity, let us assume that the solid-shell interface corresponds to a closed curve in the parameter space of the shell part. Alternatively, we may assume that all shell boundaries rely on the boundary condition formulation from Remark 3.1 (i.e. without corner terms). Let us denote the 3-dimensional side of the interface as $\Gamma_U^\star$ and its 2-dimensional boundary part of the shell's midsurface as Γ_U. With the assumption of a sufficiently thin shell with a gentle curvature, we may consider integrals over $\Gamma_U^\star$ equivalent to integrals over $\Gamma_U \times (-h/2, h/2)$ (as it is also done to derive the volume integrals of the shell model). Then, from the boundary integrals in (3.18), we may find the conjugate quantity to the boundary displacement through-the-thickness $\mathbf{U} \coloneqq \mathbf{u} + \zeta\, \boldsymbol{\theta}(\mathbf{u})$, which turns out to be

$$
Q(\mathbf{u};\mathbf{x}) = \frac{1}{h}\tilde{\mathbf{T}}(\mathbf{u}) + \frac{12}{h^3}\, \zeta\, B(\mathbf{u})\mathbf{n}\,,
$$

since, as it can be easily checked

$$
-\int_{\Gamma_U} \tilde{\mathbf{T}}(\mathbf{u}) \cdot \mathbf{v} \, \mathrm{d}\Gamma - \int_{\Gamma_U} (B(\mathbf{u})\mathbf{n}) \cdot \boldsymbol{\theta}(\mathbf{v}) \, \mathrm{d}\Gamma = -\int_{\Gamma_U} \int_{-h/2}^{h/2} \boldsymbol{Q}(\mathbf{u};\mathbf{x}) \cdot [\mathbf{v} + \zeta\, \boldsymbol{\theta}(\mathbf{v})] \, \mathrm{d}\zeta \, \mathrm{d}\Gamma\,.
$$

We may couple $\mathcal{Q}(\mathbf{u};\mathbf{x})$ directly with the boundary traction from the solid part, i.e. $\boldsymbol{\sigma}(\mathbf{U}_{\text{solid}})\mathbf{n}$, but if we look carefully at it, we realize that the two quantities do not match physically. Indeed, if we introduce $\mathbf{U} = \mathbf{u} + \zeta\,\boldsymbol{\theta}(\mathbf{u})$ in the expression for the solid traction, we find that

$$\boldsymbol{\sigma}(\mathbf{U})\mathbf{n} = \frac{1}{h}A(\mathbf{u})\mathbf{n} + \frac{12}{h^3}\,\zeta\,B(\mathbf{u})\mathbf{n} =: \hat{\mathcal{Q}}(\mathbf{u};\mathbf{x}),$$

which is the shell quantity that is physically equivalent to the solid boundary traction. This leaves a lingering boundary term among the shell integrals, namely

$$\begin{aligned}
&\int_{\Gamma_U}\int_{-h/2}^{h/2}\left[\hat{\mathcal{Q}}(\mathbf{u};\mathbf{x}) - \mathcal{Q}(\mathbf{u};\mathbf{x})\right]\cdot\left[\mathbf{v} + \zeta\,\boldsymbol{\theta}(\mathbf{v})\right]\,\mathrm{d}\zeta\,\mathrm{d}\Gamma = \\
&= \int_{\Gamma_U}\int_{-h/2}^{h/2}\frac{1}{h}\left[A(\mathbf{u})\mathbf{n} - \tilde{\mathbf{T}}(\mathbf{u})\right]\cdot\mathbf{v}\,\mathrm{d}\zeta\,\mathrm{d}\Gamma = \int_{\Gamma_U}\left(b\,(B(\mathbf{u})\mathbf{n}) - \left[\left(\underline{\nabla}\cdot B(\mathbf{u})\right)\cdot\mathbf{n}\right]\mathbf{a}_3\right)\cdot\mathbf{v}\,\mathrm{d}\Gamma,
\end{aligned}$$

and thus we are forced to pick one of two unsatisfactory approaches: we either treat the above integral as a homogeneous natural boundary condition for $\left[\left(\underline{\nabla}\cdot B(\mathbf{u})\right)\cdot\mathbf{n}\right]\mathbf{a}_3 - b\,(B(\mathbf{u})\mathbf{n})$ on Γ_U, or we simply couple $\mathcal{Q}(\mathbf{u};\mathbf{x})$ to the solid boundary traction, leaving no space for said natural boundary condition. We choose the second approach.

Given that, the corresponding stabilization function γ_U for the penalty term

$$\int_{\Gamma_U}\gamma_U\int_{-h/2}^{h/2}\left[\mathbf{u} + \zeta\,\boldsymbol{\theta}(\mathbf{u})\right]\cdot\left[\mathbf{v} + \zeta\,\boldsymbol{\theta}(\mathbf{v})\right]\,\mathrm{d}\zeta\,\mathrm{d}\Gamma\,,$$

with boundary moments based on $\mathcal{Q}(\cdot;\mathbf{x})$, has to satisfy the trace inequality

$$\exists\rho\in(0,1)\quad:\quad\int_{\Gamma_U}\gamma_U^{-1}\int_{-h/2}^{h/2}|\mathcal{Q}(\mathbf{v};\mathbf{x})|^2\,\mathrm{d}\zeta\,\mathrm{d}\Gamma \le \rho^2\hat{a}^S(\mathbf{v},\mathbf{v}),\quad\forall\mathbf{v}\in V_n,$$

where integration over ζ yields

$$\exists\rho\in(0,1)\quad:\quad\int_{\Gamma_U}\gamma_U^{-1}\left(\frac{1}{h}\left|\tilde{\mathbf{T}}(\mathbf{v})\right|^2 + \frac{12}{h^3}\,|B(\mathbf{v})\mathbf{n}|^2\right)\mathrm{d}\Gamma \le \rho^2\hat{a}^S(\mathbf{v},\mathbf{v}),\quad\forall\mathbf{v}\in V_n. \tag{3.19}$$

Ignoring boundary conditions on both sides, the Nitsche formulation can then be written as

$$\begin{aligned}
&\int_{\Omega_{\text{solid}}}\boldsymbol{\sigma}(\mathbf{u}_{\text{solid}}):\boldsymbol{\varepsilon}(\mathbf{v}_{\text{solid}})\,\mathrm{d}\Omega + \hat{a}^S(\mathbf{u}_{\text{shell}},\mathbf{v}_{\text{shell}}) \\
&+\int_{\Gamma_U^\star}\left[\alpha\gamma_{\text{solid}} + (1-\alpha)\gamma_{\text{shell}}\right]\left[\mathbf{u}_{\text{solid}} - \mathcal{S}(\mathbf{u}_{\text{shell}})\right]\cdot\left[\mathbf{v}_{\text{solid}} - \mathcal{S}(\mathbf{v}_{\text{shell}})\right]\,\mathrm{d}\Gamma^\star \\
&-\int_{\Gamma_U^\star}\left[\alpha\boldsymbol{\sigma}(\mathbf{u}_{\text{solid}})\mathbf{n} - (1-\alpha)\mathcal{Q}(\mathbf{u}_{\text{shell}};\mathbf{x})\right]\cdot\left[\mathbf{v}_{\text{solid}} - \mathcal{S}(\mathbf{v}_{\text{shell}})\right]\,\mathrm{d}\Gamma^\star \\
&-\int_{\Gamma_U^\star}\left[\alpha\boldsymbol{\sigma}(\mathbf{v}_{\text{solid}})\mathbf{n} - (1-\alpha)\mathcal{Q}(\mathbf{v}_{\text{shell}};\mathbf{x})\right]\cdot\left[\mathbf{u}_{\text{solid}} - \mathcal{S}(\mathbf{u}_{\text{shell}})\right]\,\mathrm{d}\Gamma^\star \\
&\qquad\qquad = \int_{\Omega_{\text{solid}}}\mathbf{f}_{\text{solid}}\cdot\mathbf{v}_{\text{solid}}\,\mathrm{d}\Omega + \int_{\Omega_{\text{shell}}}\mathbf{f}_{\text{shell}}\cdot\mathbf{v}_{\text{shell}}\,\mathrm{d}\Omega,
\end{aligned}$$

where we denote $\mathcal{S}(\mathbf{u}_{\text{shell}}) = \mathbf{u}_{\text{shell}} + \zeta\, \boldsymbol{\theta}\, (\mathbf{u}_{\text{shell}})$ (and analogously for test functions), and where $\mathbf{n}$ is the surface normal on $\Gamma_U^\star$ pointing from the solid to the shell part. Note that within the shell quantities $Q(\cdot; \mathbf{x})$ we use the reverse normal. As in Section 3.2, we can pick any $\alpha\colon \Gamma_U^\star \to [0, 1]$, and in particular a γ-averaging combination like (3.10) for γ_{solid} and γ_{shell}, i.e.

$$\alpha^\star = \frac{\gamma_{\text{shell}}}{\gamma_{\text{solid}} + \gamma_{\text{shell}}}. \tag{3.20}$$

Note that γ_{solid} has to satisfy (3.8) for $\Gamma_U^\star$, while γ_{shell} inherits the role of γ_U in our derivations above and thus has to satisfy (3.19).

In our practical implementation, in order to evaluate quantities pertaining to the shell part appearing in the integrals over $\Gamma_U^\star$ (the trial and test functions, but also γ_{shell} and the shell parametrization with its required derivatives), for every point $\mathbf{x} \equiv (\bar{\mathbf{x}}, \zeta) \in \Gamma_U^\star \equiv \Gamma_U \times [-h/2, h/2]$ we need to retrieve the parameter associated to $\bar{\mathbf{x}}$ in the shell parametrization. A sketch of the map between parameter domains is provided in Figure 3.2 for the simplest case of two rectangular parameter domains. A more complicated case with trimmed parameter domains is illustrated in Figure 3.3. Since the ζ variable from the solid parametrization plays no role in this geometric coupling, we restrict our illustration to the coupling of midsurface parametrizations (i.e. φ_1 denotes the parametrization of the midsurface of the solid part).

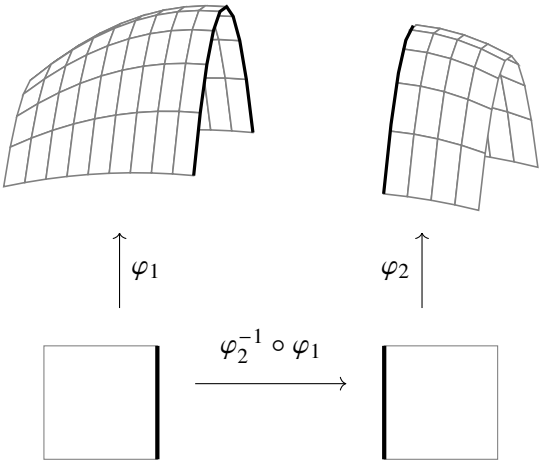


Figure 3.2: Map between the coupled boundary parts of two shell parametrizations. For visualization purposes, the two shells are depicted separately in physical space, but they are understood to coincide at the interface.

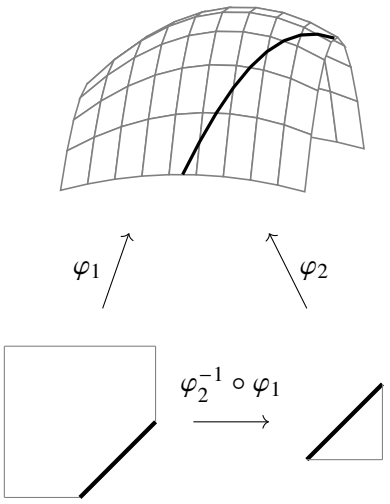


Figure 3.3: Shell coupling with trimmed parameter spaces.

Since we work exclusively with NURBS parametrizations, we have implemented the action of φ_2^{-1} via two NURBS projection algorithms (for curves and surfaces). The details regarding this work, which was by no means trivial, are provided in Appendix A.

3.5 PUM-based stabilization function

After having introduced Nitsche's method for a variety of different problems, it remains to show how to properly use it in combination with the PUM. For this, the main ingredient is the definition of the stabilization function, which we undertook in [14] and which we will briefly restate here. The paper is attached in Appendix C.1.

For simplicity, let us consider the case of a single body in the regime of linear elasticity as described in Section 3.1. We seek to construct a stabilization function γ fulfilling the trace inequality (3.8) for $\mathbf{v} \in V^{\mathrm{PU}} = \sum_{j=0}^{n} \varphi_j V_j$. For another PU $\{\psi_i\}_{i=0}^{m}$, we may construct the stabilization function as

$$\gamma = \sum_{i=0}^{m} \gamma_{(i)} \psi_i,$$

with $\gamma_{(i)}$ being constant coefficients. Since γ only intervenes at the Dirichlet boundary Γ_u, we may ignore basis functions whose support does not intersect Γ_u (and their associated coefficients).

As shown in [14, Section 3.3] for the Poisson equation, a sufficient condition for (3.8) to hold for this definition of γ is that $\gamma_{(i)} > C_i$, where each C_i satisfies

$$\int_{\Gamma_u} \psi_i |\boldsymbol{\sigma}(\mathbf{v})\mathbf{n}|^2 \, \mathrm{d}\Gamma \leq C_i \int_{\Omega} \psi_i \boldsymbol{\sigma}(\mathbf{v}) : \boldsymbol{\varepsilon}(\mathbf{v}) \, \mathrm{d}\Omega, \quad \forall \mathbf{v} \in V^{\mathrm{PU}}.$$

Trivially, we can reduce the space considered for each C_i to

$$\breve{V}_i = \sum_{j \in \mathcal{N}_i} \varphi_j V_j, \quad \mathcal{N}_i \coloneqq \{j : \ \mathrm{supp}(\varphi_j) \cap \mathrm{supp}(\psi_i) \cap \Gamma_u \neq \emptyset\},$$

i.e. each constant C_i must satisfy

$$\int_{\Gamma_u} \psi_i |\boldsymbol{\sigma}(\mathbf{v})\mathbf{n}|^2 \, \mathrm{d}\Gamma \leq C_i \int_{\Omega} \psi_i \boldsymbol{\sigma}(\mathbf{v}) : \boldsymbol{\varepsilon}(\mathbf{v}) \, \mathrm{d}\Omega, \quad \forall \mathbf{v} \in \breve{V}_i,$$

which can be understood as a local generalized eigenvalue problem (gEVP) for each constant C_i. The coefficients $\gamma_{(i)}$ may then be chosen simply as $\gamma_{(i)} = \beta C_i$ for some $\beta > 1$, with $\beta = 2$ being a standard choice in the finite element approach. Naturally, the C_i (and thus the $\gamma_{(i)}$) depend on the function space V^{PU}.

This formulation can be easily adapted to any trace inequality that a stabilization function may have to satisfy, and generalizes the element-wise construction of stabilization functions that is commonly used in the FEM.

Additionally, in [14, Section 2.3] we introduced a patch-aggregation approach designed to remove the possibility of arbitrarily large coefficients in the stabilization functions, which may otherwise happen for patches whose so-called flat-top region intersects with the domain in some degenerate manner.

Remark 3.2. Computations can be simplified if we choose γ's PU to be based on the same cover as that of V^{PU}, i.e. $n = m$ and $\mathrm{supp}(\psi_i) = \mathrm{supp}(\varphi_i)$. We could simply pick $\{\psi_i\}_{i=0}^n = \{\varphi_i\}_{i=0}^n$, but we explicitly allow the use of a different PU because γ only needs to be piecewise-constant, and thus the regularity requirements on its basis functions are lower than those of V^{PU}.

Remark 3.3. In the paper we also introduced a *purely local* approach for the computation of stabilization functions, which was a non-rigorous approach aimed at reducing computational cost. Since then, we have significantly improved the efficiency of numerical integration in PUMA, rendering this non-rigorous approach worthless. Therefore, we kindly ask the reader to ignore it.

3.6 Multilevel issues

The use of a stabilization function in Nitsche's method, whose values forcibly depend on the employed discretization space, does not combine well with multilevel preconditioners (as those introduced for the PUM in [6]). Let us illustrate the case with the Poisson problem

$$\text{find} \quad u \in V^{\mathrm{PU}} \quad : \quad a_\gamma(v,u) = \ell_\gamma(v), \quad \forall v \in V^{\mathrm{PU}},$$

with bilinear and linear forms

$$\begin{aligned} a_\gamma(v,u) &\coloneqq \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}\Omega + \int_{\partial\Omega} [\gamma uv - (\nabla u \cdot \mathbf{n})v - (\nabla v \cdot \mathbf{n})u] \,\mathrm{d}\Gamma, \\ \ell_\gamma(v) &= \int_\Omega fv \,\mathrm{d}\Omega + \int_{\partial\Omega} (\gamma v - \nabla v \cdot \mathbf{n})g \,\mathrm{d}\Gamma, \end{aligned}$$

and a two-level preconditioner with coarse and fine level spaces V_c and V_f, respectively. Let γ_f be a suitable stabilization function for the fine level, i.e. fulfilling the trace inequality

$$\exists \rho \in (0,1) \quad : \quad \int_{\partial\Omega} \gamma_f^{-1} (\nabla v \cdot \mathbf{n})^2 \,\mathrm{d}\Gamma \leq \rho^2 \int_\Omega |\nabla v|^2 \,\mathrm{d}\Omega, \quad \forall v \in V_f,$$

which makes $a_{\gamma_f} : V_f \times V_f \to \mathbb{R}$ coercive. We then have two options to define a bilinear form for the coarse level: by restriction of the fine-level form as

$$a_{\gamma_f}^G(v,u) \coloneqq a_{\gamma_f}(Pv, Pu), \quad u, v \in V_c$$

(which in matrix form is usually called the Galerkin triple product), with $P : V_c \to V_f$ the prolongation operator, or directly from the Poisson problem as a_{γ_c}, with γ_c fulfilling

$$\exists \rho \in (0,1) \quad : \quad \int_{\partial\Omega} \gamma_c^{-1} (\nabla v \cdot \mathbf{n})^2 \,\mathrm{d}\Gamma \leq \rho^2 \int_\Omega |\nabla v|^2 \,\mathrm{d}\Omega, \quad \forall v \in V_c.$$

In the first case, $a_{\gamma_f}^G$ can be guaranteed to be coercive if $P : V_c \hookrightarrow V_f$, which for multilevel PUM is however not the case, as $V_c \not\subset V_f$ in general. In the second case, the coarse-level correction $u_c \in V_c$ for a fine-level solution $u_f \in V_f$ solves

$$a_{\gamma_c}(v, u_c) = \ell_{\gamma_f}(Pv) - a_{\gamma_f}(Pv, u_f), \quad \forall v \in V_c,$$

which can be expanded to

$$\int_\Omega \nabla u_c \cdot \nabla v \,\mathrm{d}\Omega - \int_{\partial\Omega} (\nabla u_c \cdot \mathbf{n}) v \,\mathrm{d}\Gamma + \int_{\partial\Omega} u_c \left[\gamma_c v - (\nabla v \cdot \mathbf{n})\right] \mathrm{d}\Gamma =$$
$$\int_\Omega \left[(Pv)\, f - \nabla u_f \cdot \nabla(Pv)\right] \mathrm{d}\Omega + \int_{\partial\Omega} (\nabla u_f \cdot \mathbf{n})(Pv) \,\mathrm{d}\Gamma$$
$$+ \int_{\partial\Omega} (g - u_f) \left[\gamma_f (Pv) - \nabla(Pv) \cdot \mathbf{n}\right] \mathrm{d}\Gamma.$$

For an injective P, this corresponds to solving $-\Delta u_c = f + \Delta u_f$ with boundary conditions $u_c = g - u_f$, which would however be affected by a mismatch between γ_f and γ_c (unless $u_f = g$ at the boundary to begin with). The fact that P is not injective in the multilevel PUM (and thus $Pv \neq v$ in general) is nevertheless independent of Nitsche's method.

In both cases, relatively simple fixes can be conceived. In the first one, if we construct a stabilization function $\hat{\gamma}_c$ fulfilling

$$\exists \rho \in (0,1) \quad : \quad \int_{\partial\Omega} \hat{\gamma}_c^{-1} \left(\nabla(Pv) \cdot \mathbf{n}\right)^2 \mathrm{d}\Gamma \leq \rho^2 \int_\Omega |\nabla(Pv)|^2 \,\mathrm{d}\Omega, \quad \forall v \in V_c,$$

and choose some $\hat{\gamma} \geq \max(\gamma_f, \hat{\gamma}_c)$, then $a_{\hat{\gamma}} \colon V_f \times V_f \to \mathbb{R}$ as well as $a^G_{\hat{\gamma}} \colon V_c \times V_c \to \mathbb{R}$ will both be coercive. However, this approach is slightly impractical and cumbersome to implement in the PUMA library, and we have not pursued it.

In the second case, we may simply replace both γ_f and γ_c by some $\hat{\gamma} \geq \max(\gamma_f, \gamma_c)$. This, however, enforces the largest stabilization function to be used for all levels of a multilevel sequence, which makes coarse-level operators ill-conditioned. Furthermore, usually the largest stabilization function will be that of the finest level, and thus this approach would not be substantially different than using Galerkin triple products to propagate the finest-level operator to all coarser levels. For this reason, we have also not implemented this fix.

These issues will be studied in some of the examples from Chapter 6.

4 Laminated composite materials

To leave behind the simple case of isotropic linear elasticity, we will now consider orthotropic linear elasticity, which is commonly used to model fiber-reinforced composite materials. Additionally, multiple layers with different material orientations can be combined in the form of laminated composites, which provides an interesting use case for us because, by design, such laminated composites usually possess a plate or shell structure.

4.1 Orthotropic linear elasticity

The orthotropic model of linear elasticity deviates from the isotropic case (3.4) through the constitutive relation

$$\begin{bmatrix} \varepsilon_{00} \\ \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{01} \\ \varepsilon_{12} \\ \varepsilon_{02} \end{bmatrix} = \begin{bmatrix} \frac{1}{E_0} & -\frac{\nu_{10}}{E_1} & -\frac{\nu_{20}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{01}}{E_0} & \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{02}}{E_0} & -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2G_{01}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2G_{12}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2G_{02}} \end{bmatrix} \begin{bmatrix} \sigma_{00} \\ \sigma_{11} \\ \sigma_{22} \\ \sigma_{01} \\ \sigma_{12} \\ \sigma_{02} \end{bmatrix},$$

where the E_α are Young's moduli in each dimension, $G_{\alpha\beta}$ are the shear moduli, and $\nu_{\alpha\beta}$ are Poisson's ratios, all given with respect to some orthonormal coordinate system, and fulfilling the symmetry relations $\nu_{\beta\alpha}E_\alpha = \nu_{\alpha\beta}E_\beta$.

Accordingly, defining $\delta = 1 - \nu_{01}\nu_{10} - \nu_{12}\nu_{21} - \nu_{20}\nu_{02} - 2\nu_{01}\nu_{12}\nu_{21}$, we may write

$$\begin{bmatrix} \sigma_{00} \\ \sigma_{11} \\ \sigma_{22} \\ \sigma_{01} \\ \sigma_{12} \\ \sigma_{02} \end{bmatrix} = \begin{bmatrix} \frac{1-\nu_{12}\nu_{21}}{\delta}E_0 & \frac{\nu_{10}+\nu_{12}\nu_{20}}{\delta}E_0 & \frac{\nu_{20}+\nu_{21}\nu_{10}}{\delta}E_0 & 0 & 0 & 0 \\ \frac{\nu_{01}+\nu_{02}\nu_{21}}{\delta}E_1 & \frac{1-\nu_{02}\nu_{20}}{\delta}E_1 & \frac{\nu_{21}+\nu_{20}\nu_{01}}{\delta}E_1 & 0 & 0 & 0 \\ \frac{\nu_{02}+\nu_{01}\nu_{12}}{\delta}E_2 & \frac{\nu_{12}+\nu_{10}\nu_{02}}{\delta}E_2 & \frac{1-\nu_{01}\nu_{10}}{\delta}E_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2G_{01} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2G_{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 2G_{02} \end{bmatrix} \begin{bmatrix} \varepsilon_{00} \\ \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{01} \\ \varepsilon_{12} \\ \varepsilon_{02} \end{bmatrix},$$

To represent fiber-reinforced composite materials, we may consider a particular orthotropic model, namely that with a single privileged direction and an isotropic material response in the plane orthogonal to it. By identifying the privileged direction with index 0 above, this transversely-isotropic linear elasticity model can be subsumed in the assumptions

$$E_1 = E_2, \quad \nu_{12} = \nu_{21}, \quad \nu_{01} = \nu_{02}, \quad \nu_{10} = \nu_{20}, \quad G_{01} = G_{02}, \quad G_{12} = \frac{E_1}{2(1+\nu_{12})}. \tag{4.1}$$

4.2 Three-dimensional discretization

Drawing from the notation we used in Section 3.3, we will describe a laminated shell with n laminae and total thickness $h > 0$ via a parameter domain $\Xi \in \mathbb{R}^2$ and a physical map $\mathcal{F} : \Xi \to \mathbb{R}^3$ for the midsurface, leading to the physical domain

$$\mathcal{B} \coloneqq \left\{ \mathcal{F}(\xi_1, \xi_2) + \zeta\, \mathbf{a}_3(\xi_1, \xi_2) : \ (\xi_1, \xi_2) \in \Xi,\ \zeta \in \left(-\frac{h}{2}, \frac{h}{2}\right) \right\},$$

but this time with a series of material interfaces through the thickness, at ζ-positions $\{\zeta_k\}_{k=1}^{n-1}$ such that

$$-\frac{h}{2} =: \zeta_0 < \zeta_1 < \zeta_2 < \cdots < \zeta_{n-1} < \zeta_n \coloneqq \frac{h}{2}.$$

The subdomain corresponding to the i-th lamina is thus

$$\mathcal{B}_i \coloneqq \{\mathcal{F}(\xi_1, \xi_2) + \zeta\, \mathbf{a}_3(\xi_1, \xi_2) : \ (\xi_1, \xi_2) \in \Xi,\ \zeta \in (\zeta_{i-1}, \zeta_i)\}, \quad i = 1, \ldots, n,$$

and we additionally denote the position of its own midsurface as $z_i \coloneqq \frac{\zeta_{i-1}+\zeta_i}{2}$, and its thickness as $h_i = \zeta_i - \zeta_{i-1}$. Such a configuration is illustrated in Figure 4.1.

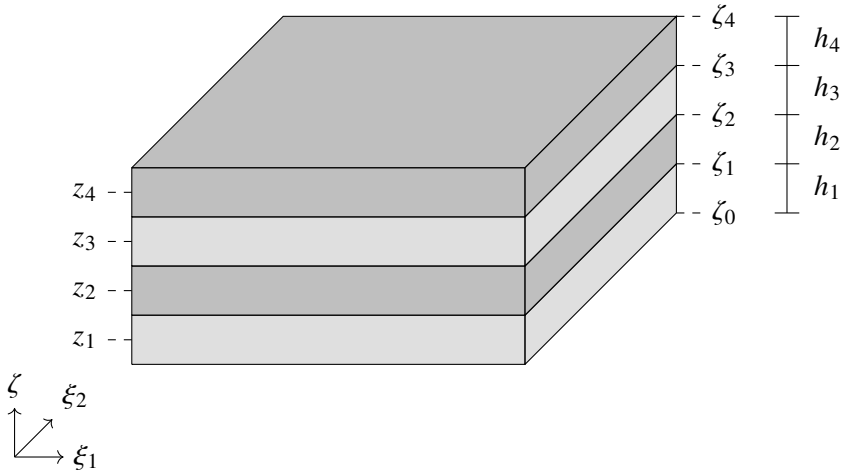


Figure 4.1: Sketch of a laminated composite structure.

If we are to discretize the equations of linear elasticity directly on $\mathcal{B}$, we need to account for the fact that the different material properties of each lamina, together with the last interface constraint from (3.9), imply that the solution $\mathbf{u}$ will not have C^1 regularity at material interfaces, but only C^0. Given the nature of the problem, we do not aim to impose the interface constraints with the Nitsche formulation of Section 3.2: the large ratio of interface area to solid volume makes it impractical, in the sense that each lamina will only be minimally resolved through the thickness, yet it has to be coupled to the ones directly above and below, which means that every d.o.f. would be involved in some Nitsche interface-coupling term. For this reason, in this case we would like to incorporate the gradient discontinuities at ply interfaces into the function space, which we will achieve by nesting PU spaces through-the-thickness within PU spaces over the shell's midsurface. The precise explanation follows.

Let $\{\hat{\varphi}_i\}_{i\in\mathcal{I}}$ be a PU over Ξ, and $\{\hat{\psi}_j\}_{j=0}^n$ the set of piecewise-linear hat functions defined on $(-h/2,\ h/2)$ such that $\hat{\psi}_j(\zeta_k) = \delta_{jk}$ (which also form a PU). We may trivially extend both PUs to the parameter space of $\mathcal{B}$ as

$$\varphi_i(\xi_1,\xi_2,\zeta) \coloneqq \hat{\varphi}_i(\xi_1,\xi_2), \quad \psi_j(\xi_1,\xi_2,\zeta) \coloneqq \hat{\psi}_j(\zeta), \quad (\xi_1,\xi_2)\in\Xi, \quad \zeta\in\left(-\frac{h}{2},\ \frac{h}{2}\right).$$

For local spaces $V_{(i,j)}$ defined on $\operatorname{supp}(\varphi_i)\cap\operatorname{supp}(\psi_j)$ for $i\in\mathcal{I}$, $j\in\{0,\dots,n\}$, we define the PU space

$$V^{\mathrm{PU}} = \sum_{i\in\mathcal{I}}\varphi_i\sum_{j=0}^{n}\psi_j V_{(i,j)} = \sum_{i\in\mathcal{I}}\sum_{j=0}^{n}(\varphi_i\psi_j)V_{(i,j)}. \tag{4.2}$$

This PUM space can be understood in two ways, given precisely by the two expressions above:

1. by seeing $\{\varphi_i\}_i$ as the PU and $W_i = \sum_{j=0}^n \psi_j V_{(i,j)}$ as the local spaces, or
2. by seeing $\{\varphi_i\psi_j\}_{i,j}$ as the PU, and $V_{(i,j)}$ as the local spaces.

Strictly speaking, we could also see $\{\psi_j\}_j$ as the PU, but that PU is fixed by the laminate structure, and is therefore not subject to refinement.

With this construction, the PU $\{\psi_j\}_{j=0}^n$ contributes the necessary gradient discontinuities at the ply interfaces. Each PU function $\varphi_i\psi_j$ overlaps plies j and $j+1$ (provided they exist), and has kinks at ζ_{j-1}, ζ_j and ζ_{j+1} (also provided they exist). We support this explanation with a graphical sketch in Figure 4.2.

Let us further note that the PU functions $\{\varphi_i\psi_j\}_{i,j}$ have no flat-top regions, given the nature of the $\{\hat{\psi}_j\}_j$. Nevertheless, we may consider the flat-top regions of $\{\varphi_i\}_{i\in\mathcal{I}}$, and define the *inherited* FT-region of $\varphi_i\psi_j$ as the intersection of its support with the FT-region of φ_i. An equivalent definition is the subdomain of its support overlapped at most by one other PU function (which must be either $\varphi_i\psi_{j+1}$ or $\varphi_i\psi_{j-1}$, shall they exist).

Remark 4.1. The cover construction of the PUM, at least the one implemented in the PUMA library, allows us only to generate hat functions for equidistant nodes $\{\zeta_k\}_{k=0}^n$, i.e. for the case where all laminae have the same thickness. To handle the general case, we may also parameterize the

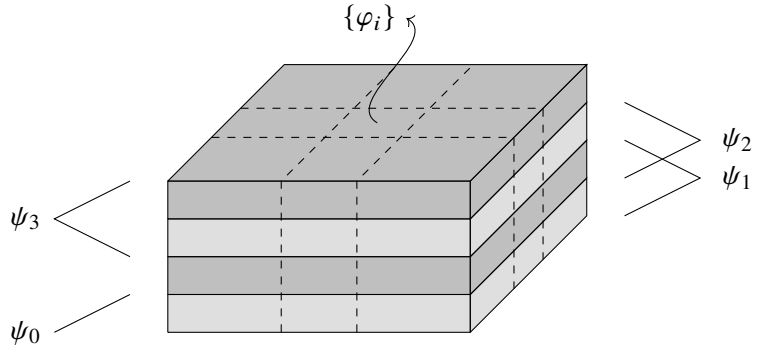


Figure 4.2: Sketch of the PU construction for a laminated composite structure.

volume through the thickness via a continuous piecewise-linear map $\mathcal{T}\colon [-1,1] \to \left(-\frac{h}{2}, \frac{h}{2}\right)$, with derivative discontinuities at the equidistant nodes

$$\tau_k = \frac{2k}{n} - 1 \in [-1,1], \quad k = 0, \dots, n,$$

at which $\mathcal{T}(\tau_k) = \zeta_k$.

4.3 Classical Laminate Theory

When applying the Kirchhoff-Love ansatz to a laminated composite as the one described in the previous section, the resulting reduced model has to account for the different materials and their corresponding locations and thicknesses, and is therefore not as simple as the one discussed in Section 3.3 for a single material. To begin with, every integral over $(-h/2, h/2)$ has to be split over laminae, yielding factors

$$\int_{\zeta_{i-1}}^{\zeta_i} \mathrm{d}\zeta = h_i, \quad \int_{\zeta_{i-1}}^{\zeta_i} \zeta \,\mathrm{d}\zeta = \frac{\zeta_i^2 - \zeta_{i-1}^2}{2} = z_i h_i, \quad \int_{\zeta_{i-1}}^{\zeta_i} \zeta^2 \,\mathrm{d}\zeta = \frac{\zeta_i^3 - \zeta_{i-1}^3}{3} = \frac{h_i^3}{12} + z_i^2 h_i.$$

Note that, in this case, terms proportional to ζ do not directly vanish when integrating over the thickness. The Kirchhoff-Love ansatz (3.11), when introduced in the load-free energy functional of linear elasticity, leads again to

$$\frac{1}{2}\int_{\mathcal{B}} \boldsymbol{\sigma}(\mathbf{U})\colon \boldsymbol{\varepsilon}(\mathbf{U}) \,\mathrm{d}\mathcal{B} \approx \frac{1}{2}\int_{\Omega} A(\mathbf{u})\colon \alpha(\mathbf{u}) \,\mathrm{d}\Omega + \frac{1}{2}\int_{\Omega} B(\mathbf{u})\colon \beta(\mathbf{u}) \,\mathrm{d}\Omega,$$

where α and β denote the membrane and bending strains from (3.12), but the membrane and bending stresses are defined in this case as

$$\begin{aligned} A(\mathbf{u}) &\coloneqq \sum_{i=1}^{n} h_i \mathbb{C}_i\colon \alpha(\mathbf{u}) + \sum_{i=1}^{n} z_i h_i \mathbb{C}_i\colon \beta(\mathbf{u}), \\ B(\mathbf{u}) &\coloneqq \sum_{i=1}^{n} z_i h_i \mathbb{C}_i\colon \alpha(\mathbf{u}) + \sum_{i=1}^{n} \left(\frac{h_i^3}{12} + z_i^2 h_i\right) \mathbb{C}_i\colon \beta(\mathbf{u}), \end{aligned} \tag{4.3}$$

with $\mathbb{C}_i$ being the stiffness tensor for each material. To describe it, let us restrict our analysis to the transversely-isotropic model (recall (4.1)) and operate on the coordinate system given by the main orthotropicity direction (in the shell's tangent plane, and denoted by index 0), the orthogonal direction to it in the tangent plane (denoted by 1), and the surface normal direction (denoted by 2). Note that this coordinate system is in general location-dependent, in particular for non-planar shells. We may then write the action of the stiffness tensor onto a symmetric strain tensor, $\mathbb{W} = \mathbb{C} : \mathbb{S}$, as

$$\begin{pmatrix} \mathbb{W}_{00} & \mathbb{W}_{01} \\ \mathbb{W}_{10} & \mathbb{W}_{11} \end{pmatrix} = \begin{pmatrix} \dfrac{E_0}{1-\nu_{01}\nu_{10}}\mathbb{S}_{00} + \nu_{10}\mathbb{S}_{11} & 2G_{01}\mathbb{S}_{01} \\ 2G_{01}\mathbb{S}_{01} & \dfrac{E_1}{1-\nu_{01}\nu_{10}}\mathbb{S}_{11} + \nu_{01}\mathbb{S}_{00} \end{pmatrix},$$

which is the generalization of (3.13) and thus incorporates a plane-stress assumption. Within a composite material, each $\mathbb{C}_i$ may have its own parameters $E_0, E_1, \nu_{01}, \nu_{10}, G_{01}$, as well as its own orientation in the shell's tangent plane (leading to a different meaning of the 0 and 1 directions in physical space).

Finally, let us mention that the aggregated stresses (4.3) are usually written as

$$A(\mathbf{u}) = \mathbb{A} : \alpha(\mathbf{u}) + \mathbb{B} : \beta(\mathbf{u}), \quad B(\mathbf{u}) = \mathbb{B} : \alpha(\mathbf{u}) + \mathbb{D} : \beta(\mathbf{u})$$

with the definitions $\mathbb{A} \coloneqq \sum_{i=1}^n h_i \mathbb{C}_i$, $\mathbb{B} \coloneqq \sum_{i=1}^n z_i h_i \mathbb{C}_i$ and $\mathbb{D} \coloneqq \sum_{i=1}^n \left(\frac{h_i^3}{12} + z_i^2 h_i\right) \mathbb{C}_i$, which is usually referred to as the $\mathbb{ABD}$-matrix of the composite laminate. We refer to [5] for more details.

Remark 4.2. For a so-called *symmetric stacking sequence*, meaning a laminate whose material distribution is symmetric with respect to the thickness dimension (around $\zeta = 0$), the $\mathbb{B}$ matrix is null, and thus there is no coupling between membrane and bending modes. Indeed, for every lamina occupying an interval (ζ_{k-1}, ζ_k), it either occupies the middle position, and thus $z_i = 0$, or there is another lamina from the same material and occupying the interval $(\zeta_{i-1}, \zeta_i) = (-\zeta_k, -\zeta_{k-1})$, in which case $z_i h_i = -z_k h_k$ and their contributions to $\mathbb{B}$ cancel out.

Remark 4.3. For blending solid and shell models of laminated composites, we can rely on the developments of Section 3.4, simply considering the new definitions of the membrane and bending stresses when constructing the boundary moment $\mathcal{Q}(\mathbf{u}; \mathbf{x})$. The resulting $\mathcal{Q}$ is then a sort of "homogenized" version that will be used through the whole thickness, incorporating contributions from every material at each $\zeta \in (-h/2, h/2)$. This is not necessarily a problem, since, after all, the shell model itself corresponds to a homogenization through the thickness.

5 Efficient iterative solvers

The new applications handled with the PUM in this thesis, in particular

- the higher-order Kirchhoff-Love shell problems, and
- the anisotropic discretizations of three-dimensional composite shell structures, with PU spaces in the shape of (4.2),

revealed that the already available preconditioners in the PUMA library (merely the Jacobi and the Gauss-Seidel iteration) were not particularly efficient or even effective for the iterative solution of the resulting linear systems, and not even when used as smoothers of a multilevel preconditioner. The need for an effective and efficient preconditioner was eventually fulfilled by the Factorized Sparse Approximate Inverse (FSAI) preconditioner [10, 12].

We dedicate this chapter to briefly summarize the work we presented in [15]: the adaptive and nested block-FSAI preconditioners, based on the block structure of matrices arising from the PUM, and the Chebyshev iteration of the fourth kind, which may be used for smoothing instead of the usual Richardson iteration. Our work [15] is attached in Appendix C.2.

5.1 The FSAI preconditioner

Let us succinctly introduce the idea behind the FSAI preconditioner. For an invertible matrix $A \in \mathbb{R}^{N\times N}$ and a predefined sparsity pattern $\mathcal{P} \subset \{1,\dots,N\}^2$, a *sparse approximate inverse* (SAI) of A based on $\mathcal{P}$ is some matrix $M \in \mathbb{R}^{N\times N}$ with sparsity pattern $\mathcal{P}(M) \coloneqq \{(i,j) : M_{ij} \neq 0\} \subset \mathcal{P}$ which approximates A^{-1}, usually by solving

$$M \coloneqq \operatorname*{arg\,min}_{\mathcal{P}(\tilde{M})\subset\mathcal{P}} \left\| I - \tilde{M}A \right\|_F ,$$

with $\|\cdot\|_F$ the Frobenius norm, defined by $\|B\|_F^2 = \operatorname{tr}(B^\top B)$.

With this simple process, however, the preconditioning matrix M will in general not be symmetric positive definite (s.p.d.), even if the original matrix A is. The Factorized Sparse Approximate Inverse (FSAI) preconditioner overcomes this issue by constructing the preconditioning matrix as $M = F^\top S^{-1} F$, with F a lower triangular matrix with unit diagonal, and $S = \operatorname{diag}(FAF^\top)$. In the block-structured case, F is lower triangular with respect to such block structure, its diagonal blocks are identity matrices, and S is the block diagonal component of $FAF^\top$.

For a given lower triangular sparsity pattern in the block structure $\mathcal{P}$ (containing the diagonal), the resulting matrix F can be defined in a row-wise manner as

$$F[\{r\}, \tilde{\mathcal{P}}_r] = -A[\{r\}, \tilde{\mathcal{P}}_r]\, A[\tilde{\mathcal{P}}_r, \tilde{\mathcal{P}}_r]^{-1}, \qquad r \in \{0,\dots,n\}, \tag{5.1}$$

where $\tilde{\mathcal{P}}_r \coloneqq \{c : (r,c) \in \mathcal{P}\} \setminus \{r\}$ is the corresponding row in the sparsity pattern, without the r index itself. By $A[\mathcal{I}, \mathcal{J}]$ we denote the restriction of the (block-)matrix A to row indices in $\mathcal{I}$ and column indices in $\mathcal{J}$.

The preconditioned matrix can thus be written as $GAG^\top$ with $G = S^{-1/2}F$. A fundamental property of the FSAI preconditioner is that the Kaporin number of the preconditioned matrix $\beta(GAG^\top)$ is a minimizer of $\beta(\tilde{G}A\tilde{G}^\top)$ over all full-rank matrices $\tilde{G}$ with sparsity pattern $\mathcal{P}$. The so-called Kaporin number of an s.p.d. matrix is the ratio between the algebraic and the geometric means of its eigenvalues, or in other words,

$$\beta(B) \coloneqq \frac{\operatorname{tr}(B)}{N \det(B)^{1/N}}, \quad B \in \mathbb{R}^{N\times N}.$$

One may also rely on the Kaporin number to adaptively construct a sparsity pattern for F, as initially done by Janna et al. [11]. In [15, Section 2.1] we presented our own adaptive algorithm for the generation of the sparsity pattern, conceived for block-structured matrices as those naturally arising from PUM discretizations (2.4). We provide a simplified version of it in Algorithm 5.1.

Algorithm 5.1 Adaptive construction of the block-FSAI preconditioner

Input: A s.p.d. matrix A with a block structure, and an initial lower triangular sparsity pattern $\mathcal{P}_0 \subset \{1,\dots,n\}^2$ on said block structure, containing all diagonal pairs of indices. The number of adaptive steps $t_{\max} \geq 1$ and a threshold parameter $\tau \in (0,1]$.

Initialize $\mathcal{P} = \mathcal{P}_0$.

for $t = 1,\dots,t_{\max}$ **do**
  Compute the FSAI matrix F based on $\mathcal{P}$, according to (5.1), and $H \coloneqq FA$.
  for $k \in \{1,\dots,n\}$ **do**
    Let $C \coloneqq \{c \in \{0,\dots,k-1\} : H_{kc} \not\equiv 0\}$ be the set of admissible column indices.
    if $C = \emptyset$ **then** continue
    Let $W_c \coloneqq A_{cc} - A[\{c\}, \tilde{\mathcal{P}}_k]\, A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1}\, A[\tilde{\mathcal{P}}_k, \{c\}]$ for $c \in C$, and find

$$c^\star \in \operatorname*{arg\,min}_{c\in C} \rho_c, \quad \rho_c \coloneqq \frac{\det(W_c - H_{kc}^\top H_{kk}^{-1} H_{kc})}{\det(W_c)}.$$

    if $\rho_{c^\star} < \tau$ **then** $\mathcal{P} \leftarrow \mathcal{P} \cup \{(k, c^\star)\}$
Compute the FSAI matrix F based on the current state of $\mathcal{P}$, according to (5.1).
return F

Our derivation is similar to the one described by Sedlacek in [21, Appendix C], but we arrive at a different adaptive algorithm, based on exact reduction factors of Kaporin numbers. In our work, following the available literature [13], we additionally explore the concatenation of FSAI preconditioners in a nested manner.

We have found that our adaptive FSAI algorithm properly adapts to the anisotropy introduced by PUM spaces like (4.2) used for the discretization of composite shell structures, as will be indirectly shown in several examples from Chapter 6. Additionally, increasing the density of the FSAI preconditioners (either by increasing the number of adaptive steps $t_{\max}$, or by nesting FSAI preconditioners) yields efficient smoothers for higher-order problems when using a multilevel preconditioner, such as those originally introduced for the PUM in [6] (see [15, Section 5]).

5.2 Chebyshev smoothing

In order to increase smoothing efficiency within a multilevel preconditioner, instead of the usual Richardson iteration, we may rely on the Chebyshev iteration for concatenating smoothing steps, and particularly on the Chebyshev iteration of the fourth kind introduced by Lottes [16].

As a polynomial iteration, the Chebyshev iteration of degree k acts on the error $e_k = x - x_k$ as

$$e_k = p_k(A)e_0, \quad k \geq 1,$$

where the polynomial p_k must fulfill $p_k(0) = 1$. Chebyshev polynomials (of the first kind) arise from the minimization of the maximum of $|p_k(\lambda)|$ over an interval containing the spectrum of A, i.e. for $\sigma(A) \subset [\alpha, \beta]$ with $\beta > \alpha > 0$, they solve

$$\min_{\substack{p_k \in \mathbb{P}_k \\ p_k(0)=1}} \max_{\lambda \in [\alpha, \beta]} |p_k(\lambda)|.$$

The fact that α cannot be chosen as zero enforces the need for an estimate of the lowest eigenvalue of A or, for smoothing purposes, an estimate of the lower end of the fine spectrum. However, by including a weight in the minimization as

$$\min_{\substack{p_k \in \mathbb{P}_k \\ p_k(0)=1}} \max_{\lambda \in [\alpha, \beta]} \lambda^{1/2} |p_k(\lambda)|,$$

which concedes prevalence to the higher end of the spectrum, the solution is given by Chebyshev polynomials of the fourth kind, and a limit argument allows the choice $\alpha = 0$. This means that only an estimate of the largest eigenvalue of A is needed. When performing a preconditioned Chebyshev iteration (which we do with FSAI), we need an estimate of $\lambda_{\max}(MA)$, where M is the preconditioning matrix.

In [15, Section 3.1], we proposed a simpler version of the Chebyshev iteration of the fourth kind than the one from [16], which we derived following the work of Gutknecht and Röllin [9] for the original Chebyshev iteration. Given its simplicity, for completeness we include it again here, in Algorithm 5.2.

Algorithm 5.2 Chebyshev iteration of the fourth kind

Input: A linear system $Ax = b$ with a s.p.d. matrix A, and an equally s.p.d. preconditioner M. An initial tentative solution x_0. An upper bound on the spectrum of MA, $\beta \geq \lambda_{\max}(MA)$. A positive integer $k \geq 1$.

1: Initialize $r = b - Ax_0$, $p = Mr$.
2: Initialize $\mu = 1/3$.
3: Set $d = 4\beta^{-1}$.

4: Initialize $x = x_0 + d\mu\, p$

5: **for** $i = 2, \ldots, k$ **do**
6: $\quad r \leftarrow b - Ax$
7: $\quad p \leftarrow \mu^2\, p + Mr$
8: $\quad \mu \leftarrow (2 - \mu)^{-1}$
9: $\quad x \leftarrow x + d\mu\, p$

10: **return** x

6 Numerical experiments

In this section, we will gather some numerical experiments to complement those of [14] (for Nitsche's method) and [15] (for Chebyshev-smoothing with FSAI preconditioners). We will focus our attention on using the multilevel solver for 3-dimensional solid models of composite materials, to showcase how the adaptive FSAI preconditioner yields an efficient smoother, on the combination of Nitsche's method with multilevel preconditioners (following the discussion from Section 3.6), and on solid-shell blending examples with both simple and composite materials.

Example 6.1. We begin by showing the effectiveness of adaptive FSAI preconditioners as smoothers for a composite plate, discretized following the approach described in Section 4.2. In particular, we consider the rectangular domain

$$\Omega = \left[(-3,3)\times(-0.75,0.75)\times\left(-\frac{h}{2},\frac{h}{2}\right)\right] \setminus \left\{(x,y,z):\ x^2+y^2 \leq 0.125^2\right\},$$

i.e. a rectangular plate of length 6 m, width 1.5 m and thickness h, with a hole of radius 0.125 m at its center. Its thickness will be determined by the number of laminae N_ℓ, which we will consider of fixed thickness 0.006 m, and thus $h = 0.006N_\ell$ m.

Each lamina will hold the same kind of transversely-isotropic material, with a varying orientation in the xy-plane. Angles will be given in degrees and understood clock-wise, with 0 corresponding to the x-axis. A very simple stacking sequence is illustrated in Figure 6.1. The subscript s is used to denote a symmetric stacking sequence, which is to be understood as appending the provided angles in the reverse order, e.g. $[0, 45, 90]_s$ is simply shorthand notation for $[0, 45, 90, 90, 45, 0]$.

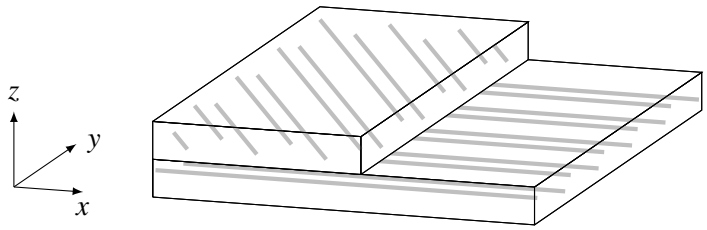


Figure 6.1: Stacking sequence [0, 45].

As material properties, we will use Young's moduli $E_0 = 23.8$ MPa and $E_1 = 1.3$ MPa, Poisson's ratios $\nu_{01} = 0.32$ and $\nu_{12} = 0.45$, and shear modulus $G_{01} = 0.73$ MPa. We will consider different numbers of layers N_ℓ, and the stacking sequences will be generated from [45, 0, -45, 90] as follows:

N_ℓ	Stacking sequence
6	$[0, -45, 90]_s$
8	$[45, 0, -45, 90]_s$
12	$[-45, 90, 45, 0, -45, 90]_s$
16	$[45, 0, -45, 90, 45, 0, -45, 90]_s$

As boundary conditions, we consider the left side of the plate fixed, displace the right side by $(0.04\,\text{m}, 0\,\text{m}, 0\,\text{m})$, and leave all remaining boundaries traction-free (i.e. homogeneous Neumann conditions). We discretize the plate in the xy-plane with increasingly smaller patches close to the hole. We begin with a coarsest level made of patches with unstretched side length 0.375 m (which we then stretch by 1.3), and for each new level we split patches intersecting $\{(x, y, z) : x^2 + y^2 \leq 0.25^2\}$ into 4, propagating this refinement to contiguous patches if necessary, so that the refinement level difference between neighboring patches is at most 1 (see Figure 6.2). We perform 5 such refinements to generate our multilevel sequence, and since the patches at the left and right boundaries undergo no refinement at any step, the stabilization functions are the same at all levels. No patch aggregation is performed, which would only affect patches intersecting the hole, at which no Dirichlet boundary condition is enforced.

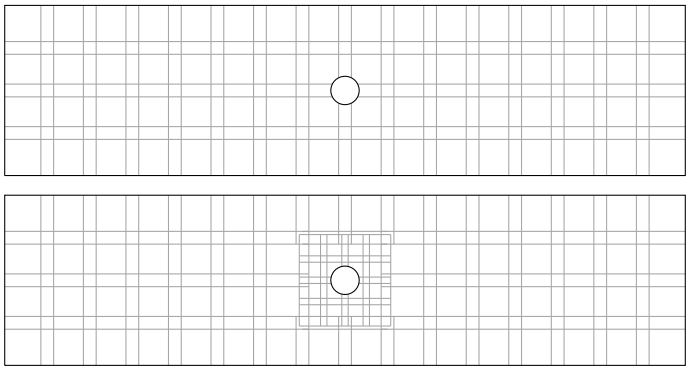


Figure 6.2: The two coarsest covers in the xy-plane for the plate domain of Example 6.1.

For each local space we use bilinear polynomials in the xy variables, i.e. $\{1, x, y, xy\}$. For the multilevel preconditioner, we construct prolongations via the local-to-local transfer from [6, Section 5.1], and rely on the assembled stiffness matrices at each level. For the FSAI smoothers, we set the number of adaptive steps equal to N_ℓ (which we denote N_ℓ-adaptive FSAI). Finally, we embed the multilevel preconditioner based on a $V(3, 3)$-cycle into a CG solver, with the stopping criterion $\|r_n\|_2 \leq 10^{-15} \|r_0\|_2$, and as initial solution for each new level we consider the prolongated solution from the coarser level. The results for each choice of N_ℓ are shown in Table 6.1 (for Chebyshev smoothing of the fourth kind) and Table 6.2 (for Richardson smoothing). As it can be seen, the impact of Chebyshev smoothing is quite limited, but the adaptive FSAI preconditioners keep the number of iterations mostly level-independent.

To reveal where Chebyshev smoothing actually has a significant impact, we turn to using 8-adaptive FSAI preconditioners for the cases $N_\ell = 12, 16$. This limits the effectiveness of the preconditioners themselves, and that is where the effect of Chebyshev smoothing can be seen, as revealed by the results that we gather in Table 6.3 and Table 6.4. It not only improves convergence, but it makes it possible where Richardson smoothing simply does not.

Level	N_ℓ			
	6	8	12	16
2	16	15	15	14
3	24	21	19	18
4	25	21	19	17
5	27	23	20	17
6	27	22	20	17

Table 6.1: CG iterations when using Chebyshev smoothing with N_ℓ-adaptive FSAI.

Level	N_ℓ			
	6	8	12	16
2	21	18	17	15
3	44	27	22	19
4	29	23	19	17
5	32	25	20	18
6	30	23	19	17

Table 6.2: CG iterations when using Richardson smoothing with N_ℓ-adaptive FSAI.

Level	$N_\ell = 12$	$N_\ell = 16$
2	26	25
3	33	30
4	24	22
5	25	22
6	24	22

Table 6.3: CG iterations when using Chebyshev smoothing with 8-adaptive FSAI.

Level	$N_\ell = 12$	$N_\ell = 16$
2	70	71
3	178	311
4	280	775
5	489	>1000
6	>1000	>1000

Table 6.4: CG iterations when using Richardson smoothing with 8-adaptive FSAI.

We can conceive one further configuration in-between Chebyshev and Richardson smoothing: damped Richardson smoothing, with a damping factor of

$$\omega = \frac{4}{3}\frac{1}{\lambda_{\max}(MA)},$$

where MA is the (FSAI-)preconditioned matrix. With this choice, one step of the damped Richardson iteration is equivalent to one step of the Chebyshev iteration of the fourth kind. The corresponding results are shown in Table 6.5, revealing that it is actually the damping based on $\lambda_{\max}(MA)$ that helps keep the iteration numbers under control for increasing levels, while concatenating multiple smoothing steps with the Chebyshev iteration (here we consider only 3) simply allows to further reduce the number of iterations.

Level	$N_\ell = 12$	$N_\ell = 16$
2	39	38
3	54	50
4	37	34
5	36	33
6	31	28

Table 6.5: CG iterations when using damped Richardson smoothing with 8-adaptive FSAI.

Example 6.2. For our second example, we simply transform the plate of the previous one into a cylindrical shell. For an axis parallel to the x-axis and a curvature radius R such that $\pi R \geq 1.5$ m, we can easily construct a NURBS (actually a Bézier) parametrization of a cylindrical surface of length 6 m and (curved) width 1.5 m, as in the previous example. Indeed, such a surface has u-degree 1, v-degree 2, and after defining

$$\alpha = \frac{0.75\,\mathrm{m}}{R} \leq \frac{\pi}{2}, \quad q = \frac{\sin(\alpha)}{2\sin(\alpha/2)}, \quad \phi = \arccos(q),$$
$$\hat{x} = 3\,\mathrm{m}, \quad \hat{y} = R\sin(\alpha), \quad \hat{z} = \hat{y}\tan(\phi),$$

its control points can be written as

$$\mathbf{P}_{0,0} = (-\hat{x}, -\hat{y}, 0), \quad \mathbf{P}_{0,1} = (-\hat{x}, 0, \hat{z}), \quad \mathbf{P}_{0,2} = (-\hat{x}, \hat{y}, 0),$$
$$\mathbf{P}_{1,0} = (\hat{x}, -\hat{y}, 0), \quad \mathbf{P}_{1,1} = (\hat{x}, 0, \hat{z}), \quad \mathbf{P}_{1,2} = (\hat{x}, \hat{y}, 0),$$

and the corresponding weights as

$$w_{0,0} = 1, \quad w_{0,1} = q, \quad w_{0,2} = 1,$$
$$w_{1,0} = 1, \quad w_{1,1} = q, \quad w_{1,2} = 1.$$

We finally turn the physical domain of our previous example into the parameter domain for this one, i.e.

$$\Xi = \left[(-3,3) \times (-0.75, 0.75) \times \left(-\frac{h}{2}, \frac{h}{2}\right)\right] \setminus \left\{(x,y,z) : \; x^2 + y^2 \leq 0.125^2\right\}.$$

To construct the multilevel preconditioner, we proceed as in the previous example. We provide the number of CG iterations in Table 6.6 and Table 6.7, respectively with either Chebyshev or Richardson smoothing. In this case, we directly rely on FSAI preconditioners constructed with $\min(N_\ell, 8)$ adaptive steps, which as before reveals the impact of Chebyshev smoothing for $N_\ell = 12, 16$.

Level	N_ℓ			
	6	8	12	16
2	16	15	15	14
3	24	21	19	18
4	25	21	19	17
5	27	23	20	17
6	27	22	20	17

Table 6.6: CG iterations when using Chebyshev smoothing with $\min(N_\ell, 8)$-adaptive FSAI.

Level	N_ℓ			
	6	8	12	16
2	21	17	68	71
3	34	27	198	391
4	29	24	226	469
5	32	26	305	>1000
6	31	25	535	>1000

Table 6.7: CG iterations when using Richardson smoothing with $\min(N_\ell, 8)$-adaptive FSAI.

Example 6.3. Let us now focus our attention on the combination of Nitsche's method with multilevel preconditioners, which we already discussed in Section 3.6. For that matter, we pick the biharmonic equation in the unit square domain $\Omega = (0,1)^2$ with essential boundary conditions imposed over the whole boundary, i.e.

$$\begin{cases} \Delta^2 u = f & \text{in } \Omega, \\ u = g_0 & \text{on } \partial\Omega, \\ \nabla u \cdot \mathbf{n} = g_1 & \text{on } \partial\Omega. \end{cases}$$

For a discrete space $V_n \subset H^2(\Omega)$, the weak formulation following Nitsche's method corresponds to finding $u \in V_n$ such that

$$a_n(v,u) = \ell_n(v), \quad \forall v \in V_n,$$

with bilinear and linear forms

$$\begin{aligned} a_n(v,u) = &\int_\Omega \Delta u \Delta v \,\mathrm{d}\Omega + \int_{\partial\Omega} \left(\gamma_n^{(0)} uv + v\nabla(\Delta u)\cdot\mathbf{n} + u\nabla(\Delta v)\cdot\mathbf{n} \right) \mathrm{d}\Gamma \\ &+ \int_{\partial\Omega} \left(\gamma_n^{(1)} (\nabla u\cdot\mathbf{n})(\nabla v\cdot\mathbf{n}) - \Delta u(\nabla v\cdot\mathbf{n}) - \Delta v(\nabla u\cdot\mathbf{n}) \right) \mathrm{d}\Gamma, \\ \ell_n(v) = &\int_\Omega fv \,\mathrm{d}\Omega + \int_{\partial\Omega} g_0 \left(\gamma_n^{(0)} v + \nabla(\Delta v)\cdot\mathbf{n} \right) \mathrm{d}\Gamma + \int_{\partial\Omega} g_1 \left(\gamma_n^{(1)} (\nabla v\cdot\mathbf{n}) - \Delta v \right) \mathrm{d}\Gamma. \end{aligned}$$

We will solve the problem for the manufactured solution

$$\hat{u}(x,y) = \cos(2\pi x)\sin(2\pi y),$$

and we will construct the multilevel sequence of function spaces by homogeneous refinement from level 2 (with $2^2 \cdot 2^2 = 16$ patches) up to level 7 (with $2^7 \cdot 2^7 = 16,384$ patches), For the local spaces, we will use local spaces of polynomials of combined degree $p = 3$, i.e.

$$\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3\}.$$

We will explore the following cases:

- The use of coarser operators obtained via Galerkin triple products $A_{l-1} = P_l^\top A_l P_l$ cascading down from the solving level, or the use of coarser operators assembled at each level, with its corresponding stabilization function(s). These are the two approaches discussed in Section 3.6 for the Poisson equation.
- The use of a single stabilization function $\hat{\gamma}_n$, in which case it has to satisfy the trace inequality

$$\exists \rho \in (0,1) \quad : \quad \int_{\partial\Omega} \hat{\gamma}_n^{-1} \left((\nabla(\Delta v)\cdot\mathbf{n})^2 + (\Delta v)^2 \right) \mathrm{d}\Gamma \leq \rho^2 \int_\Omega (\Delta v)^2 \,\mathrm{d}\Omega, \quad \forall v \in V_n,$$

instead of using the two $\gamma_n^{(0)}$ and $\gamma_n^{(1)}$, with trace inequalities

$$\exists \rho \in (0,1) \quad : \quad \int_{\partial\Omega} \left(\gamma_n^{(0)}\right)^{-1} (\nabla(\Delta v)\cdot \mathbf{n})^2 \,\mathrm{d}\Gamma \leq \frac{\rho^2}{2} \int_\Omega (\Delta v)^2 \,\mathrm{d}\Omega, \quad \forall v \in V_n,$$
$$\exists \rho \in (0,1) \quad : \quad \int_{\partial\Omega} \left(\gamma_n^{(1)}\right)^{-1} (\Delta v)^2 \,\mathrm{d}\Gamma \leq \frac{\rho^2}{2} \int_\Omega (\Delta v)^2 \,\mathrm{d}\Omega, \quad \forall v \in V_n.$$

Finally, for the multilevel preconditioners we will use $V(3,3)$-cycles with adaptive FSAI smoothers, constructed with 7 adaptive steps, and the interlevel transfer prolongations will be constructed following a global-to-local L^2-projection approach, as described in [6, Section 5.1]. Smoothing will be performed with the Chebyshev iteration. At each level, the initial solution will be derived by prolongating the final solution of the previous level. In Table 6.8 we show the number of CG iterations needed to fulfill the convergence criterion $\|r\|_2 < 10^{-15}$. As it can be seen, the greatest effect comes from using a single stabilization function, which hinders convergence, and is significantly amplified by the usage of assembled coarse-level operators, which rely on their own stabilization function.

To understand the reasons behind this, it helps to visualize the different values of $\gamma_n^{(0)}$ and $\gamma_n^{(1)}$ in the case of two stabilization functions, compared with the single $\hat{\gamma}_n$. We provide these measurements in Table 6.9, which clearly show that $\gamma_n^{(1)}$ scales linearly with the inverse of the patch radius, while $\gamma_n^{(0)}$ scales cubically with it, differing by several orders of magnitude at all levels (note that the patch radius is divided by 2 when the level is increased by 1). Additionally, as expected also from this difference, the choice of a single stabilization function produces $\hat{\gamma}_n \approx 0.5\gamma_n^{(0)}$.

Level	a_{coarse}		RAP	
	$1\times\gamma$	$2\times\gamma$	$1\times\gamma$	$2\times\gamma$
3	36	17	17	12
4	90	19	26	12
5	208	19	36	12
6	462	18	46	13
7	1041	18	57	15

Table 6.8: Number of multilevel-preconditioned CG iterations needed for the solution of the biharmonic problem from Example 6.3. By a_{coarse} we denote the generation of coarser-level operators by assembling the corresponding form, and by RAP, their generation via Galerkin triple products starting at each solving level. $N\times\gamma$ denotes the use of N stabilization functions.

Level	$\gamma_n^{(0)}$	$\gamma_n^{(1)}$	$\hat{\gamma}_n$
3	$1.08\cdot10^6$	$2.40\cdot10^2$	$5.38\cdot10^5$
4	$8.55\cdot10^6$	$4.80\cdot10^2$	$4.27\cdot10^6$
5	$6.82\cdot10^7$	$9.60\cdot10^2$	$3.41\cdot10^7$
6	$5.45\cdot10^8$	$1.92\cdot10^3$	$2.72\cdot10^8$
7	$4.35\cdot10^9$	$3.84\cdot10^3$	$2.18\cdot10^9$

Table 6.9: $\|\cdot\|_{L^2(\partial\Omega)} / |\partial\Omega|$ for each stabilization function in Example 6.3.

We note, however, that the better RAP results in Table 6.8 come at the cost of denser coarse-level operators, which are the result of using prolongation matrices based on the global-to-local L^2-projection approach, as mentioned above. To visualize this, in Table 6.10 we gather the number of non-zero entries per row at each coarse-level operator when solving at level 7.

Level	a_{coarse}	RAP
2	6.25	12.25
3	7.56	18.06
4	8.27	21.39
5	8.63	23.16
6	8.81	24.07

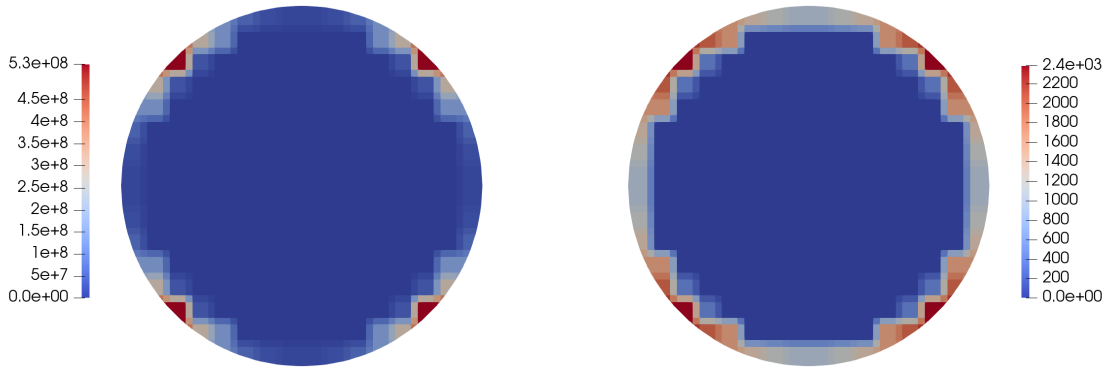
Table 6.10: Density of the coarse-level operators used for solving at level 7 generated either with the a_{coarse} or the RAP approach, measured as the number of non-zero entries in the matrix divided by its number of rows.

Alternatively, we could use a local-to-local L^2-projection approach (see also [6, Section 5.1]), which is less accurate, but yields coarse-level operators with the same sparsity pattern (and thus the same density) as those that are generated through assembly (the a_{coarse} approach). We show the number of CG iterations for this projection approach in Table 6.11. As it can be seen, the use of these interlevel prolongations produces level-independent iteration numbers for the a_{coarse} approach, but only for it, since the lower quality of the coarse-level operators in the RAP approach deteriorates multilevel convergence, leading to iteration numbers that increase with the solving level.

Level	a_{coarse}		RAP	
	$1\times\gamma$	$2\times\gamma$	$1\times\gamma$	$2\times\gamma$
3	24	20	18	15
4	34	22	33	22
5	37	22	55	32
6	38	22	89	51
7	39	21	138	78

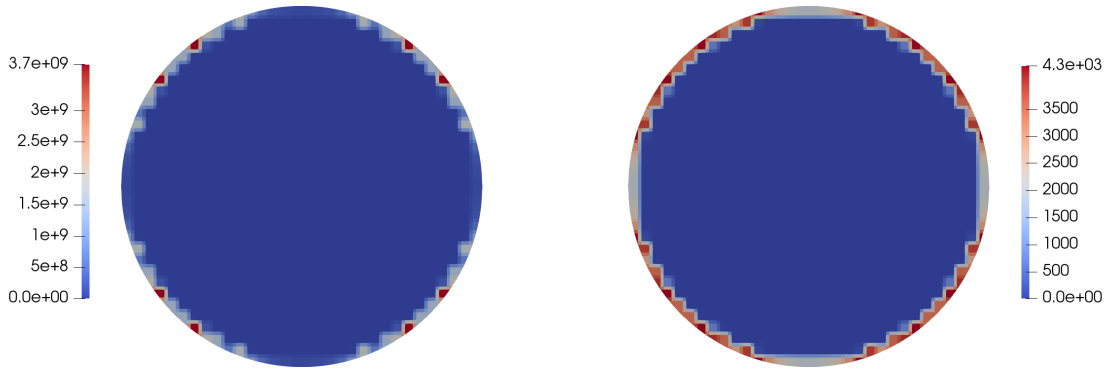
Table 6.11: Number of multilevel-preconditioned CG iterations needed for the solution of the biharmonic problem from Example 6.3, when using the local-to-local L^2-projection approach to generate the interlevel prolongations.

Example 6.4. Let us repeat the previous example, but this time considering the circular domain $\Omega = B((0.5, 0.5), 0.5) \subset (0, 1)^2$. In this case, patch aggregation is necessary to prevent the stabilization functions from reaching arbitrarily large values at certain patches, something that was not necessary for the unit square domain from the previous example. Additionally, the stabilization functions are not homogeneous over the boundary, and their heterogeneity differs between refinement levels. In Figure 6.3 we show $\gamma_n^{(0)}$ and $\gamma_n^{(1)}$ for refinement levels 4 and 5.



(a) $\gamma_n^{(0)}$ for level 4.

(b) $\gamma_n^{(1)}$ for level 4.



(c) $\gamma_n^{(0)}$ for level 5.

(d) $\gamma_n^{(1)}$ for level 5.

Figure 6.3: Stabilization functions for the problem of Example 6.4.

We share the number of CG iterations for each configuration in Table 6.12. Following the results for the previous example, we use the local-to-local L^2-projection to generate prolongation operators in the a_{coarse} case, and the global-to-local one in the RAP case. Clearly, the RAP approach with two distinct stabilization functions remains the most stable one, showing also the smallest variation with respect to its corresponding results for the unit square domain.

Level	a_{coarse}		RAP	
	$1\times\gamma$	$2\times\gamma$	$1\times\gamma$	$2\times\gamma$
3	37	21	13	10
4	58	30	24	13
5	95	29	34	12
6	169	46	61	15
7	1088	57	94	18

Table 6.12: Number of multilevel-preconditioned CG iterations needed for the solution of the biharmonic problem from Example 6.4.

Example 6.5. For our next example, we recover the composite plate with hole from Example 6.1, where now we will restrict the solid model to a region around the hole, modeling everything else as a plate. Remembering that the hole had radius $r = 0.125$ m, we set the solid part to be $\Omega_{\text{solid}} = (-3r, 3r) \times \left(-\frac{h}{2}, \frac{h}{2}\right)$. For this solid part, we use the same refinement strategy as in Example 6.1 and keep the finest level, while for the remaining plate, we keep the coarsest refinement considered there. For the solid-shell coupling terms, we use γ-averaging as in (3.20). Finally, we restrict our attention to the case $N_\ell = 8$. An illustration of the discretization around the solid part is provided in Figure 6.4.

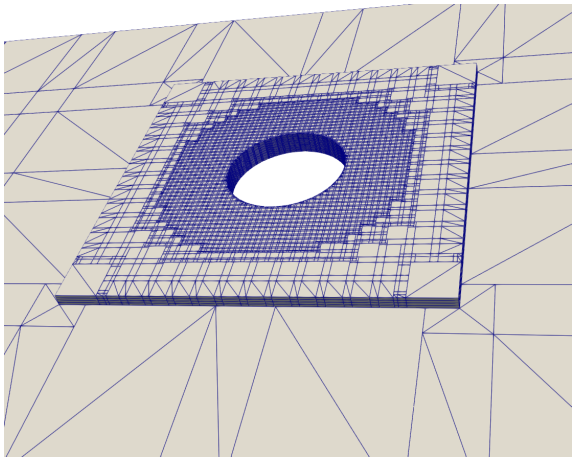
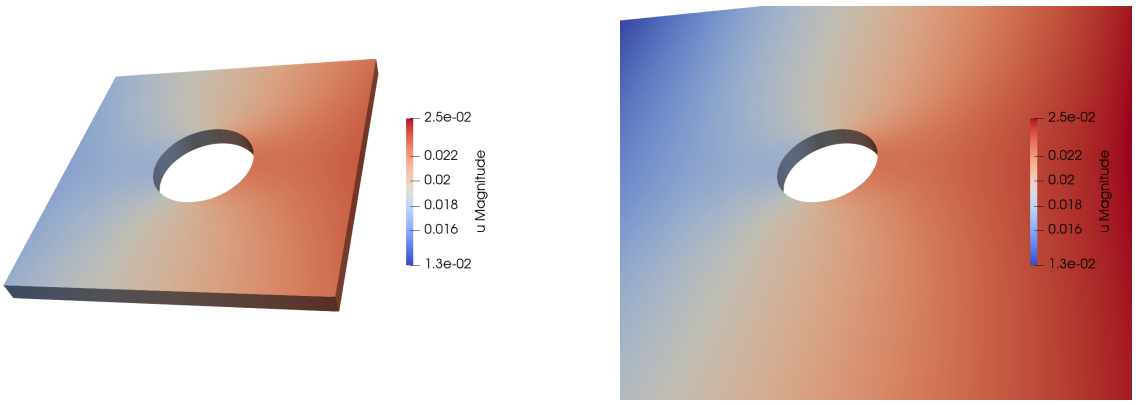


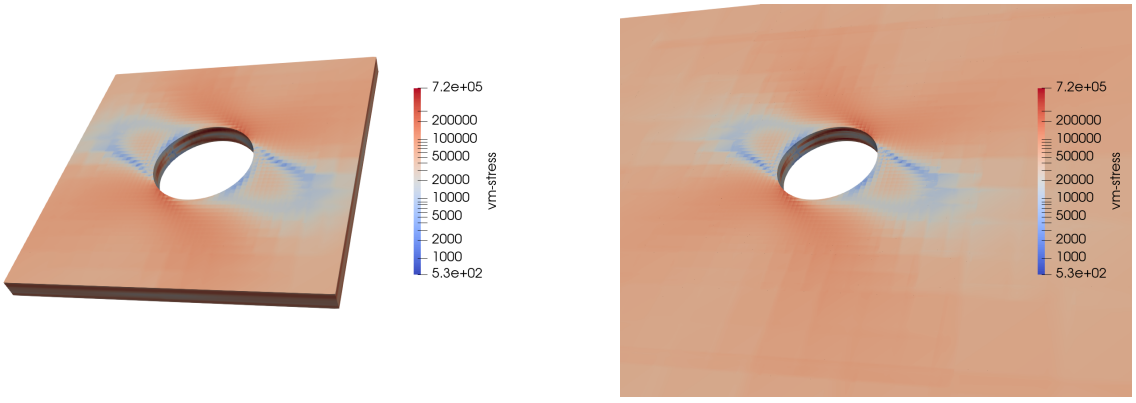
Figure 6.4: Integration cells generated for the problem in Example 6.5, revealing the structure of the covers used for the solid and plate parts.

The displacements obtained with each approach are shown in Figure 6.5, and the Von Mises stress in Figure 6.6. As we can see, the results in both cases are practically the same, at least visually.



(a) **(b)**

Figure 6.5: Displacements (m) obtained for the problem from Example 6.5: (a) restricting the solid model to a region around the hole, and (b) relying on a solid model for the whole domain, as in Example 6.1.



(a) **(b)**

Figure 6.6: Von Mises stress $\sqrt{1.5\,\mathrm{tr}(\boldsymbol{\sigma}^2) - 0.5\,\mathrm{tr}(\boldsymbol{\sigma})^2}$ (Pa) obtained for the problem from Example 6.5: (a) restricting the solid model to a region around the hole, and (b) relying on a solid model for the whole domain, as in Example 6.1.

Example 6.6. For our last example, we consider a non-trivial case of solid-shell blending, this time based on a single isotropic material. The physical structure consists of three parts: a plate (with a hole) a cylinder, and a joint smoothly connecting the two. We illustrate it in Figure 6.7. We consider the shell to have thickness 0.01 m, and to be made of an isotropic material with $E = 1$ MPa and $\nu = 0.3$. As boundary conditions, we clamp the plate on its 4 sides, while we rotate the top part of the cylinder by an angle $-\pi/15$ around the x-axis, and by an angle $\pi/20$ around the y-axis (both going through the center of the plate). Finally, we consider two scenarios: in the first one, we model the joint as a solid body and the remaining two parts as shells, and in the second one we model everything as a shell. The solution for the first case is shown in Figure 6.8. In all cases, we use γ-averaging for the interfacial coupling terms.

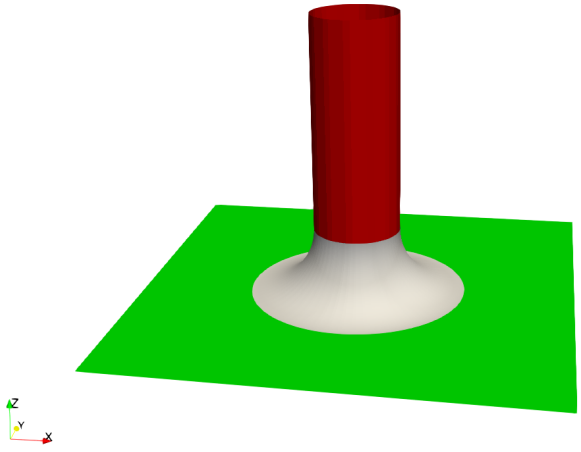


Figure 6.7: The shell structure considered for Example 6.6. The plate has sides of length 1 m and a hole of radius 0.25 m, the cylinder has length 0.5 m and radius 0.1 m, and the radius of the joint's profile is 0.25 - 0.1 = 0.15 m.

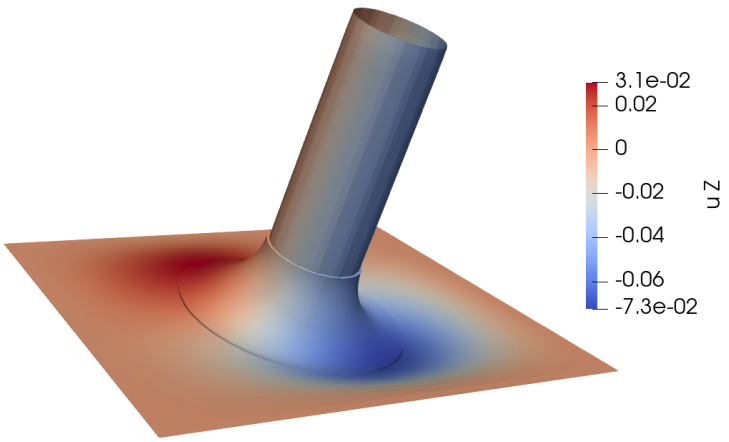


Figure 6.8: Solution for the problem from Example 6.6, with the joint part modeled as a solid, colored by the vertical displacement u_z (m).

For our discretization we refine each parameter space (a unit square $(0, 1)^2$, with a circular hole in the case of the plate) homogeneously, until level 4 for the cylinder, 5 for the plate, and 6 for the joint. In the case of the solid joint, we perform no refinement with respect to the thickness dimension. The resulting discretization for the case of three shells is shown in Figure 6.9. In all cases (for all shells as well as for the solid joint) we use local polynomial spaces of degree 2.

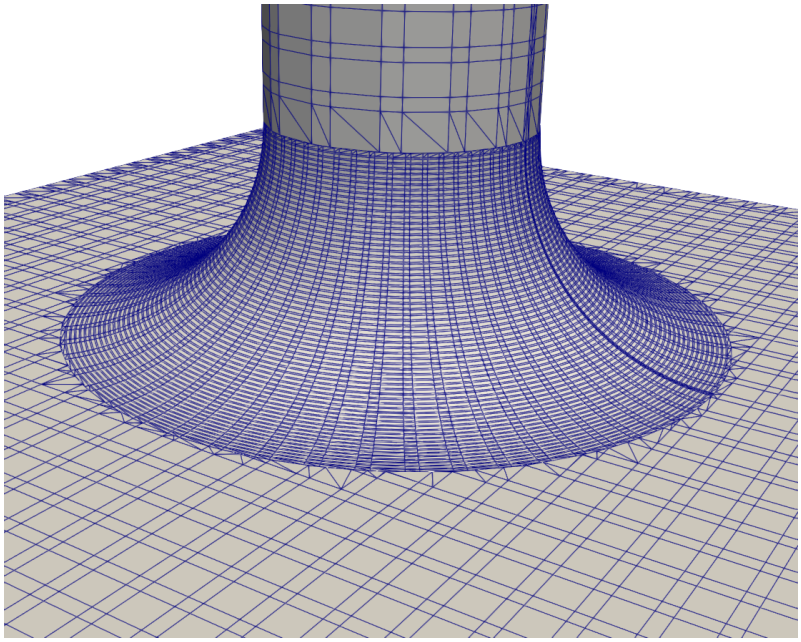
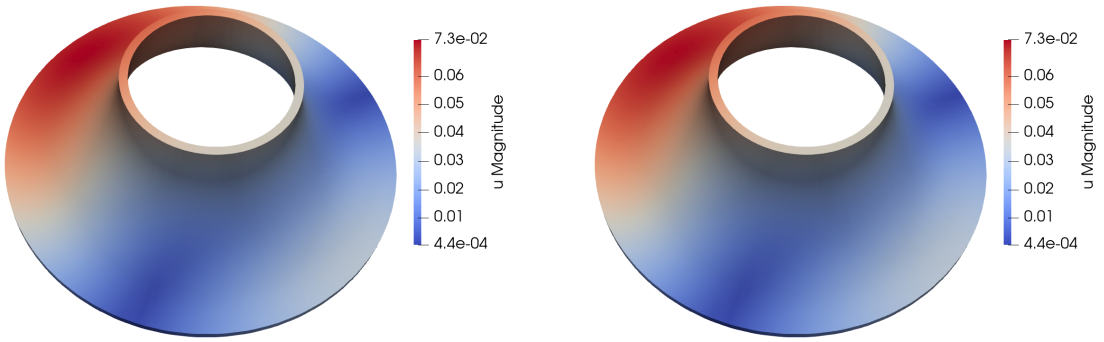


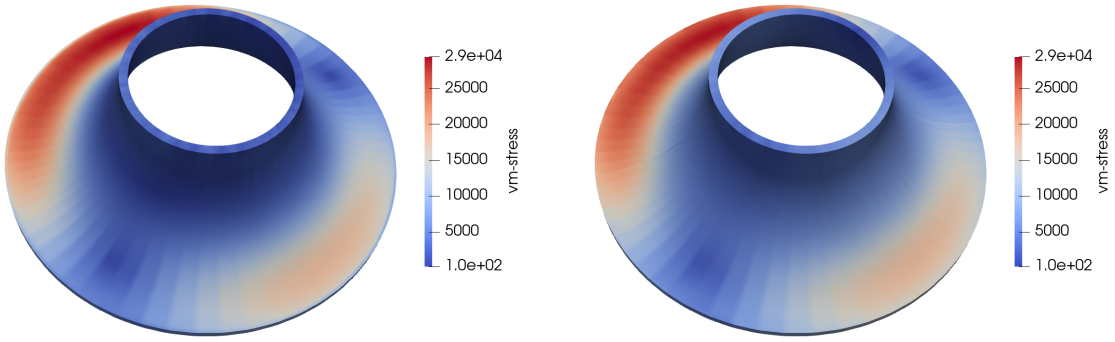
Figure 6.9: Integration cells generated for the geometry from Example 6.6, with the joint modeled as a shell, revealing the refinement level of each involved cover.

To compare the two considered scenarios, we focus our attention on the joint part, and transform the shell solution into a volumetric one following the Kirchhoff-Love ansatz (3.11). The two resulting displacements are shown in Figure 6.10, where no difference can be detected by the naked eye. In Figure 6.11 we picture the Von Mises stress, where certain differences can be observed. For a better assessment, we show the difference between the solid and shell results in Figure 6.12, both in absolute and relative terms. We note that the maximal difference occurs at the interface with the plate, likely due to the mismatch between the solid and shell boundary moments that are coupled by Nitsche's method, while in relative terms a localized but very significant difference can be observed, although limited to the region of lowest Von Mises stress.



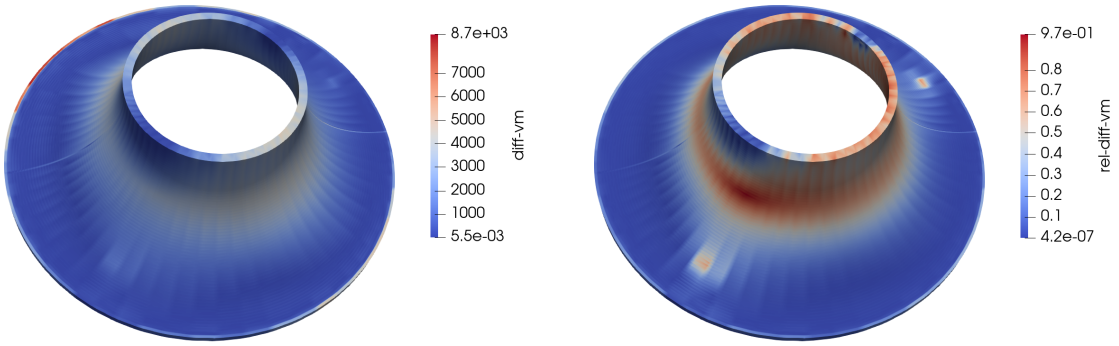
(a) **(b)**

Figure 6.10: Displacements (m) obtained for the joint part from Example 6.6: (a) modeled as a solid, and (b) modeled as a shell (with the solution extended through the thickness as $\mathbf{u} + \zeta\, \boldsymbol{\theta}(\mathbf{u})$).



(a) **(b)**

Figure 6.11: Von Mises stress $\sqrt{1.5\,\mathrm{tr}(\boldsymbol{\sigma}^2) - 0.5\,\mathrm{tr}(\boldsymbol{\sigma})^2}$ (Pa) obtained for the joint part from Example 6.6: (a) modeled as a solid, and (b) modeled as a shell (with $\boldsymbol{\sigma} = \frac{1}{h} A(\mathbf{u}) + \frac{12}{h^3}\, \zeta\, B(\mathbf{u})$).



(a) **(b)**

Figure 6.12: Absolute difference of the Von Mises stress results from Figure 6.11: (a) in absolute terms (Pa), and (b) relative to the shell result.

7 Conclusion

The achievements of this thesis and its attached papers include:

- The rigorous combination of Nitsche's method with the PUM, relying on a stabilization function that is not merely a constant over the whole boundary, and which can be constructed through the solution of many local (and small) generalized eigenvalue problems, instead of a single, big one (see Section 3.5 and Appendix C.1).
- The development of a patch-aggregation approach, which prevents Nitsche stabilization functions from attaining arbitrarily large values for boundary patches with degenerate flat-top regions (see also Appendix C.1).
- The derivation of a Nitsche formulation for the blending of solid and shell models, coupling not only the displacements but also the boundary moments (even if unphysically) of both components at the interface (see Section 3.4).
- The development of projection algorithms for a point into a NURBS curve or surface, which allow the coupling of non-trivial shell geometries (see Appendix A.3).
- The efficient discretization of laminated composite materials with patches spanning at most two laminae and capturing inter-material derivative discontinuities (see Section 4.2).
- The development of efficient smoothers for the multilevel solution of anisotropic and higher-order problems, including laminated composites in either solid or shell form (see Chapter 5 and Appendix C.2).
- And finally the efficient and well-designed implementation of all of the above in the PUMA library (see Appendix B), involving its core `paunt`, `geco`, `xymath` and `xygeo` libraries, which itself entailed:
  - Making PUMA capable of handling derivatives of order higher than 2 in a clean and systematic manner.
  - Significantly improving PUMA's internal PDE representation, allowing the solution of Kirchhoff-Love shell problems to take a few seconds instead of several hours.
  - Redesigning PUMA's integration pipeline to make more intelligent use of computational resources and drastically reduce computation time.

Nevertheless, there are several issues worth mentioning. First of all, for the coupling of solid and shell models, as mentioned in Section 3.4, a compromise has to be made with respect to the shell boundary moments, which makes the formulation not fully rigorous. This prevents convergence to an exact solution in general, but seems not to play a significant role in practice, since refinement through-the-thickness is not considered, and the goal is simply to find a "better" solution than with a single shell model.

Secondly, it is also not clear which kind of shell geometries can accommodate laminated fiber-reinforced composite materials. Planar shells clearly do, but they are the most trivial case of a shell geometry. Cylindrical shells also do, but the possible directions of the fibers are reduced to 2: the axial one and the circumference's tangent (a so-called *cross-ply* laminate). Of course, this is a limitation given by the physical process for constructing the shells, and no such limitation exists in the mathematical model itself. Needless to say, it is the applicability to a real use case which interests us (me at least), and not just the solution of some complicated mathematical model.

Finally, it was perhaps not worth it to implement all these new functionalities, which are aimed for a practical and non-academic usage, into PUMA, a project which is still eminently educational. It remains to be seen whether the thread that I leave here can be picked up by someone else, and whether all the contributions that I made to the PUMA library can be put to some actual use. With respect to this I am not particularly optimistic.

In any case, I consider my work presented here one of the most significant contributions to the PUMA project, since it covers practically all aspects involved in the numerical solution of linear and elliptic PDEs, and not only theoretically, but also taking special care with its practical implementation (whose scope can unfortunately not be fully captured in a written document such as this, even if a humble attempt has been made in Appendix B).

Bibliography

[1] J. Benzaken, J. A. Evans, S. F. McCormick, R. Tamstorf. "Nitsche's method for linear Kirchhoff–Love shells: Formulation, error analysis, and verification". In: *Computer Methods in Applied Mechanics and Engineering* 374 (2021), p. 113544. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113544. URL: https://www.sciencedirect.com/science/article/pii/S0045782520307295.

[2] J. Benzaken, J. A. Evans, R. Tamstorf. "Constructing Nitsche's Method for Variational Problems". In: *Archives of Computational Methods in Engineering* 31.4 (Apr. 2024), pp. 1867–1896. ISSN: 1886-1784. DOI: 10.1007/s11831-023-09953-6. URL: http://dx.doi.org/10.1007/s11831-023-09953-6.

[3] P. G. Ciarlet. *Mathematical Elasticity: Theory of Shells*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2021. DOI: 10.1137/1.9781611976823. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976823. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976823.

[4] P. G. Ciarlet. *Mathematical Elasticity: Three-Dimensional Elasticity*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2021. DOI: 10.1137/1.9781611976786. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976786. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976786.

[5] L. Coradello, J. Kiendl, A. Buffa. "Coupling of non-conforming trimmed isogeometric Kirchhoff–Love shells via a projected super-penalty approach". In: *Computer Methods in Applied Mechanics and Engineering* 387 (Dec. 2021), p. 114187. ISSN: 0045-7825. DOI: 10.1016/j.cma.2021.114187. URL: http://dx.doi.org/10.1016/j.cma.2021.114187.

[6] M. Griebel, M. A. Schweitzer. "A Particle-Partition of Unity Method–Part III: A Multilevel Solver". In: *SIAM J. Sci. Comput.* 24.2 (Feb. 2002), pp. 377–409. ISSN: 1064-8275. DOI: 10.1137/S1064827501395252. URL: https://doi.org/10.1137/S1064827501395252.

[7] M. Griebel, M. A. Schweitzer. "A Particle-Partition of Unity Method–Part V: Boundary Conditions". In: *Geometric Analysis and Nonlinear Partial Differential Equations*. Springer, 2003, pp. 519–542.

[8] M. Griebel, M. A. Schweitzer. "A Particle-Partition of Unity Method–Part VII: Adaptivity". In: *Meshfree Methods for Partial Differential Equations III*. Springer Berlin Heidelberg, 2001, pp. 121–147. ISBN: 9783540462149. DOI: 10.1007/978-3-540-46222-4_8. URL: http://dx.doi.org/10.1007/978-3-540-46222-4_8.

[9] M. H. Gutknecht, S. Röllin. "The Chebyshev iteration revisited". In: *Parallel Computing* 28.2 (Feb. 2002), pp. 263–283. ISSN: 0167-8191. DOI: 10.1016/s0167-8191(01)00139-9. URL: http://dx.doi.org/10.1016/S0167-8191(01)00139-9.

[10] C. Janna, M. Ferronato, G. Gambolati. "A Block FSAI-ILU Parallel Preconditioner for Symmetric Positive Definite Linear Systems". In: *SIAM Journal on Scientific Computing* 32.5 (Jan. 2010), pp. 2468–2484. ISSN: 1095-7197. DOI: 10.1137/090779760. URL: http://dx.doi.org/10.1137/090779760.

[11] C. Janna, M. Ferronato. "Adaptive Pattern Research for Block FSAI Preconditioning". In: *SIAM Journal on Scientific Computing* 33.6 (Jan. 2011), pp. 3357–3380. ISSN: 1095-7197. DOI: 10.1137/100810368. URL: http://dx.doi.org/10.1137/100810368.

[12] C. Janna, M. Ferronato, F. Sartoretto, G. Gambolati. "FSAIPACK: A Software Package for High-Performance Factored Sparse Approximate Inverse Preconditioning". In: *ACM Transactions on Mathematical Software* 41.2 (Feb. 2015), pp. 1–26. ISSN: 1557-7295. DOI: 10.1145/2629475. URL: http://dx.doi.org/10.1145/2629475.

[13] C. Janna, A. Franceschini. "Nesting Approximate Inverses for Improved Preconditioning and Algebraic Multigrid Smoothing". In: *SIAM Journal on Matrix Analysis and Applications* 46.1 (Feb. 2025), pp. 393–415. ISSN: 1095-7162. DOI: 10.1137/24m1679847. URL: http://dx.doi.org/10.1137/24M1679847.

[14] P. Jiménez Recio, M. A. Schweitzer. "A Partition of Unity construction of the stabilization function in Nitsche's method for variational problems". In: *Computer Methods in Applied Mechanics and Engineering* 426 (June 2024), p. 117002. ISSN: 0045-7825. DOI: 10.1016/j.cma.2024.117002. URL: http://dx.doi.org/10.1016/j.cma.2024.117002.

[15] P. Jiménez Recio, M. A. Schweitzer. *Chebyshev smoothing with adaptive block-FSAI preconditioners for the multilevel solution of higher-order problems*. 2025. arXiv: 2509.06744 [math.NA]. URL: https://arxiv.org/abs/2509.06744v2.

[16] J. Lottes. "Optimal polynomial smoothers for multigrid V-cycles". In: *Numerical Linear Algebra with Applications* 30.6 (June 2023). ISSN: 1099-1506. DOI: 10.1002/nla.2518. URL: http://dx.doi.org/10.1002/nla.2518.

[17] D. Schöllhammer, T. P. Fries. "Kirchhoff–Love shell theory based on tangential differential calculus". In: *Computational Mechanics* 64.1 (Nov. 2018), pp. 113–131. ISSN: 1432-0924. DOI: 10.1007/s00466-018-1659-5. URL: http://dx.doi.org/10.1007/s00466-018-1659-5.

[18] M. A. Schweitzer. *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*. Springer Berlin Heidelberg, 2003. ISBN: 9783642593253. DOI: 10.1007/978-3-642-59325-3. URL: http://dx.doi.org/10.1007/978-3-642-59325-3.

[19] M. A. Schweitzer. "An adaptive hp-version of the multilevel particle–partition of unity method". In: *Computer Methods in Applied Mechanics and Engineering* 198.13–14 (Mar. 2009), pp. 1260–1272. ISSN: 0045-7825. DOI: 10.1016/j.cma.2008.01.009. URL: http://dx.doi.org/10.1016/j.cma.2008.01.009.

[20] M. A. Schweitzer. "An Algebraic Treatment of Essential Boundary Conditions in the Particle-Partition of Unity Method". In: *SIAM J. Sci. Comput.* 31.2 (Feb. 2009), pp. 1581–1602. ISSN: 1064-8275. DOI: 10.1137/080716499. URL: http://dx.doi.org/10.1137/080716499.

[21] M. Sedlacek. "Sparse approximate inverses for preconditioning, smoothing, and regularization". PhD thesis. Technische Universität München, 2012.

[22] A. Ziegenhagel. "Robust and Efficient Treatment of Industrial-Grade CAD Geometries for a Flat-Top Partition of Unity Method". PhD thesis. Rheinische Friedrich-Wilhelms-Universität Bonn, Dec. 2022. URL: https://hdl.handle.net/20.500.11811/10526.

A Computational geometry

For completeness, in this appendix I will gather the projection algorithms that I have implemented in PUMA for the coupling of shells, either via solid-shell blending or simple shell-to-shell coupling (which is not a part of this thesis).

Before delving into the algorithms themselves, I will introduce the minimal amount of information necessary about Bézier parametrizations of curves and surfaces, as well as Non-Uniform Rational B-Spline (NURBS) parametrizations.

The projection algorithms receive a point in physical space and compute its closest point in the curve or surface (and its corresponding parameter(s)). If the original point lies already in the curve or surface, such algorithms can be used to retrieve the parameter(s) producing that point (which is sometimes referred to as "inverting" the curve or surface).

A.1 Bézier curves and surfaces

A Bézier parametrization of either a curve or a surface relies on the so-called Bernstein polynomial basis, which for each dergee $n > 0$ can be written as

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i \in \{0, \dots, n\}, \quad t \in [0, 1].$$

Among the multiple properties of these polynomials, let us mention the nodal property at the endpoints $B_i^n(0) = \delta_{i,0}$, $B_i^n(1) = \delta_{i,n}$, and the partition of unity property in the definition interval,

$$0 \leq B_i^n(t) \leq 1 \quad \forall i \in \{0, \dots, n\}, \quad \text{and} \quad \sum_{i=0}^{n} B_i^n(t) = 1, \quad \forall t \in [0, 1].$$

Additionally, the product of two Bernstein polynomials satisfies

$$B_i^n(t) B_j^m(t) = \frac{\binom{n}{i}\binom{m}{j}}{\binom{n+m}{i+j}} B_{i+j}^{n+m}(t).$$

A Bézier curve of degree n in $\mathbb{R}^d$ (with $d \in \{2, 3\}$) is then any curve that can be written as a linear combination of the Bernstein polynomials of degree n, i.e.

$$C(t) = \sum_{i=0}^{n} P_i B_i^n(t), \quad t \in [0, 1],$$

where $P_0, P_1, \dots, P_n \in \mathbb{R}^d$ are called control points.

From the partition of unity property of the Bernstein polynomial basis, it follows that any Bézier curve is contained in the convex hull of its control points (since every evaluation $C(t)$ is a convex combination of the control points).

A Bézier surface of degrees (n, m) in $\mathbb{R}^3$, on the other hand, is any surface that can be written as

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j} B_i^n(u) B_j^m(v), \quad (u, v) \in [0, 1]^2,$$

where, again, the $P_{i,j} \in \mathbb{R}^d$ are called control points. Since the tensor product of Bernstein polynomials also forms a partition of unity in $[0, 1]^2$, any Bézier surface is also contained in the convex hull of its control points.

Additional degrees of freedom can be introduced in the form of weights, giving rise to *rational* Bézier curves and surfaces. A rational Bézier curve of degree n can be written as

$$C(t) = \frac{\sum_{i=0}^{n} w_i P_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)}, \quad t \in [0, 1],$$

while a rational Bézier surface of degrees (n, m) can be written as

$$S(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} P_{i,j} B_i^n(u) B_j^m(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} B_i^n(u) B_j^m(v)}, \quad (u, v) \in [0, 1]^2,$$

with $w_i > 0$ and $w_{i,j} > 0$ being the weights in each case. In both cases, the convex hull property with respect to the control points is conserved.

In all cases, Bézier curves and surfaces are interpolatory at the endpoints of its parameter space, meaning $C(0) = P_0$ and $C(1) = P_n$ for curves, and

$$S(0, 0) = P_{0,0}, \quad S(0, 1) = P_{0,m}, \quad S(1, 0) = P_{n,0}, \quad S(1, 1) = P_{n,m}$$

for surfaces. Both curves and surfaces can be defined on parameter domains other than $[0, 1]$ and $[0, 1]^2$ via linear reparametrizations, i.e.

$$\hat{C}(\hat{t}) = C\left(\frac{\hat{t} - t_0}{t_1 - t_0}\right), \quad \hat{t} \in [t_0, t_1],$$
$$\hat{S}(\hat{u}, \hat{v}) = S\left(\frac{\hat{u} - u_0}{u_1 - u_0}, \frac{\hat{v} - v_0}{v_1 - v_0}\right), \quad (\hat{u}, \hat{v}) \in [u_0, u_1] \times [v_0, v_1].$$

Note that both Bézier curves and surfaces can be split: a curve into two Bézier curves at any parameter $t_s \in (t_0, t_1)$, and a surface at any $u_s \in (u_0, u_1)$ or $v_s \in (v_0, v_1)$. The projection algorithms to be introduced later will split a curve into 2 at its midpoint parameter $t_s = \frac{t_0+t_1}{2}$, and analogously a surface into 4, in this case first splitting at $u_s = \frac{u_0+u_1}{2}$, and then at $v_s = \frac{v_0+v_1}{2}$.

Finally, let us point out that, for a surface S, the result of fixing any of the two parameters to a bound, i.e. $S(t, 0)$, $S(t, 1)$, $S(0, t)$ or $S(1, t)$, is a Bézier curve, with control points and weights taken from the corresponding fixed dimension and bound.

A.2 NURBS curves and surfaces

NURBS curves and surfaces are more complicated than their Bézier counterparts, since in addition to control points and weights they allow for knots in their parameter space. We will briefly define them for completeness, but the only necessary detail for the subsequent projection algorithms is that they can be split into rational Bézier curves and surfaces. The projection algorithms operate on NURBS curves and surfaces only implicitly, since they actually work with collections of Bézier curves and surfaces.

Before introducing NURBS curves and surfaces, we first need to define B-spline basis functions for a knot vector $\mathcal{T} = [t_0, t_1, \ldots, t_k]$ with non-decreasing entries, i.e. $t_i \leq t_{i+1}$. The B-spline basis functions of degree p, $N_{i,p}(t)$ for $i \in \{0, \ldots, k-p-1\}$, are defined recursively as

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$
$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t), \quad p > 0.$$

The bounds of the knot vector $[t_0, t_k]$ give the parameter bounds of the curve, and the existence of at least 2 knots in the knot vector implies that $k \geq p + 2$. If some knot is repeated s times in the knot vector, it is commonly referred to as a single knot with *multiplicity* s.

The B-spline basis functions share the nodal and partition of unity properties of the Bernstein polynomials. In fact, the Bernstein polynomials of degree n are themselves B-spline basis functions of degree n for the choice of knot vector $t_0 = \cdots = t_n = 0$ and $t_{n+1} = \cdots = t_{2n+1} = 1$.

Analogously as for Bézier curves and surfaces, a NURBS curve of degree p can be written as

$$C(t) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,p}(t)}{\sum_{i=0}^{n} w_i N_{i,p}(t)}, \quad t \in [t_0, t_k],$$

while a NURBS surface of degrees (p, q) can be written as

$$S(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{ij} P_{ij} N_{i,p}(u) N_{j,q}(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{ij} N_{i,p}(u) N_{j,q}(v)}, \quad (u, v) \in [u_0, u_k] \times [v_0, v_\ell],$$

in this case with 2 knot vectors $\mathcal{U} = [u_0, u_1, \ldots, u_k]$ and $\mathcal{V} = [v_0, v_1, \ldots, v_\ell]$, giving rise to the basis functions $N_{i,p}$ and $N_{j,q}$, respectively.

By a process known as knot insertion, a given NURBS curve can be represented on a different basis of the same degree, but with a knot vector obtained by adding a new knot $t' \in [t_0, t_k]$. Inserting each unique knot until its multiplicity matches the degree p effectively yields a split representation of the curve within each non-trivial interval $[t_i, t_{i+1}]$ with $t_i < t_{i+1}$. Each such representation is a Bézier curve reparametrized to the corresponding subinterval. Thus, any NURBS curve can be represented as a list of Bézier curves, one for each non-trivial subinterval generated by its knot vector.

The same is true for NURBS surfaces with respect to any of its 2 dimensions, in which case the split representation corresponds to subdivisions of the original parameter domain of the form $[u_i, u_{i+1}] \times [v_j, v_{j+1}]$ with $u_i < u_{i+1}$ and $v_j < v_{j+1}$. In this case, each such representation is a reparametrized Bézier surface.

For more details on curves and surfaces, both of NURBS and Bézier type, we refer to the monographs [5, 9].

A.3 Projection algorithms

For projecting a point to a NURBS curve or surface, we will rely on [8] and the references therein, among others [1, 7, 11]. Both algorithms (for curves and surfaces) operate on collections of Bézier curves or surfaces, which are iteratively refined until a single closest point can be found within them. A key component of both algorithms is the maintenance of a current tentative closest point Q to the query point P (and the corresponding distance d), which is used to discard remaining sub-curves and sub-surfaces whenever they can be guaranteed to lie further away from P than Q, by the use of certain cheap geometrical checks.

Let us remark that, when projecting a point to a Bézier curve or surface, the actual solution might not be unique: multiple different points in the curve or surface may be at the same distance from the query point. Given our use cases, in such a scenario we will be interested in retrieving any single point. For that reason, if during the iteration we have found a point Q at a distance r, we will be interested only on improving such solution, i.e. finding a new point Q' at a distance $r' < r$.

A.3.1 Projection of a point onto a NURBS curve

For simplicity, let us first consider the 2-dimensional case. To begin with, we introduce the concept of *clipping hyperplanes*. First of all, from the convex hull property, if all control points of a Bézier curve lie on a given half-space of $\mathbb{R}^2$, then the whole curve does. As a result, for a point $P \in \mathbb{R}^2$ and a distance $r > 0$, we can define hyperplanes such that, if all control points of a Bézier curve C fall on one side of them, then $d(C, P) \geq r$. These tests can be written as

$$(Q - P) \cdot \hat{n} \geq r \quad \text{for all control points } Q \text{ of } C,$$

where $\hat{n}$ is the hyperplane's normal (pointing towards the corresponding half-space).

Two such hyperplanes are illustrated in Figure A.1, those with normal directions $(\pm 1, 0)$. Of course, there is an infinite number of such hyperplanes, all tangent to the ball of radius r centered at P. Following [8], we pick a small, discrete and simple subset of them: those with normals in $\{(\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1)\}$, as illustrated in Figure A.2. This allows to reduce the inner products $(Q - P) \cdot \hat{n}$ to subtractions and additions of the components of $Q - P$.

For a query point P, a tentative minimal distance r and a given Bézier curve, we can easily check whether all its control points lie on the discarding side of any of the 8 hyperplanes. If that is the case, then we know that the Bézier curve is at a distance larger or equal to r from P, and thus we may discard it.

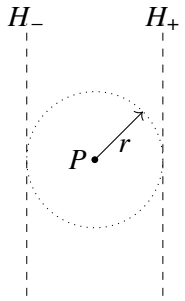


Figure A.1: Vertical hyperplanes used for clipping away 2-dimensional Bézier curves: every curve whose control points lie all to the right of H_+ or all to the left of H_- (including the hyperplane itself) is necessarily at a distance to P greater or equal to r.

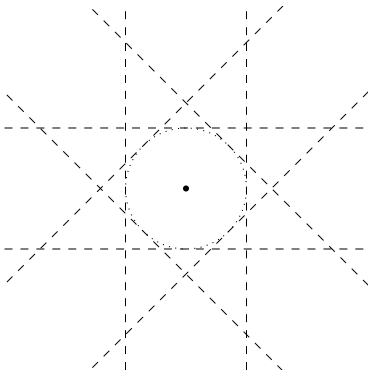


Figure A.2: All 8 hyperplanes considered for clipping away 2-dimensional Bézier curves.

This can be trivially generalized to 3-dimensional curves, where we choose the set of 26 hyperplanes given by the normals $\{(\pm1, 0, 0), (0, \pm1, 0), (0, 0, \pm1), (\pm1, \pm1, 0), (\pm1, 0, \pm1), (0, \pm1, \pm1), (\pm1, \pm1, \pm1)\}$.

If a Bézier curve could not be discarded by the previous half-space checks, there is a further geometrical check that can be performed. Based also on the convex hull property and half-spaces, we can easily check whether one of the two endpoints is the closest point to the query point P within the curve: at the endpoint E (which is a control point of the curve), we place a hyperplane going through it with normal $E - P$. Note that we may assume that $E \neq P$, otherwise E would itself be a closest point. If all other control points of the curve lie in the half-space not containing P, then E is the closest point to P in the curve. The actual condition can be written as

$$(Q - E) \cdot (E - P) \geq 0$$

for all control points Q (excluding E itself, but for $Q = E$ the condition also trivially holds). In our previous terms, this is effectively a half-space test with normal $\hat{n} = \frac{E-P}{\|E-P\|}$ and distance $\|E - P\|$, since the above inequality is equivalent to

$$(Q - P) \cdot \hat{n} \geq (E - P) \cdot \hat{n} = \|E - P\| .$$

Therefore, assuming that both endpoints have already been considered, and thus $r \leq \|E - P\|$, we can replace the condition by a half-space test with the same normal and the current minimal distance r. By doing so, the test no longer checks whether the endpoint E is the closest point, but (as in the previous cases) whether the curve is further away than r from P. We restrict this check to the closest endpoint.

If none of the half-space checks allowed us to discard the curve, the algorithm proceeds to work with the squared distance function $\|C(t) - P\|^2$. If a unique minimizer in its interval domain can be guaranteed, then a root-finding algorithm is used, otherwise the curve is split into two, and the original curve is replaced by its two components in the pending list.

For simplicity, we assume that $C\colon [0,1] \mapsto \mathbb{R}^d$, otherwise a linear re-parametrization from the original domain $[t_0, t_1]$ to $[0,1]$ is implied. Any root retrieved in the $[0,1]$ interval can be trivially mapped to the original interval.

The sufficient condition for uniqueness of a minimizer of $\|C(t) - P\|^2$ is based on the fact that it can itself be written as a scalar Bézier function, since

$$\|C(t) - P\|^2 = \left\| \frac{\sum_{i=0}^{n} w_i P_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)} - P \right\|^2 = \left\| \frac{\sum_{i=0}^{n} w_i (P_i - P) B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)} \right\|^2$$

and the inner product of two Bézier curves is itself a Bézier function (see e.g. [4]). In this case

$$D(t) \coloneqq \|C(t) - P\|^2 = \frac{\sum_{k=0}^{2n} \tilde{w}_k D_k B_k^{2n}(t)}{\sum_{k=0}^{2n} \tilde{w}_k B_k^{2n}(t)}$$

with weights

$$\tilde{w}_k = \frac{1}{\binom{2n}{k}} \sum_{j=\max(0,k-n)}^{\min(n,k)} \binom{n}{j} \binom{n}{k-j} w_j w_{k-j}$$

and control values (i.e. scalar control points)

$$D_k = \frac{1}{\binom{2n}{k} \tilde{w}_k} \sum_{j=\max(0,k-n)}^{\min(n,k)} \binom{n}{j} \binom{n}{k-j} w_j w_{k-j} (P_j - P) \cdot (P_{k-j} - P).$$

In the case of a non-rational curve, $\tilde{w}_k = 1$ (as well as the original weights w_j).

At this point, we can check if all control values satisfy $D_k \geq r^2$, in which case we may discard the curve. Otherwise, we proceed to check if $D(t)$ has a single local minimum in $[0,1]$ through its so-called *hodograph*, which is nothing else than its derivative $D'(t)$ in Bézier form. If $C(t)$ is non-rational (and thus $D(t)$ is non-rational too), the hodograph of $D(t)$ is trivially

$$D'(t) = 2n \sum_{k=0}^{2n-1} (D_{k+1} - D_k) B_k^{2n-1}(t).$$

In the rational case, we have

$$D'(t) = \left(\frac{N(t)}{W(t)}\right)' = \frac{N'(t)W(t) - N(t)W'(t)}{W(t)^2},$$

and thus the computation is more involved. We refer to [6, 10] for the specific procedure.

To derive a sufficient condition for uniqueness of a root of $D'(t)$, we will rely on the *variation diminishing* property of Bézier curves (see e.g. [5, Chapter 5]). According to this property, $D'(t)$ cannot cross the value 0 any more times than its sequence of control values does. Therefore, a sufficient condition for the existence of a unique root of $D'(t)$ is that its sequence of control values crosses 0 only once (existence follows from $D'(t)$ being continuous and having different signs at its endpoints). Under the additional condition that $D'(0) < 0 < D'(1)$, the root is a unique local minimum of $D(t)$. Additionally, if all control points are nonnegative (resp. nonpositive), that means that the first endpoint (resp. the last one) is a closest point in the curve. If the sequence crosses 0 more than once, then uniqueness of a root cannot be guaranteed, and as stated above, we split the Bézier curve into two.

Remark A.1. For the non-rational case, our use of the (trivial) hodograph is equivalent to [1, Property 2], whose authors do not explicitly adapt to the rational case. We believe the sufficient condition for uniqueness from [1, Property 2] to be wrong for the rational case, since the variation diminishing property must be applied to the hodograph (as done here).

Remark A.2. In the rational case, we may alternatively work with the *scaled hodograph*

$$W(t)^2 D'(t) = N'(t)W(t) - N(t)W'(t),$$

which is not as expensive to compute as the actual hodograph.

Remark A.3. For a given query point P, we may instead translate the original NURBS curve by $-P$ (which amounts to translating its control points) and find its closest point to the origin.

A formal description relying on the previous concepts is presented in Algorithm A.1.

A.3.2 Projection of a point onto a NURBS surface

The projection into a surface relies on similar geometric checks as the curve algorithm, but the 2-dimensionality of the parameter space introduces significant complications with respect to the simple curve case. In addition to that, the curve algorithm itself is used to project into surface boundaries when necessary.

The 26 half-space checks that were performed on the control points of 3-dimensional Bézier curves are directly valid also for the control points of Bézier surfaces. The subsequent half-space test based on a curve's closest endpoint can also be directly translated to the surface case, with the obvious caveat that in this case there are 4 endpoints to consider.

If none of these half-space checks allowed us to discard the curve, we proceed to check whether the closest point can be guaranteed to lie on any of the 4 boundaries of the considered surface. For simplicity, let us consider the case of $S(0, v)$, i.e. the boundary part which results from fixing

Algorithm A.1 Projection of a point to a NURBS curve

Input: A query point $P \in \mathbb{R}^d$ and a NURBS curve $C[t_0, t_1] \mapsto \mathbb{R}^d$ with an available splitting into Bézier components $\{C_i\}_{i=0}^n$.

Output: An element $t^\star \in \arg\min_{t \in [t_0, t_1]} \|C(t) - P\|$ and its corresponding point $Q^\star = C(t^\star)$.

Initialize the tentative closest point $Q = C(t_0)$, i.e. C's first control point (C_0's first endpoint), and its associated parameter $t_Q = t_0$.

for $i = 0, \ldots, n$ **do**
    Let E be C_i's second endpoint.
    if $\|E - P\| < \|Q - P\|$ **then** $Q \leftarrow E$ and update t_Q accordingly.

Initialize $\mathcal{B} = \{C_i\}_{i=0}^n$ a set of Bézier curves to yet visit.

while $\mathcal{B} \neq \emptyset$ **do**
    Take a curve B from $\mathcal{B}$ and remove it from the set.

    ▷ The two endpoints of B have already been considered as potential closest point, so none of them can be closer to P than Q.

    Based on the current minimal distance $\|Q - P\|$, perform half-space tests on B's control points (8 such tests in $\mathbb{R}^2$, 26 in $\mathbb{R}^3$).

    if any half-space test is positive **then** continue

    Take the closest endpoint E and perform one further half-space test with normal $\frac{E-P}{\|E-P\|}$.

    if the half-space test is positive **then** continue

    Compute the Bézier representation of $D(t) = \|B(t) - P\|^2$.

    if $D_i \geq \|Q - P\|^2$ for all control values D_i of D **then** continue

    Compute the hodograph of $D(t)$, i.e. the Bézier representation of $D'(t)$ (possibly scaled in the rational case), keeping only the control values H_i.

    if all $H_i \geq 0$ or all $H_i \leq 0$ **then** continue

    if the sequence of H_i starts negative and ends positive, crossing the value 0 only once **then**
        Use a root-finding algorithm to minimize $\|B(t) - P\|^2$.
        if $\min \|B(t) - P\|^2 < \|Q - P\|^2$ **then** update Q and t_Q
    else
        Split B into two Bézier curves at its midpoint parameter, yielding a new endpoint M, and append them to $\mathcal{B}$.

        if $\|M - P\| < \|Q - P\|$ **then** $Q \leftarrow M$ and update t_Q to B's midpoint parameter

return t_Q and Q

▷ Note that we may directly return the tentative results whenever $Q = P$.

the first component to its lower bound. A sufficient condition for the closest point to be on this boundary is, following [11, Theorem 4.4], that

$$S_u(u,v)\cdot(P_{0,j}-P)\geq 0,\quad \forall(u,v)\in[0,1]^2,\quad \forall j\in\{0,\dots,m\},$$

where $\{P_{0,j}\}_{j=0}^m$ are the control points of S associated to the $S(0,v)$ boundary part. We can fully discretize the inequality by computing the u-hodograph of S (that is, the Bézier representation of $S_u(u,v)$), with control points $H_k^{(u)}$, after which we simply need to check whether $H_k^{(u)}\cdot(P_{0,j}-P)\geq 0$ for all such control points.

Note that, in the case of fixing u to its upper bound, we need to pick the control points corresponding to the other end, and also the inequality has to be inverted, leading us to check whether $H_k^{(u)}\cdot(P_{n,j}-P)\leq 0$ for all involved k and j. The case of fixing the v-component is analogous, involving the v-hodograph of S, its control points $H_k^{(v)}$, and the surface control points $P_{i,0}$ or $P_{i,m}$. Note that scaled hodographs can be used in the rational case.

Remark A.4. In the non-rational case, these checks correspond to the ones from [8], but hodographs are not explicitly used there, which makes the algorithm incomplete for the rational case.

If any of these tests is positive, then we may find the closest point within the corresponding boundary part with Algorithm A.1, otherwise further tests are required to check uniqueness of a local minimizer of $D(u,v)=0.5\,\|S(u,v)-P\|^2$ in the surface's interior. Interior local minimizers are characterized by

$$\begin{cases} D_u(u,v)=S_u(u,v)\cdot(S(u,v)-P)=0,\\ D_v(u,v)=S_v(u,v)\cdot(S(u,v)-P)=0,\end{cases}\tag{A.1}$$

and a minimizer is unique if the tangent cones of the constraints intersect only trivially (see [3]). We will operate with S_u and S_v as follows.

First of all, we replace S_u and S_v above by the corresponding scaled hodographs, which were used previously and which we will denote $\tilde{S}_u$ and $\tilde{S}_v$, and then compute the Bézier representations of $\tilde{S}_u(u,v)\cdot(S(u,v)-P)$ and $\tilde{S}_v(u,v)\cdot(S(u,v)-P)$ (let us refer to them, respectively, as $F(u,v)$ and $G(u,v)$). In the rational case, we may keep only the numerator of both F and G, which we do by absorbing the weights into the control values by multiplication, and then ignoring them (i.e. setting them to 1). Needless to say, none of these transformations affects the solution space of the problem, i.e. the solutions of Equation (A.1) are the same as those of $F(u,v)=G(u,v)=0$. At this point, if either F or G have only one sign over $[0,1]^2$ (which we can check through its control values), then the local minimizer must lie on a boundary part:

- if $F\geq 0$, then $D_u\geq 0$, and the minimizer must be on the $\{0\}\times[0,1]$ boundary part,
- if $F\leq 0$, then $D_u\leq 0$, and the minimizer must be on the $\{1\}\times[0,1]$ boundary part,
- if $G\geq 0$, then $D_v\geq 0$, and the minimizer must be on the $[0,1]\times\{0\}$ boundary part,
- if $G\leq 0$, then $D_v\leq 0$, and the minimizer must be on the $[0,1]\times\{1\}$ boundary part.

If none of that is the case, we check for a trivial intersection of the tangent cones of F and G, i.e.

$$\{\alpha\nabla F(u,v):\ \alpha\in\mathbb{R},\ 0<u,v<1\}\cap\{\beta\nabla F(u,v):\ \beta\in\mathbb{R},\ 0<u,v<1\}=\{\mathbf{0}\},$$

which, as indicated in [3, 8], would imply uniqueness of a solution in the interior (provided a solution exists at all). For this, we need (once again) to compute hodographs, this time F_u, F_v, G_u and G_v. We may then combine F_u and F_v into a single Bézier $\mathbb{R}^2$-function as

$$\nabla F(u, v) = \begin{pmatrix} F_u(u, v) \\ F_v(u, v) \end{pmatrix},$$

ensuring that degrees match via degree elevation, and analogously with ∇G. This allows us to compute the tangent cone of the control points of ∇F (or ∇G), which contains the tangent cone of ∇F (resp. ∇G). In Figure A.3 we illustrate tangent cones and their possible intersections.

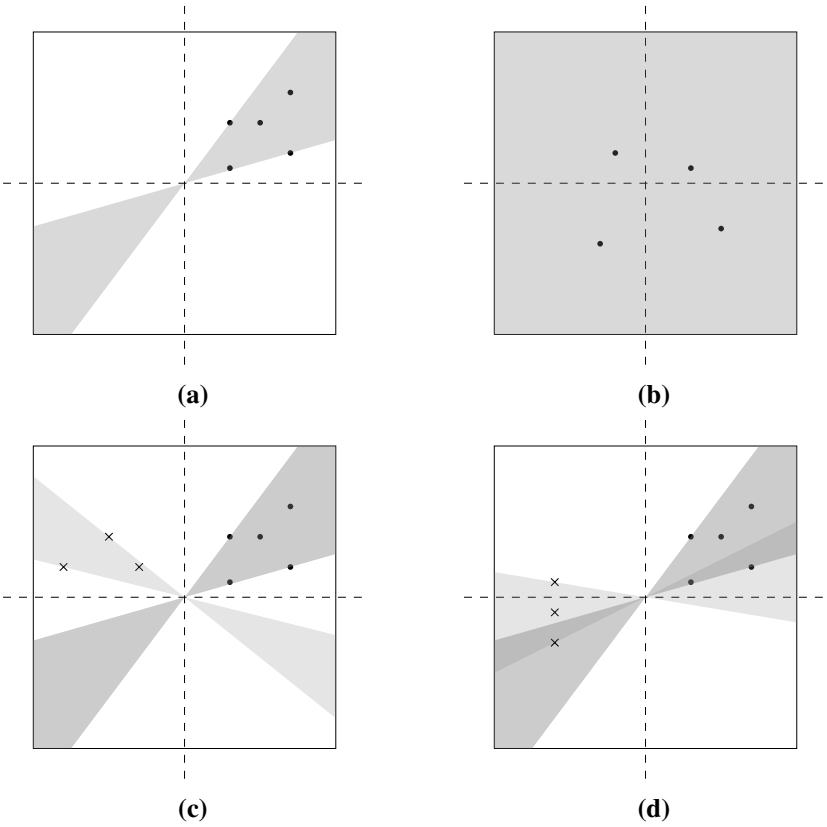


Figure A.3: Illustration of cones associated to sets of control points: (a) a non-trivial cone; (b) a trivial cone; (c) two cones intersecting only at the origin; (d) two cones with non-trivial overlap.

If the tangent cones do not intersect only at the origin, we split the surface into 4 and append the results to the pending list, otherwise a root-finding algorithm is used. If the algorithm does not converge, or if the result does not correspond to a local minimum (whenever the hessian matrix of $\|S(u, v) - P\|^2$ has a negative eigenvalue at it), the surface is also split into 4. If, on the other hand, a local minimizer is found, we compare the result against the tentative projection result (and update the latter if necessary).

Our description of the algorithm is now complete and, as before, we provide a more formal description in Algorithm A.2.

Algorithm A.2 Projection of a point to a NURBS surface

Input: A query point $P \in \mathbb{R}^3$ and a NURBS surface $S[u_0, v_1] \times [v_0, v_1] \mapsto \mathbb{R}^3$ with an available splitting into Bézier components $\{S_i\}_{i=0}^n$.

Output: An element $(u^\star, v^\star) \in \arg\min_{(u,v)\in[u_0,v_1]\times[v_0,v_1]} \|S(u, v) - P\|$ and its corresponding point $Q^\star = S(u^\star, v^\star)$.

Initialize the tentative closest point Q and its associated parameters (u_Q, v_Q) as those of the closest endpoint of all Bézier components $\{S_i\}$. Initialize $\mathcal{B} = \{S_i\}$, the set of Bézier surfaces to yet visit.

while $\mathcal{B} \neq \emptyset$ **do**

Take a surface B from $\mathcal{B}$ and remove it from the set.

▷ The four endpoints of B have already been considered as potential closest point, so none of them can be closer to P than Q.

Based on the current minimal distance $\|Q - P\|$, perform 26 half-space tests on B's control points.

if any half-space test is positive **then** continue

Take the closest endpoint E and perform one further half-space test with normal $\frac{E-P}{\|E-P\|}$.

if the half-space test is positive **then** continue

Compute the scaled hodographs $\tilde{B}_u$ and $\tilde{B}_v$ of B, and use their control points to check whether the closest point of B to P can be guaranteed to lie in any of its 4 boundary parts.

if the closest point lies in two contiguous boundary parts **then** continue

if the closest point lies in a boundary part **then** find the closest point in it, update Q if necessary and continue

With the scaled hodographs $\tilde{B}_u$ and $\tilde{B}_v$, compute

$$F(u, v) = \tilde{B}_u(u, v) \cdot (B(u, v) - P), \quad G(u, v) = \tilde{B}_v(u, v) \cdot (B(u, v) - P)$$

. in Bézier form and, in the rational case, keep only their numerators.

if all control points of either F or G have the same sign **then** the closest point in B lies on a boundary part: find it, update Q if necessary and continue

Compute ∇F and ∇G in Bézier form (as combination of u- and v-hodographs).

if the tangent cones of their control points intersect only at the origin **then**

For $D(u, v) = \|B(u, v) - P\|^2$, attempt to solve $D_u(u, v) = D_v(u, v) = 0$.

if a root is found and it is a local minimizer **then** update Q if necessary and continue

Split B into 4 Bézier patches at its midpoint parameter and append them to $\mathcal{B}$. Visit the new 5 endpoints and update Q if necessary.

return Q and its associated parameters.

Projection of a point onto a trimmed NURBS surface

In practical cases, see e.g. [2], shell parts are usually not represented as simple NURBS surfaces, but as *trimmed* NURBS surfaces: that means that the parameter domain is not some rectangular domain, but a subdomain of it, with boundaries represented by NURBS curves (in the 2-dimensional parameter space). A simple illustration is provided in Figure A.4.

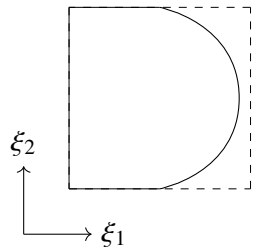


Figure A.4: A trimmed NURBS domain.

For the coupling of shell parts, we are thus interested in projecting a point in physical space to a certain boundary part of the trimmed NURBS surface. We do this first by projecting into the surface (ignoring the trimming subdomain), and then projecting the obtained 2D parameter to the trimming NURBS curve in parameter space associated to the target boundary part. Although in general the combined map is not itself a projection, it allows us to map between shell boundaries, and it works as a projection if the involved shells match exactly at their boundaries (which is however not always the case in real-world scenarios).

Numerical experiments

To assess the accuracy and efficiency of the proposed algorithms, we will perform some tests on the (nonrational) NURBS surface shown in Figure A.5.

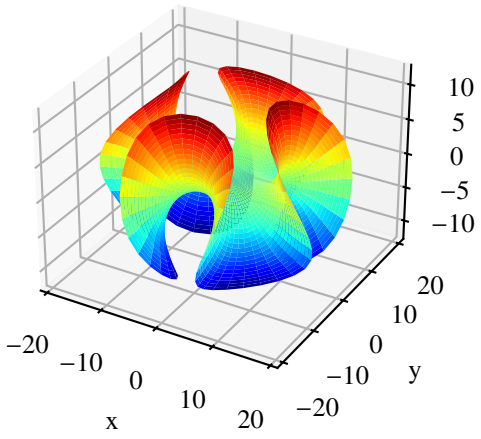


Figure A.5: Enneper surface with 4-fold symmetry, taken from https://www.craftsmanspace.com/free-3d-models/enneper-3d-surface.html.

First of all, we assess the accuracy of our algorithm by generating a regular grid of 100×100 nodes in the rectangular parameter domain, then evaluating the surface at each node, and finally projecting the resulting point back into the surface. In Figure A.6 we report the distances between the original surface point q and the projection result $P(q)$. As we can see, while most of the results fall below 10^{-14}, a fraction of them are in the interval $(10^{-9}, 10^{-6})$. Although this is negligible in comparison with the surface's dimensions, it is nevertheless undesired. Still, we have not been able to figure out the cause and whether it can be avoided.

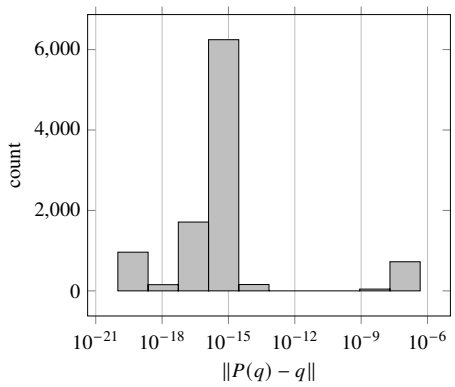
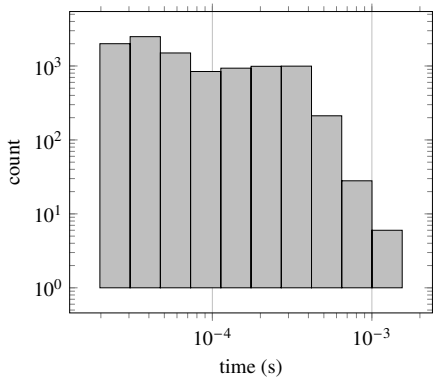
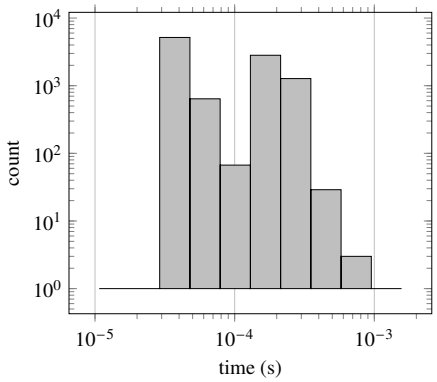


Figure A.6: Histogram of euclidean distance results between the original point and the projected point for 10,000 points in the Enneper surface from Figure A.5. We have added 10^{-20} to all results in order to gather all "very small" values together.

Secondly, in order to assess the efficiency of our point-to-surface projection algorithm, we first randomly generate 10,000 points from a uniform distribution in the (tight) bounding box of the surface and project them into the surface, the results of which we show in Figure A.7a. As we can see, most measurements fall below 1 ms (and all above 0.01 ms). As a second test, we generate 10,000 points on the surface boundary (this time not randomly, but simply by picking equidistant points from the corresponding side of the rectangular parameter domain) and project them into the surface. We show the new results in Figure A.7b, where again most (if not all) time measurements fall below 1 ms.



(a) Results for 10,000 randomly generated points in the bounding box of the surface.



(b) Results for 10,000 points on the boundary of the surface.

Figure A.7: Time measurements for two projection tests into the Enneper surface from Figure A.5.

References

[1] X.-D. Chen, J.-H. Yong, G. Wang, J.-C. Paul, G. Xu. "Computing the minimum distance between a point and a NURBS curve". In: *Computer-Aided Design* 40.10–11 (Oct. 2008), pp. 1051–1054. ISSN: 0010-4485. DOI: 10.1016/j.cad.2008.06.008. URL: http://dx.doi.org/10.1016/j.cad.2008.06.008.

[2] L. Coradello, J. Kiendl, A. Buffa. "Coupling of non-conforming trimmed isogeometric Kirchhoff–Love shells via a projected super-penalty approach". In: *Computer Methods in Applied Mechanics and Engineering* 387 (Dec. 2021), p. 114187. ISSN: 0045-7825. DOI: 10.1016/j.cma.2021.114187. URL: http://dx.doi.org/10.1016/j.cma.2021.114187.

[3] G. Elber, M.-S. Kim. "Geometric constraint solver using multivariate rational spline functions". In: *Proceedings of the sixth ACM symposium on Solid modeling and applications*. SM01. ACM, May 2001. DOI: 10.1145/376957.376958. URL: http://dx.doi.org/10.1145/376957.376958.

[4] R. Farouki, V. Rajan. "Algorithms for polynomials in Bernstein form". In: *Computer Aided Geometric Design* 5.1 (June 1988), pp. 1–26. ISSN: 0167-8396. DOI: 10.1016/0167-8396(88)90016-7. URL: http://dx.doi.org/10.1016/0167-8396(88)90016-7.

[5] J. Gallier. *Curves and surfaces in geometric modeling: theory and algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. ISBN: 1558605991.

[6] D.-S. Kim, T. Jang, H. Shin, J. Park. "Rational Bézier form of hodographs of rational Bézier curves and surfaces". In: *Computer-Aided Design* 33.4 (Apr. 2001), pp. 321–330. ISSN: 0010-4485. DOI: 10.1016/s0010-4485(00)00091-9. URL: http://dx.doi.org/10.1016/S0010-4485(00)00091-9.

[7] Y. L. Ma, W. Hewitt. "Point inversion and projection for NURBS curve and surface: Control polygon approach". In: *Computer Aided Geometric Design* 20.2 (May 2003), pp. 79–99. ISSN: 0167-8396. DOI: 10.1016/s0167-8396(03)00021-9. URL: http://dx.doi.org/10.1016/S0167-8396(03)00021-9.

[8] Y.-T. Oh, Y.-J. Kim, J. Lee, M.-S. Kim, G. Elber. "Efficient point-projection to freeform curves and surfaces". In: *Computer Aided Geometric Design* 29.5 (June 2012), pp. 242–254. ISSN: 0167-8396. DOI: 10.1016/j.cagd.2011.04.002. URL: http://dx.doi.org/10.1016/j.cagd.2011.04.002.

[9] L. Piegl, W. Tiller. *The NURBS Book*. Springer Berlin Heidelberg, 1997. ISBN: 9783642592232. DOI: 10.1007/978-3-642-59223-2. URL: http://dx.doi.org/10.1007/978-3-642-59223-2.

[10] T. Saito, G.-J. Wang, T. W. Sederberg. "Hodographs and normals of rational curves and surfaces". In: *Computer Aided Geometric Design* 12.4 (June 1995), pp. 417–430. ISSN: 0167-8396. DOI: 10.1016/0167-8396(94)00023-l. URL: http://dx.doi.org/10.1016/0167-8396(94)00023-L.

[11] I. Selimovic. "Improved algorithms for the projection of points on NURBS curves and surfaces". In: *Computer Aided Geometric Design* 23.5 (July 2006), pp. 439–445. ISSN: 0167-8396. DOI: 10.1016/j.cagd.2006.01.007. URL: http://dx.doi.org/10.1016/j.cagd.2006.01.007.

B PUMA's new building blocks

In this final chapter I will gather certain core aspects of the current PUMA implementation of which I have been mainly (or even fully) responsible, and that without my intervention would likely not exist in the library (and would perhaps not have ever existed).

B.1 Product and quotient rules

The distinctive property of the PUM is the construction of discrete function spaces as

$$V^{\mathrm{PU}} = \sum_{i=0}^{n} \varphi_i V_i,$$

which in particular means that basis functions arise from a combination of the PU functions φ_i and the basis functions of each local space $V_i = \operatorname{span}\langle \vartheta_i^k \rangle_{k=1}^{m_i}$ as (recalling (2.3))

$$\left\{ \varphi_i \vartheta_i^k : \; i = 0, \dots, n; \; k = 1, \dots, m_i \right\}.$$

This means that, for computing derivatives of the basis functions, we have to combine those of the φ_i and those of the ϑ_i^k through product rules. Additionally, the PU functions themselves are actually computed from so-called *Shepard weights* W_i as

$$\varphi_i(\mathbf{x}) = \frac{W_i(\mathbf{x})}{\sum_{j \in \mathcal{N}_i} W_j(\mathbf{x})}, \quad \mathcal{N}_i \coloneqq \{ j : \; \operatorname{supp}(W_j) \cap \operatorname{supp}(W_i) \neq \emptyset \}$$

(see e.g. [1]), which means that the derivatives of each φ_i have to be computed from those of W_i and the sum of its neighboring W_j through quotient rules.

Remark B.1. Currently, PUMA is capable of handling derivatives of arbitrary order. In more specific detail, a maximal value has to be fixed at compile time, and certain compilation bounds of the current implementation set a hard limit of 10 (which we are nevertheless not interested in reaching). The implementation was a joint work of Jannik Michels and me, where before only first and second order derivatives were allowed, and it revealed the need for a flexible and generic implementation of product and quotient rules.

Our implementation of product rules is based on the following analysis. Given two scalar functions $f : \mathbb{R}^d \to \mathbb{R}$, $g : \mathbb{R}^d \to \mathbb{R}$ and their product $p = fg$, the product rule for first-order derivatives is

$$\partial_i p = [g \, \partial_i f] + [f \, \partial_i g],$$

for second-order derivatives,

$$\partial_{ij}p = [g\,\partial_{ij}f] + [\partial_i f\,\partial_j g + \partial_j f\,\partial_i g] + [f\,\partial_{ij}g],$$

for those of third order,

$$\partial_{ijk}p = [g\,\partial_{ijk}f] + [\partial_{ij}f\,\partial_k g + \partial_{ik}f\,\partial_j g + \partial_{jk}f\,\partial_i g] + [\partial_{ij}g\,\partial_k f + \partial_{ik}g\,\partial_j f + \partial_{jk}g\,\partial_i f] + [f\,\partial_{ijk}g],$$

and so on, where $i, j, k \in \{1, \ldots, d\}$. Note that the ordering of the indices is important: every sub-tuple of indices that is transferred to derivatives of either f or g must respect the ordering of the original index tuple for p. We may understand each bracketed term as a "symmetric outer product" that we denote by the symbol $\oplus$, i.e.

$$\left(D^1 f \oplus D^1 g\right)_{ij} = \partial_i f\,\partial_j g + \partial_j f\,\partial_i g$$

would be the symmetric outer product of the gradients of f and g, and

$$\left(D^2 f \oplus D^1 g\right)_{ijk} = \partial_{ij}f\,\partial_k g + \partial_{ik}f\,\partial_j g + \partial_{jk}f\,\partial_i g$$

would be that of the hessian of f and the gradient of g. For a general definition, let $P(n, m)$ be the set of all possible partitions of $\{1, \ldots, n+m\}$ into 2 subsets of sizes n and m. We then define the symmetric outer product of two tensors $\mathbb{A}$ and $\mathbb{B}$ of order n and m, respectively (with size d in all dimensions), as the tensor of order $n+m$ (also with size d in all dimensions) with entries

$$\left(\mathbb{A} \oplus \mathbb{B}\right)_\alpha = \sum_{(S,T)\in P(n,m)} \mathbb{A}_{\alpha[S]}\,\mathbb{B}_{\alpha[T]}, \quad \alpha = (\alpha_1, \ldots, \alpha_{n+m}) \in \{1, \ldots, d\}^{n+m},$$

where by $\alpha[S]$ we understand the n-array formed by the entries of α with indices in S (ordered by increasing index), and analogously for $\alpha[T]$. Note that $|P(n, m)| = \binom{n+m}{n}$. In these terms, we can write the k-th order derivatives of fg as

$$D^k(fg) = \sum_{i=0}^{k} \left(D^i f \oplus D^{k-i} g\right).$$

To derive the quotient rule, let us consider f/g with $g > 0$. Applying the product rule to $f = g(f/g)$ yields

$$D^k f = \sum_{i=0}^{k} \left[D^i g \oplus D^{k-i}\left(\frac{f}{g}\right)\right] = g\,D^k\left(\frac{f}{g}\right) + \sum_{i=1}^{k} \left[D^i g \oplus D^{k-i}\left(\frac{f}{g}\right)\right],$$

from which it follows that

$$D^k\left(\frac{f}{g}\right) = \frac{1}{g}\left[D^k f - \sum_{i=1}^{k} D^i g \oplus D^{k-i}\left(\frac{f}{g}\right)\right].$$

Note that the computation of each new derivative of f/g relies on all its lower-order derivatives, which was not the case for the product rule.

B.2 Vectorized integration

For many years after its original conception, the main bottleneck of any computation made with PUMA was the integration of multilinear forms into vectors and sparse matrices (or even distributed third order tensors, if necessary). Because of the role that numerical integration plays in any software library dedicated to the discretization and solution of PDEs with the Galerkin method, this was its main drawback when compared to other available simulation software. This situation came to an end in late 2024, when I finally managed to refactor our whole integration pipeline, leaving behind a per-quadrature-point process for a new approach in which, for each cell, the critical operations are performed on all its quadrature points at once, through innermost loops that are usually vectorized by the compiler. This is what we have termed *vectorized integration*, an idea that Albert Ziegenhagel and Lukas Troska had already had for a while, but which I carried to completion, and which I consider my main contribution to the PUMA library.

With the old point-wise approach, the integration process over a given set of cells for the Poisson bilinear form $\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega$ for a 2-dimensional domain Ω would roughly look like the following:

---

for cell **in** integration cells **do**
  Generate quadrature points.
  for quadrature point (q, w) **in** quadrature points **do**
    Evaluate all basis functions supported at the cell at location q.

    for matrix block A_{ij} **in** blocks affected by this cell **do**
      for local basis index $\ell \in \{1, \ldots, m_j\}$ **do**
        Let $B^{(u)} = \nabla(\varphi_j \vartheta_\ell^j)$ be the gradient of the corresponding trial basis function.
        for local basis index $k \in \{1, \ldots, m_i\}$ **do**
          Let $B^{(v)} = \nabla(\varphi_i \vartheta_k^i)$ be the gradient of the corresponding test basis function.

          Add weighted point-wise contribution into the destination, i.e.
            $(A_{ij})_{k\ell}$ += $w \left(B_0^{(u)} B_0^{(v)} + B_1^{(u)} B_1^{(v)} \right)$

In the vectorized setting, however, it looks like this:

---

for cell **in** integration cells **do**
  Generate quadrature points.
  Evaluate all basis functions supported at the cell *at all quadrature points*.

  for matrix block A_{ij} **in** blocks affected by this cell **do**
    for local basis index $\ell \in \{1, \ldots, m_j\}$ **do**
      Let $B^{(u)} = \nabla(\varphi_j \vartheta_\ell^j)$ holding values *for all quadrature points*.
      for local basis index $k \in \{1, \ldots, m_i\}$ **do**
        Let $B^{(v)} = \nabla(\varphi_i \vartheta_k^i)$ also holding values *for all quadrature points*.

        for quadrature point (q, w) **in** quadrature points **do**
          Add weighted contribution into the destination, i.e.
            $(A_{ij})_{k\ell}$ += $w \left(B_0^{(u)}[q] B_0^{(v)}[q] + B_1^{(u)}[q] B_1^{(v)}[q] \right)$

In particular, this means that the structures used to hold basis values ($B^{(u)}$ and $B^{(v)}$ for the gradients in our pseudocode algorithms above) now have to hold results for all quadrature points. For the case of 2-dimensional gradients, in our old approach we would use

```
xymath::FixedVector<Scalar, 2>,
```

where `xymath::FixedVector<T, D>` is a vector of size `D` (known at compile time) holding entries of type `T`, which are simply updated at every new quadrature point. In the new approach, those objects have been replaced by

```
xymath::FixedVector<xymath::AlignedVector<Scalar>, 2>,
```

i.e. their two components are no longer scalar values, but vectors of them, since `xymath::AlignedVector<T>` is simply a `std::vector<T, Allocator>`, with a specific allocator designed to perform aligned memory allocation, optimizing for memory accesses within vectorized for-loops. This is not strictly necessary (a simple `std::vector<Scalar>` would do the work), but it yields slight performance benefits.

For the remaining derivatives, entries of type `Scalar` have analogously been replaced by `xymath::AlignedVector<Scalar>`, i.e. second order derivatives are stored in objects of type

```
xymath::FixedMatrix<xymath::AlignedVector<Scalar>, D, D>,
```

and higher-order derivatives in

```
xymath::FixedTensor<xymath::AlignedVector<Scalar>, Sizes<D, D, D, ...>>.
```

The new implementation allows the innermost for-loop over a cell's quadrature points (a reduction which contains trivial algebraic operations like additions and products) to be vectorized by the compiler, which allows the processor to simultaneously perform multiple steps (up to 8, depending on the CPU). The operations involved in the basis evaluation may now also be embedded into vectorized for-loops (as long as they are based on simple algebraic operations and contain no branching).

Of course, the difficulty lied not in coming up with this idea, but in refactoring all the involved code, not only in the `paunt` library (where it was mostly tedious) but mainly in the `geco` library, which we use to automatically generate C++ code for forms, tensor fields and enrichment spaces.

Multiple other changes incorporated to PUMA in the meantime (particularly affecting the generation of integration cells) prevent a direct comparison between the non-vectorized and the vectorized integration approaches, but the reduction in integration times should be driven mainly by the vectorized approach. In Table B.1, we share the integration times of the Poisson bilinear form in the unit square domain $\Omega = (0, 1)^2$, with PUM spaces based on homogeneous refinement levels $\ell \in \{7, 8\}$ (i.e. with $2^{2\ell}$ patches) and local polynomial spaces of degree $p \in \{4, 5\}$. The choice of large polynomial degrees forces the need for more quadrature points per cell and thus reveals the benefits of the new approach even more clearly. In this specific case and for the few considered parameter combinations, we observe a reduction factor larger than 2.5.

Case		Time(s)	
ℓ	p	old	new
7	4	2.9	1.1
7	5	6.1	2.4
8	4	11.5	4.5
8	5	24.3	8.7

Table B.1: Integration times for the Poisson bilinear form $\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega$ in $\Omega = (0,1)^2$ for PUM spaces of refinement level ℓ and local polynomial spaces of degree p, with the old (non-vectorized) and the new (vectorized) approaches for integration. A single thread on a single process is used for integration.

B.3 Pre-compiled tensor fields

The functionality I will refer to here has been available in PUMA for a long time, just not properly exploited. Multilinear forms may of course contain terms independent of the basis functions, e.g. $\int_\Omega \kappa \nabla u \cdot \nabla v \, \mathrm{d}\Omega$ with some $\kappa \colon \Omega \subset \mathbb{R}^2 \to \mathbb{R}$. In this case, the actual definition of κ may be passed directly to the form evaluation code, for example for the choice $\kappa(\mathbf{x}) \coloneqq 1 + \|\mathbf{x}\|^2$ we would have

---

for local basis index $\ell \in \{1, \dots, m_j\}$ **do**
  Let $B^{(u)} = \nabla(\varphi_j \vartheta_\ell^j)$ holding values *for all quadrature points*.

  for local basis index $k \in \{1, \dots, m_i\}$ **do**
    Let $B^{(v)} = \nabla(\varphi_i \vartheta_k^i)$ also holding values *for all quadrature points*.

    for quadrature point (q, w) **in** quadrature points **do**
      $(A_{ij})_{k\ell}$ += $w \left(1 + \|q\|^2\right) \left(B_0^{(u)}[q] B_0^{(v)}[q] + B_1^{(u)}[q] B_1^{(v)}[q]\right)$

Alternatively, we may embed κ into a `TensorField<Point2D, Scalar>`, which can be evaluated (for all quadrature points) prior to the loops over basis indices, i.e.

Let K be the vector of evaluations of κ at all given quadrature points.

for local basis index $\ell \in \{1, \dots, m_j\}$ **do**
  Let $B^{(u)} = \nabla(\varphi_j \vartheta_\ell^j)$ holding values *for all quadrature points*.

  for local basis index $k \in \{1, \dots, m_i\}$ **do**
    Let $B^{(v)} = \nabla(\varphi_i \vartheta_k^i)$ also holding values *for all quadrature points*.

    for quadrature point (q, w) **in** quadrature points **do**
      $(A_{ij})_{k\ell}$ += $wK[q] \left(B_0^{(u)}[q] B_0^{(v)}[q] + B_1^{(u)}[q] B_1^{(v)}[q]\right)$

The second option leads to a more efficient form integration, since the evaluation of κ does not have to be repeated for each pair of basis indices. Additionally, the form evaluation code does not have to know the specific shape of κ, simply that it is some scalar field $\kappa \colon \Omega \subset \mathbb{R}^2 \to \mathbb{R}$. This makes it

easier to generate and compile the form evaluation code, which we do with our `geco` library. This library is also used to generate the code for evaluating the scalar field κ itself. In more complicated scenarios, with multiple tensor fields involving complicated operations and combinations with the form (e.g. all quantities related to the shell parametrization in the Kirchhoff-Love shell model), this amounts to a significant improvement in both compilation and integration times. Notably, we can embed a tensor field not only within a form, but also within another tensor field.

In addition to all that, the evaluation results of tensor fields are cached with respect to the most recent set of quadrature points, which means that, when a further evaluation is requested for the same set of quadrature points, we can easily detect it and return the results of the first evaluation, avoiding any recomputation. This allows us to avoid reevaluating tensor fields when visiting the different blocks of a vector or matrix affected by a certain integration cell, and also when the tensor field appears in multiple different forms (to be integrated in the same loop over cells). The performance gains associated to this cached evaluation are also significant, and it took us several years after the original introduction of `TensorField<PointType, ValueType>` to come up with the idea. In this case, Lukas Troska is responsible for it.

To visualize the impact of precompiling certain form components, we may focus our attention on the volume term from the bilinear form of the Kirchoff-Love shell model from (3.14)

$$\int_\Omega A(\mathbf{u}) : \alpha(\mathbf{v}) \, \mathrm{d}\Omega + \int_\Omega B(\mathbf{u}) : \beta(\mathbf{v}) \, \mathrm{d}\Omega. \tag{B.1}$$

In this case, the surface normal $\mathbf{a}_3$, the contravariant basis vectors $\mathbf{a}^1$ and $\mathbf{a}^2$, the factor $\|\mathbf{a}_1 \times \mathbf{a}_2\|$ for integration over the parameter domain, and the in-plane projection $P = \mathbb{I} - \mathbf{a}_3 \otimes \mathbf{a}_3$ lend themselves naturally to be precompiled into separate tensor fields. We do not mention the parametrization itself and the covariant basis vectors because their evaluation is directly implemented as a tensor field, relying on our `xygeo` library. For completeness, we share a schematic version of our `python` class for a shell parametrization (with precompiled tensor fields) in Listing B.1, with the class for an isotropic Kirchhoff-Love shell model in Listing B.2. Both of them are simplified versions of the actual code, which has been developed over the years by Jannik Michels and me.

For a cylindrical shell with homogeneously refined parameter space $(0, 1)^2$, considering refinement levels $\ell \in \{5, 6\}$ and local polynomial spaces of degree $p \in \{2, 3\}$, our results, gathered in Table B.2, reveal a 25% reduction in integration times with precompiled tensor fields.

Case		Time(s)	
ℓ	p	no-prec	prec
5	2	1.9	1.5
5	3	6.0	4.5
6	2	7.8	6.0
6	3	24.8	18.3

Table B.2: Integration times for the volume terms from the bilinear form of the Kirchhoff-Love shell model (B.1), for a cylindrical shell parametrization and PUM spaces of refinement level ℓ and local polynomial spaces of degree p, either with precompiled tensor fields for certain physical quantities (prec) or without them (no-prec). A single thread on a single process is used for integration.

```
class ShellParametrization:

  def __init__(self, physical_map):
    J = grad(physical_map)

    x = Position()

    # The columns of Q are the contravariant basis vectors
    self._Q = J * inv(J.T * J)
    self._Q = PrecompiledTensorFieldFunction(self._Q, 2)(x)

    n_S = cross(J[:,0], J[:,1])

    self._surface_factor = sqrt(inner(n_S, n_S))
    self._surface_factor = PrecompiledTensorFieldFunction(self._surface_factor, 2)(x)

    self._n = normalized(n_S)
    self._n = PrecompiledTensorFieldFunction(self._n, 2)(x)

    self._P = self._Q * J.T
    self._P = PrecompiledTensorFieldFunction(self._P, 2)(x)

  def surface_factor(self):
    return self._surface_factor

  def surface_normal(self):
    return self._n

  def orthogonal_projection(self):
    return self._P

  def Q(self):
    return self._Q

  def dir_grad(self, f):
    if f.shape().is_scalar():
      return self._Q * grad(f)

    elif f.shape().is_vector():
      return grad(f) * self._Q.T

    assert f.shape().is_matrix()

    # grad(f)'s 3rd (last) dimension is contracted with self._Q's 2nd (and also last) one
    return tensordot(grad(f), self._Q, [[2], [1]])
```

Listing B.1: A peek into the `python` class used to represent shell parametrizations in PUMA.

```python
class IsotropicKirchhoffLoveShellModel:

  def __init__(self, shell_parametrization: ShellParametrization, **parameters):
    self._pm = shell_parametrization

    self._thickness = parameters["thickness"]

    self._E = parameters["E"]
    self._nu = parameters["nu"]

  def membrane_strain(self, u):
    return sym(self._pm.orthogonal_projection() * self._pm.dir_grad(u))

  def bending_strain(self, u):
    n_S = self._pm.surface_normal()

    return -sym(tensordot(n_S, self._pm.dir_grad(self._pm.dir_grad(u)), [[0], [0]]))

  def bending_stress(self, u):
    return ((self._thickness**3) / 12.) * self.apply_stiffness_tensor(self.bending_strain(u))

  def membrane_stress(self, u):
    return self._thickness * self.apply_stiffness_tensor(self.membrane_strain(u))

  def second_variation_of_strain_energy(self, u, v):
    # Bilinear form with test function v and trial function u
    svse = inner(self.membrane_stress(u), self.membrane_strain(v)) \
        + inner(self.bending_stress(u), self.bending_strain(v))

    return svse * self._pm.surface_factor() * dx

  def apply_stiffness_tensor(self, oper):
    P = self._pm.orthogonal_projection()

    D = self._E /(1. - self._nu**2)

    return D * ((1. - self._nu) * oper + self._nu * tr(oper) * P)
```

Listing B.2: A peek into the python class used to represent the isotropic Kirchhoff-Love shell model in PUMA.

B.4 Function space views

Finally, the multiple ways in which basis functions have to be manipulated to properly integrate the forms appearing in this thesis prompted the implementation of multiple new derived classes of `FunctionSpace<PointType>` in `paunt`. Such is the case of

- `ElementAggregationFunctionSpace<PointType>`,
- `BoundaryRestrictedFunctionSpace<PointType>`,
- `SubDomainRestrictedFunctionSpace<PointType>`,
- `MappedBoundaryFunctionSpace<PointType>`,
- `PeriodicFunctionSpace<PointType>`,

together with the improvement of the already available

- `EmbeddedFunctionSpace<SourcePointType, TargetPointType>`.

Their shared characteristic is that all of them rely on some underlying function space to perform the actual evaluations, with an outer layer of manipulations involving the element supports and/or the basis values.

The core function space in `paunt` is `PumSpace<PointType>`, and an element aggregation space may be wrapped around a PUM space to perform element aggregation as described in [1, Section 2.3]. Boundary-restricted function spaces (or their subdomain counterparts) are used for Nitsche stabilization functions, while mapped-boundary function spaces allow for shell coupling (and solid-shell blending) by transforming the supports of basis functions and evaluation locations between the parameter domains of the involved shells (see Figure 3.2 and Figure 3.3). For solid-shell blending, embedded function spaces allow for the treatment of the shell function space (which is inherently two-dimensional) as a three-dimensional space by an extruded view of its parameter domain. Finally, periodic function spaces can be used to perform shell coupling at different boundary parts of the same parameter domain, e.g. for cylindrical or toroidal geometries.

Needless to say, not only a `PumSpace<PointType>` can be embedded into one of this function spaces views, but views can also be embedded into one another, producing some *matrioska*-like structures, e.g. an `EmbeddedFunctionSpace<Point2D, Point3D>` may be wrapped around a `MappedBoundaryFunctionSpace<Point2D>`, itself wrapped around some `ElementAggregationFunctionSpace<Point2D>`, finally wrapped around a `PumSpace<Point2D>`.

References

[1] P. Jiménez Recio, M. A. Schweitzer. “A Partition of Unity construction of the stabilization function in Nitsche’s method for variational problems”. In: *Computer Methods in Applied Mechanics and Engineering* 426 (June 2024), p. 117002. ISSN: 0045-7825. DOI: 10.1016/j.cma.2024.117002. URL: http://dx.doi.org/10.1016/j.cma.2024.117002.

C Attached papers

This final appendix includes the two papers [1] and [2] written in collaboration with my advisor. The first one, attached in Appendix C.1, was briefly described in Section 3.5, and the second one, attached in Appendix C.2, was summarized in Chapter 5. Since the latter has not yet been officially published at the time of printing this thesis, we provide here a pre-print version of it.

While the developments for both works have been carried out exclusively by me, my advisor contributed careful reviews and suggestions for the numerical experiments. I particularly appreciate the suggestion that led to [2, Example 5.2], an experiment which very neatly shows how the sparsity pattern of the FSAI preconditioner adapts to an anisotropic setting.

We considered the possibility of making a third paper out of Appendix A in case somebody would find it helpful, but since computational geometry is not our field of expertise, we decided to keep it as part of this thesis.

References

[1] P. Jiménez Recio, M. A. Schweitzer. "A Partition of Unity construction of the stabilization function in Nitsche's method for variational problems". In: *Computer Methods in Applied Mechanics and Engineering* 426 (June 2024), p. 117002. ISSN: 0045-7825. DOI: 10.1016/j.cma.2024.117002. URL: http://dx.doi.org/10.1016/j.cma.2024.117002.

[2] P. Jiménez Recio, M. A. Schweitzer. *Chebyshev smoothing with adaptive block-FSAI preconditioners for the multilevel solution of higher-order problems*. 2025. arXiv: 2509.06744 [math.NA]. URL: https://arxiv.org/abs/2509.06744v2.

Computer Methods in Applied Mechanics and Engineering 426 (2024) 117002

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

A Partition of Unity construction of the stabilization function in Nitsche's method for variational problems

Pablo Jiménez Recio *, Marc Alexander Schweitzer

Institut für Numerische Simulation, Universität Bonn, Friedrich-Hirzebruch-Allee 7, 53115, Bonn, Germany
Fraunhofer Institute for Algorithms and Scientific Computing, Schloss Birlinghoven, 53757, Sankt Augustin, Germany

ARTICLE INFO

Keywords:
Meshfree method
Partition of unity method
Essential boundary conditions
Nitsche's method

ABSTRACT

In this paper we develop a partition-of-unity construction of the stabilization function required in Nitsche's method, which can be seen as a generalization of the element-wise construction that is widely used in finite element methods. This allows for the use of Nitsche's method within the Partition of Unity Method with a stabilization function that is not simply a constant over the whole boundary. In addition to that, we introduce a patch-aggregation approach designed to avoid arbitrarily large values of the stabilization function and the associated ill-conditioned systems and deteriorated convergence rates. We present numerical results to validate the proposed methods, covering Dirichlet boundary conditions, interface constraints and higher-order problems. These results clearly show that our approach leads to optimal convergence rates.

1. Introduction

A critical though sometimes overlooked component in the numerical discretization of PDEs is the translation of essential boundary or interface conditions from the strong or the weak formulation to the discrete setting. The nodal assignment of Dirichlet boundary conditions in the standard Finite Element Method (FEM) constitutes the simplest approach, but that stems from the direct availability of nodal basis functions with the Kronecker-delta property in the FEM, and from the simplicity of the boundary condition itself.

However, it is not straightforward to generalize that simple approach for discrete function spaces that lack a nodal basis, at least at the boundary (e.g. some meshfree methods), or in the context of more complicated boundary conditions, such as those from higher-order problems (e.g. shell problems), those associated to complicated curved boundaries, or interface constraints (e.g. contact with or without friction). Furthermore, these two cases are not independent: usually, those problems that give rise to more complicated boundary conditions can typically be better approximated with modern discretization methods that lack a nodal basis at the boundary, such as the Partition of Unity Method (PUM) [1,2].

One technique to impose essential constraints without the need for a nodal basis at the boundary is Nitsche's method, first introduced in [3]. This method has received significant attention in recent years, including the abstract framework formulated by Benzaken et al. [4], as well as a rigorous derivation of the method for Kirchhoff–Love plates and shells [5], an exploration of the stabilization function within unfitted FEM discretizations by de Prenter et al. [6], and its extension for contact problems by Chouly et al. [7], among others.

In the context of the PUM, the usual approach to impose Dirichlet boundary conditions has been the algebraic technique described by Schweitzer [8], based on splitting each local space into two subspaces: one to impose the boundary conditions, and the other one

* Corresponding author at: Institut für Numerische Simulation, Universität Bonn, Friedrich-Hirzebruch-Allee 7, 53115, Bonn, Germany.
E-mail addresses: pablo.jimenez.recio@scai.fraunhofer.de (P. Jiménez Recio), marc.alexander.schweitzer@scai.fraunhofer.de (M.A. Schweitzer).

https://doi.org/10.1016/j.cma.2024.117002
Received 15 February 2024; Received in revised form 16 April 2024; Accepted 16 April 2024
Available online 25 April 2024

to solve the PDE (which can be seen essentially as the limit case of Nitsche's method for an arbitrarily large stabilization function). Nonetheless, Nitsche's method with finite stabilization had already been used earlier by Griebel and Schweitzer [9], albeit simply with a constant stabilization function. Due to its simplicity, this was also the original approach for Nitsche's method within the FEM, where nevertheless it is now common to construct a stabilization function that is constant only on every element [10,11]. This allows to adapt such function to the local properties both of the problem and of the discrete function space. In this work, we describe the construction of a stabilization function which generalizes this FEM approach, and which can be used within the PUM context. In addition to that, we present a patch-aggregation technique designed to avoid arbitrarily large values of the stabilization function and ill-conditioned systems, an issue which is analogous to the one presented in [6].

The paper is organized as follows. In Section 2, we introduce the basic notions of the PUM required for the subsequent arguments, including the construction of a so-called *cover* of the domain, and the patch-aggregation technique. In Section 3, we describe Nitsche's method in its most basic setting: the Poisson problem with Dirichlet boundary conditions only. We rely on this setting to present our partition-of-unity construction of the stabilization function, which can nevertheless be generalized to any other Nitsche formulation. In Section 4, we illustrate the use of Nitsche's method in the PUM with several numerical examples considering a wide range of applications: from second order interface problems, over shell problems, to sixth-order PDEs. We conclude the paper with a brief discussion in Section 5.

2. The Partition of Unity Method

In this section, we briefly introduce the Flat-Top Partition of Unity Method (FT-PUM) by Griebel and Schweitzer [1,2], focusing on those aspects that will be relevant in subsequent sections. All the functionalities that we describe here are implemented within the PUMA software framework developed at Fraunhofer SCAI.[1]

2.1. Cover construction

Given some open Lipschitz domain $\Omega \subset \mathbb{R}^d$, the first step will be to construct an *open cover* of it

$$\mathcal{C}_\Omega = \{\omega_i\}_{i=1}^N,$$

consisting of open sets ω_i (called *patches*) such that $\bar{\Omega} \subset \bigcup_{i=1}^N \omega_i$. In what follows, we describe a simple procedure for the construction of open covers that we will use in this work. For a detailed description of a more general procedure, we refer to [12].

First, we take a d-dimensional bounding box of the domain

$$Q := \prod_{j=1}^{d} (a_j - r,\, a_j + r) \quad \text{with } r > 0, \quad \Omega \subset Q,$$

a_j being the coordinates of its center. Now, for $k \in \{0\} \cup \mathbb{N}$, let $\mathcal{R}_Q$ be the uniformly-refined structured decomposition of Q which arises from splitting each interval $(a_j - r,\, a_j + r)$ into 2^k elements of equal length. We will refer to k as the *discretization level*. This set is then composed of $M_k = 2^{kd}$ d-dimensional boxes whose sides are of length $2h = 2^{1-k}r$, i.e.

$$\mathcal{R}_Q := \{R_i\}_{i=1}^{M_k}, \quad R_i := \prod_{l=1}^{d} \left((x_i)_l - h, (x_i)_l + h\right),$$

where $x_i \in \mathbb{R}^d$ is the center of the d-dimensional box R_i (with coordinates $(x_i)_l$), as shown in Fig. 1. Note that we can also construct a locally-refined structured decomposition of Q, as shown in Fig. 2. For simplicity we will stick to uniform refinement and will come back to the general case later. We may ignore those boxes not intersecting the domain, leading to the definition of

$$\mathcal{R}_\Omega := \left\{ R \in \mathcal{R}_Q \,:\, R \cap \Omega \neq \emptyset \right\}.$$

At this point, we define a set $\mathcal{C}_\Omega$ of overlapping patches ω_i by stretching the boxes $R_i \in \mathcal{R}_\Omega$ with some factor $\alpha \in (1, 2)$:

$$\mathcal{C}_\Omega := \left\{ \omega := S_\alpha(R) \,:\, R \in \mathcal{R}_\Omega \right\},$$

where the stretching operator S_α is defined as

$$\prod_{l=1}^{d} (c_l - s, c_l + s) \xmapsto{S_\alpha} \prod_{l=1}^{d} (c_l - \alpha s, c_l + \alpha s).$$

By an abuse of notation, we may write $\mathcal{C}_\Omega = \{\omega_i\}_{i=1}^N$ after a reindexing of the patches. The stretch factor α controls the size of the overlap between patches: $\alpha > 1$ implies that the overlap is non-empty, while $\alpha < 2$ guarantees that the following pointwise overlap condition holds

$$\forall x \in \Omega \,:\, \operatorname{card}\left(\left\{ i \,:\, x \in \omega_i,\, \omega_i \in \mathcal{C}_\Omega \right\}\right) \leq 2^d. \tag{1}$$

[1] https://www.scai.fraunhofer.de/en/business-research-areas/meshfree-multiscale-methods/products/puma.html

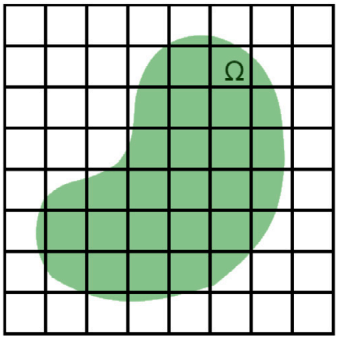


Fig. 1. Globally-refined structured decomposition of a bounding box Q of a domain Ω.

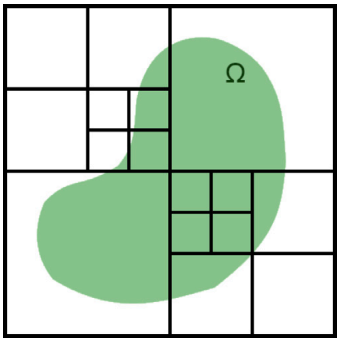


Fig. 2. Locally-refined structured decomposition of a bounding box Q of a domain Ω.

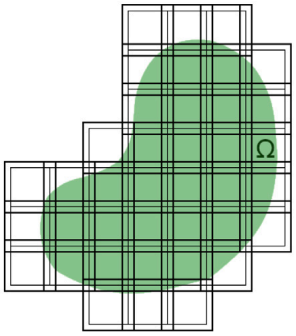


Fig. 3. Open cover of a domain Ω.

An example of such a cover can be seen in Fig. 3. In addition, we define

$$\omega_{i,\mathrm{FT}} := \omega_i \setminus \bigcup_{\substack{j=1 \\ j\neq i}}^{N} \omega_j, \tag{2}$$

which we will refer to as the *flat-top region* (hence the FT subscript) of patch ω_i, for reasons that will become clear later.

We say the *weak flat-top property* holds whenever there exists $C_{\mathrm{FT}} > 0$ such that, for any cover $\mathcal{C}_\Omega$,

$$\mu_d(\omega_{i,\mathrm{FT}}) \geq C_{\mathrm{FT}}\, \mu_d(\omega_i), \quad \forall \omega_i \in \mathcal{C}_\Omega, \tag{3}$$

where μ_d is the d-dimensional Lebesgue measure. Note that, for a cover constructed as above, it holds that

$$\frac{\mu_d(\omega_{i,\mathrm{FT}})}{\mu_d(\omega_i)} \geq \left(\frac{2-\alpha}{\alpha}\right)^d > 0, \quad \forall \omega_i \in \mathcal{C}_\Omega.$$

Let us now refer to the more general case of locally-refined covers where, beginning with level 0, each patch might be refined to the next level (i.e. split into 2^d boxes of equal size) or not. In the resulting cover, two neighboring, unstretched patches may belong to different levels of refinement. For the weak flat-top property to hold in these cases, we first need to impose a maximum level

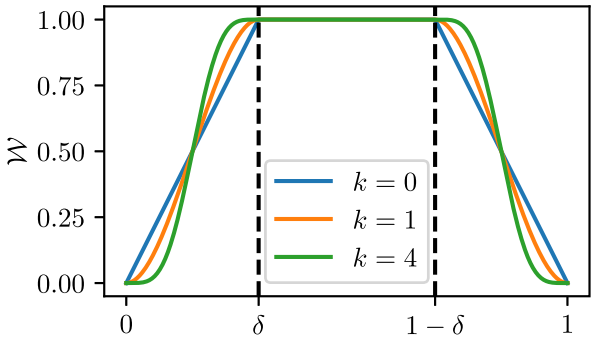


Fig. 4. Weight-generating functions $\mathcal{W}$ for different regularities k.

difference ΔL between neighboring patches, in which case it holds that

$$\frac{\mu_d(\omega_{i,\mathrm{FT}})}{\mu_d(\omega_i)} \geq \left(\frac{2-(2^{\Delta L}+1)(\alpha-1)}{2\alpha}\right)^d, \quad \forall \omega_i \in C_\Omega,$$

which can be derived by a simple 1D argument with two segments of length 1 and one of length $2^{\Delta L}$. For the right hand side of the inequality to be positive, we need the new upper bound

$$\alpha < 1 + \frac{2}{2^{\Delta L}+1},$$

which reduces to the prior $\alpha < 2$ for uniform refinement, i.e. if $\Delta L = 0$.

The weak flat-top property also prevents the stretching of patches from creating new neighborhood-relationships apart from those coming from the unstretched patches, which in particular implies that the pointwise overlap condition (1) also holds.

2.2. Construction of a partition of unity

Having a cover $C_\Omega = \{\omega_i\}_{i=1}^N$, the next step is the construction of a *partition of unity* (PU) associated to it: a finite set of Lipschitz functions $\{\varphi_i\}_{i=1}^N$ with $\varphi_i : \mathbb{R}^d \to [0,1]$, $\sum_{i=1}^N \varphi_i \equiv 1$ in $\bar{\Omega}$, and satisfying

$$\mathrm{supp}(\varphi_i) \subset \bar{\omega}_i, \quad \|\varphi_i\|_{L^\infty(\mathbb{R}^d)} \leq C_\infty, \quad \|\nabla \varphi_i\|_{L^\infty(\mathbb{R}^d)} \leq \frac{C_\nabla}{\mathrm{diam}(\omega_i)}, \quad \forall i = 1, \dots, N.$$

We will construct these PU functions by a localized version of Shepard's method, as in [2,12]. This approach amounts to taking weight functions $W_i : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ with $\mathrm{supp}(W_i) \subset \bar{\omega}_i$ and $\sum_{i=1}^N W_i > 0$ on $\bar{\Omega}$, and then setting

$$\varphi_i(x) = \frac{W_i(x)}{\sum_{j=1}^N W_j(x)} = \frac{W_i(x)}{\sum_{j \in C_i} W_j(x)},$$

where $C_i := \{j : \omega_j \cap \omega_i \neq \emptyset\}$ is the set of indices corresponding to patches in the neighborhood of ω_i. As a result, $\varphi_i \equiv 1$ on $\omega_{i,\mathrm{FT}} \subset \omega_i$, which is why we call it the *flat-top region* of ω_i (see [2]).

In particular, for our previous choice of ω_i as d-dimensional boxes, we can construct the weight functions W_i from a univariate function $\mathcal{W} : [0,1] \to \mathbb{R}_{\geq 0}$ as

$$W_i(x) = \prod_{l=1}^d \mathcal{W}\left(\frac{x_l - a_l^{(i)}}{b_l^{(i)} - a_l^{(i)}}\right), \quad x = (x_1, \dots, x_d) \in \omega_i = \prod_{l=1}^d \left(a_l^{(i)}, b_l^{(i)}\right) \tag{4}$$

and 0 for $x \notin \omega_i$. For a given regularity $k \geq 0$ and $\delta \in (0, 0.5)$, we will construct $\mathcal{W} \in C^k([0,1])$ as the B-spline function of degree $p = 2k+1$ with knots

$$(\underbrace{0, \dots, 0}_{p+1}, \underbrace{\delta, \dots, \delta}_{p}, \underbrace{1-\delta, \dots, 1-\delta}_{p}, \underbrace{1, \dots, 1}_{p+1})$$

and control points

$$(\underbrace{0, \dots, 0}_{(p+1)/2}, \underbrace{1, \dots, 1}_{2p}, \underbrace{0, \dots, 0}_{(p+1)/2}).$$

It then holds that $W_i \in C^k(\mathbb{R}^d)$, and thus $\varphi_i \in C^k(\mathbb{R}^d)$ as well. The shape of $\mathcal{W}$ for different values of k is shown in Fig. 4.

By choosing $\delta = 1-\alpha^{-1}$, α being the stretch factor of the cover, we can make the flat top region of $\mathcal{W}$ coincide with that of every interior patch of a uniformly refined cover. More generally, for an interior patch, i.e. $\omega_i \cap \partial\Omega = \emptyset$, it holds that

$$\sum_{j=1}^{N} W_j(x) = 1, \quad \forall x \in \omega_i,$$

implying that each Shepard weight coincides with the corresponding PU function at any such point.

Finally, as a special case (which we will use later), discontinuous PU functions $\varphi_i \in L^2(\Omega)$ can be generated by the simple choice $\mathcal{W} = \mathbb{1}_{(0,1)}$.

2.3. Strong flat-top property and patch aggregation

We have already stated a weak flat-top property (3), but it involved the flat-top region defined in (2), which does not account for the domain Ω. We may remediate that now: we say that the *strong* flat-top property holds if there exists $C_{\mathrm{FT}}(\Omega) > 0$ such that, for any cover $\mathcal{C}_\Omega$,

$$\frac{\mu_d(\omega_{i,\mathrm{FT}} \cap \Omega)}{\mu_d(\omega_i)} \geq C_{\mathrm{FT}}(\Omega), \quad \forall \omega_i \in \mathcal{C}_\Omega.$$

The cover construction process defined previously can only account for the weak flat-top property through the maximum level difference between neighboring patches ΔL and the stretch factor α. A rigorous enforcement of the strong property is not yet available, but some strategies can be devised to avoid the appearance of patches for which $\mu_d(\omega_{i,\mathrm{FT}} \cap \Omega)/\mu_d(\omega_i)$ is very small (or even 0).

Our procedure, which we describe in Algorithm 1, is based on the one developed by Johansson and Larson [13] in the context of unfitted FEM discretizations. For different approaches (restricted to the FEM), we refer to [14, Section 3.4] and the references therein.

Algorithm 1 Patch Aggregation

Input: A partition of unity $\{\varphi_i\}_{i\in\mathcal{I}}$ associated to a cover $\mathcal{C} = \{\omega_i\}_{i\in\mathcal{I}}$ of some open Lipschitz domain $\Omega \subset \mathbb{R}^d$, and a threshold parameter $\delta \in (0,1)$.

Output: An *aggregated* partition of unity $\{\tilde{\varphi}_i\}_{i\in\mathcal{J}}$ with $\mathcal{J} \subset \mathcal{I}$.

1: Split the index set into $\mathcal{I} = \mathcal{I}_{\mathrm{large}} \uplus \mathcal{I}_{\mathrm{small}}$ according to

$$i \in \mathcal{I}_{\mathrm{small}} \quad \Longleftrightarrow \quad \omega_i \cap \partial\Omega \neq \emptyset \quad \wedge \quad \frac{\mu_d(\omega_{i,\mathrm{FT}} \cap \Omega)}{\mu_d(\omega_i)} < \delta.$$

2: For every $i \in \mathcal{I}_{\mathrm{small}}$, let

$$\mathcal{Q}(i) := \{j \in \mathcal{I}_{\mathrm{large}} \,:\, \omega_j \cap \omega_i \cap \Omega \neq \emptyset\},$$

i.e. the neighboring indices of i in $\mathcal{I}_{\mathrm{large}}$. Whenever $\mathcal{Q}(i) \neq \emptyset$, we may define

$$\mathcal{N}(i) := \operatorname*{argmax}_{j\in\mathcal{Q}(i)} \mu_d(\omega_j \cap \omega_i \cap \Omega),$$

as the index of the patch to which ω_i will be "aggregated".

3: Finally, let

$$\mathcal{J} := \mathcal{I}_{\mathrm{large}} \cup \left\{i \in \mathcal{I}_{\mathrm{small}} \,:\, \mathcal{Q}(i) = \emptyset\right\}.$$

4: **return** $\{\tilde{\varphi}_i\}_{i\in\mathcal{J}}$ defined by

$$\tilde{\varphi}_i := \varphi_i + \sum_{\substack{j\in\mathcal{I}\setminus\mathcal{J} \\ i=\mathcal{N}(j)}} \varphi_j, \quad i \in \mathcal{J}.$$

In step 2, we may want $\{\tilde{\varphi}_i\}$ to maintain the same sparsity pattern as $\{\varphi_i\}$, which we can enforce by restricting the candidates $\mathcal{Q}(i)$ to those neighboring patches ω_j that intersect all other neighbors of ω_i. If the threshold parameter δ is small enough and the cover is sufficiently refined, $\mathcal{Q}(i)$ should be non-empty for every $i \in \mathcal{I}_{\mathrm{small}}$, in which case $\mathcal{J} = \mathcal{I}_{\mathrm{large}}$.

Note that the possibility of $\mathcal{N}(i_1) = \mathcal{N}(i_2)$ for $i_1, i_2 \in \mathcal{I}_{\mathrm{small}}$, $i_1 \neq i_2$, is perfectly allowed. For an illustration of these 2 steps, see Fig. 5.

Note further that, for a PU $\{\varphi_i\}_{i\in\mathcal{I}}$ constructed with Shepard's method, aggregating the φ_i is equivalent to aggregating the weights W_i, i.e.

$$\tilde{\varphi}_i = \frac{\tilde{W}_i}{\sum_{j\in\mathcal{J}} \tilde{W}_j}, \quad \tilde{W}_i := W_i + \sum_{\substack{j\in\mathcal{I}\setminus\mathcal{J} \\ i=\mathcal{N}(j)}} W_j, \quad i \in \mathcal{J}.$$

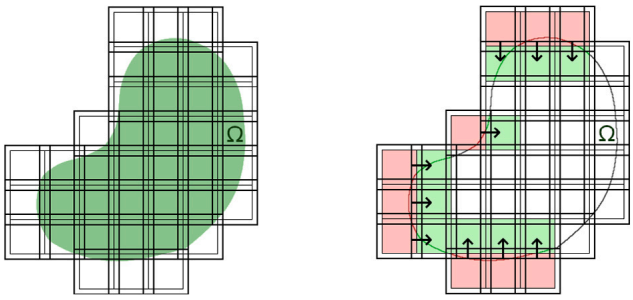


Fig. 5. Patch aggregation: patches in C_{small} are colored in red. For each of them, the corresponding $\omega_{\mathcal{N}(i)}$ is colored in green, and the respective aggregation is indicated by an arrow.

The name *patch aggregation* comes from the fact that

$$\operatorname{supp}(\tilde{\varphi}_i) = \operatorname{supp}(\varphi_i) \cup \bigcup_{\substack{j\in\mathcal{I}\setminus\mathcal{J}\\ i=\mathcal{N}(j)}} \operatorname{supp}(\varphi_j) = \omega_i \cup \bigcup_{\substack{j\in\mathcal{I}\setminus\mathcal{J}\\ i=\mathcal{N}(j)}} \omega_j.$$

2.4. PU function space

Given some PU $\{\varphi_i\}_{i=1}^N$, we can construct function spaces for a Galerkin discretization by multiplying each PU function φ_i with some local approximation space $V_i = \operatorname{span}\langle\{\vartheta_i^n\}_{n=1}^{d_i}\rangle$ defined on $\operatorname{supp}(\varphi_i)$. This yields the *Partition of Unity space*

$$V^{\text{PU}} := \sum_{i=1}^N \varphi_i V_i = \sum_{i=1}^N \varphi_i \operatorname{span}\langle\{\vartheta_i^n\}_{n=1}^{d_i}\rangle = \operatorname{span}\langle\{\varphi_i\vartheta_i^n \,:\, n=1,\dots,d_i\}_{i=1}^N\rangle.$$

A usual choice is $V_i = V^{p_i}$ the local spaces of polynomials of order p_i. However, it is possible to *enrich* the local spaces with special functions designed for each particular problem (e.g. singularities, crack or interface discontinuities), in which case we write $V_i = V^{p_i} + \mathcal{E}_i$. Since, as we have seen, PU functions can be constructed with any required regularity, and combined with local polynomial spaces of any desired degree, PU spaces can be directly used for the numerical solution of PDEs of any order.

Note, however, that one of the differences with the FEM is that the resulting PUM basis functions $\varphi_i\vartheta_i^n$ are not interpolatory in general.

3. Nitsche's method

3.1. The Poisson problem

Nitsche's method [3] can be described as a consistent penalty method for the weak imposition of essential boundary conditions in Galerkin discretizations of PDEs. To illustrate it in a simple manner, let us consider the Poisson problem with Dirichlet boundary conditions in a Lipschitz domain $\Omega \subset \mathbb{R}^d$ with $d \in \{2,3\}$,

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega,\\ u &= g \quad \text{on } \partial\Omega. \end{aligned} \tag{5}$$

Given a finite-dimensional subspace $V_N \subset H^1(\Omega)$, the original method proposed by Nitsche consists in finding $u_N \in V_N$ such that

$$\begin{aligned} \int_\Omega \nabla u_N \cdot \nabla v \mathrm{d}\Omega - \int_{\partial\Omega} \left((\nabla u_N\cdot\mathbf{n})v + (\nabla v\cdot\mathbf{n})u_N\right)\mathrm{d}\Gamma + \int_{\partial\Omega}\gamma_N u_N v\mathrm{d}\Gamma =\\ = \int_\Omega f v\mathrm{d}\Omega - \int_{\partial\Omega}(\nabla v\cdot\mathbf{n})g\mathrm{d}\Gamma + \int_{\partial\Omega}\gamma_N g v\mathrm{d}\Gamma, \quad \forall v\in V_N, \end{aligned} \tag{6}$$

with a *stabilization function* $\gamma_N \in L^2(\partial\Omega)$, $\gamma_N > 0$. After denoting

$$\begin{aligned} a_N(v,w) &:= \int_\Omega \nabla v\cdot\nabla w\mathrm{d}\Omega - \int_{\partial\Omega}((\nabla w\cdot\mathbf{n})v + (\nabla v\cdot\mathbf{n})w)\,\mathrm{d}\Gamma + \int_{\partial\Omega}\gamma_N v w\mathrm{d}\Gamma,\\ \ell_N(v) &:= \int_\Omega f v\mathrm{d}\Omega - \int_{\partial\Omega}(\nabla v\cdot\mathbf{n})g\mathrm{d}\Gamma + \int_{\partial\Omega}\gamma_N g v\mathrm{d}\Gamma, \end{aligned}$$

Problem (6) can be written as

$$u_N \in V_N \quad : \quad a_N(v,u_N) = \ell_N(v), \quad \forall v\in V_N. \tag{7}$$

Note that, if $u \in H^2(\Omega)$ is a solution to (5), then after multiplying the Poisson equation by a test function $v \in H^1(\Omega)$ and integrating by parts,

$$\int_\Omega \nabla u \cdot \nabla v \mathrm{d}\Omega - \int_{\partial\Omega} (\nabla u \cdot \mathbf{n}) v \mathrm{d}\Gamma = \int_\Omega f v \mathrm{d}\Omega,$$

which, together with the Dirichlet boundary condition, implies that

$$a_N(v,u) = \ell_N(v), \quad \forall v \in H^1(\Omega).$$

This also holds, in particular, for all $v \in V_N \subset H^1(\Omega)$, from which the method is said to be consistent. As a result, as long as Problem (7) has a solution $u_N \in V_N$, the Galerkin orthogonality property holds,

$$a_N(v, u - u_N) = 0, \quad \forall v \in V_N. \tag{8}$$

Furthermore, it is well-known that Problem (7) is equivalent to

$$\min_{w \in V_N} \left[\int_\Omega \left(\frac{1}{2} |\nabla w|^2 - f w \right) \mathrm{d}\Omega + \int_{\partial\Omega} \left(\frac{\gamma_N}{2} (w - g) - (\nabla w \cdot \mathbf{n}) \right) (w - g) \mathrm{d}\Gamma \right].$$

To study the well-posedness of Problem (7), let us first define the following *energy norm* (following [6])

$$|||v|||^2 := \|\nabla v\|^2_{L^2(\Omega)} + \left\| \gamma_N^{-\frac{1}{2}} (\nabla v \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega)} + \left\| \gamma_N^{\frac{1}{2}} v \right\|^2_{L^2(\partial\Omega)}. \tag{9}$$

With respect to this norm, we can establish V_N-coercivity and continuity of $a_N(\cdot,\cdot)$ with constants that are independent of γ_N, provided that the following trace inequality holds:

$$\exists \varrho \in (0,1) \quad : \quad \left\| \gamma_N^{-\frac{1}{2}} \nabla v \cdot \mathbf{n} \right\|^2_{L^2(\partial\Omega)} \leq \varrho^2 \|\nabla v\|^2_{L^2(\Omega)}, \quad \forall v \in V_N. \tag{10}$$

Indeed, the continuity result

$$a_N(v,u) \leq 2 |||u||| |||v||| \quad \forall u, v \in V_N \oplus H^2(\Omega)$$

holds independently of the choice of γ_N (see [6, Lemma 4]). Moreover, using Cauchy–Schwarz and Young's inequalities, for any $\epsilon > 0$

$$2 \int_{\partial\Omega} (\nabla v \cdot \mathbf{n}) v \mathrm{d}\Gamma \leq 2 \left\| \gamma_N^{-\frac{1}{2}} \nabla (v \cdot \mathbf{n}) \right\|_{L^2(\partial\Omega)} \left\| \gamma_N^{\frac{1}{2}} v \right\|_{L^2(\partial\Omega)} \leq \frac{1}{\epsilon} \left\| \gamma_N^{-\frac{1}{2}} \nabla (v \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega)} + \epsilon \left\| \gamma_N^{\frac{1}{2}} v \right\|^2_{L^2(\partial\Omega)}, \quad \forall v \in V_N, \tag{11}$$

so that

$$\begin{aligned} a_N(v,v) &= \|\nabla v\|^2_{L^2(\Omega)} - 2 \int_{\partial\Omega} (\nabla v \cdot \mathbf{n}) v \mathrm{d}\Gamma + \left\| \gamma_N^{\frac{1}{2}} v \right\|^2_{L^2(\partial\Omega)} \geq \\ &\geq \|\nabla v\|^2_{L^2(\Omega)} + (1-\epsilon) \left\| \gamma_N^{\frac{1}{2}} v \right\|^2_{L^2(\partial\Omega)} - \frac{1}{\epsilon} \left\| \gamma_N^{-\frac{1}{2}} (\nabla v \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega)} \geq \\ &\geq (1-\epsilon) \left(\|\nabla v\|^2_{L^2(\Omega)} + \left\| \gamma_N^{\frac{1}{2}} v \right\|^2_{L^2(\partial\Omega)} \right) + \left(\frac{\epsilon}{\varrho^2} - \frac{1}{\epsilon} \right) \left\| \gamma_N^{-\frac{1}{2}} (\nabla v \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega)} \end{aligned}$$

holds for all $v \in V_N$. Coercivity follows by choosing $\epsilon \in (\varrho, 1)$, in which case it additionally holds that

$$\frac{\epsilon}{\varrho^2} - \frac{1}{\epsilon} \geq \frac{\epsilon}{\varrho^2} - \frac{1}{\varrho} = \frac{\epsilon - \varrho}{\varrho^2},$$

and the particular choice $\epsilon^\star = (\varrho + \varrho^2)/(1 + \varrho^2) \in (\varrho, 1)$ leads to

$$a_N(v,v) \geq \frac{1-\varrho}{1+\varrho^2} |||v|||^2, \quad \forall v \in V_N.$$

Existence and uniqueness of a solution to Problem (7) follow then from the Lax–Milgram Theorem. Furthermore, those two results, combined with Galerkin orthogonality (8), imply Céa's Lemma: if $u \in H^2(\Omega)$ is the solution to (5) and $u_N \in V_N$ is the solution to (7), then

$$|||u - u_N||| \leq \left(1 + 2 \frac{1+\varrho^2}{1-\varrho} \right) \inf_{v \in V_N} |||u - v|||. \tag{12}$$

3.2. Constant stabilization function

If γ_N is chosen as a constant over $\partial\Omega$, then (10) reduces to $\gamma_N > C_N$, for a constant $C_N > 0$ such that

$$\|\nabla v \cdot \mathbf{n}\|^2_{L^2(\partial\Omega)} \leq C_N \|\nabla v\|^2_{L^2(\Omega)}, \quad \forall v \in V_N, \tag{13}$$

as it is well known. In this case, given some basis $\{\phi_i\}$ of the function space V_N, C_N can be determined as the largest eigenvalue of the generalized eigenvalue problem (GEP) $Ax = \lambda Bx$, with

$$A_{ij} = \int_{\partial\Omega} (\nabla\phi_i \cdot \mathbf{n})(\nabla\phi_j \cdot \mathbf{n}) \mathrm{d}\Gamma, \quad B_{ij} = \int_{\Omega} \nabla\phi_i \cdot \nabla\phi_j \mathrm{d}\Omega,$$

(as done already within the PUM context by Griebel and Schweitzer [9]). If we then set $\gamma_N = \beta C_N$ for some $\beta > 1$, (10) holds with $\varrho = \beta^{-1/2}$. A usual choice is $\beta = 2$.

3.3. A partition of unity construction of the stabilization function

A common alternative procedure in Nitsche-FEM methods is to use a Nitsche stabilization function that, instead of being globally constant, is only constant on every element, with each element value determined independently (see [10,11]). In order to generalize this approach, let $\{\psi_k\}_{k=1}^M$ be a non-negative partition of unity of $\bar{\Omega}$ with

$$\mu_d(\mathrm{supp}(\psi_k) \cap \Omega) > 0 \quad \forall k = 1, \dots, M,$$

and define $\gamma_N := 1/\eta_N$ with

$$\eta_N := \sum_{k\in\mathcal{I}} \psi_k|_{\partial\Omega}\, \eta_N^{(k)}, \quad \mathcal{I} := \left\{k \in \{1,\dots,M\} : \mu_{d-1}(\mathrm{supp}(\psi_k) \cap \partial\Omega) > 0\right\}$$

and coefficients $\eta_N^{(k)} > 0$. Generalizing (13), we will choose these $\eta_N^{(k)}$ based on constants $C_N^{(k)} > 0$ which satisfy

$$\left\| \psi_k^{\frac{1}{2}} (\nabla v \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega)} \le C_N^{(k)} \left\| \psi_k^{\frac{1}{2}} \nabla v \right\|^2_{L^2(\Omega)}, \quad \forall v \in V_N, \quad k \in \mathcal{I}. \tag{14}$$

Indeed, given such constants, we have that

$$\begin{aligned} \left\| \gamma_N^{-\frac{1}{2}} (\nabla v \cdot \mathbf{n}) \right\|^2_{\partial\Omega} &= \int_{\partial\Omega} \left(\sum_{k\in\mathcal{I}} \eta_N^{(k)} \psi_k \right) (\nabla v \cdot \mathbf{n})^2 \mathrm{d}\Gamma = \sum_{k\in\mathcal{I}} \eta_N^{(k)} \left\| \psi_k^{\frac{1}{2}} (\nabla v \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega)} \le \\ &\le \sum_{k\in\mathcal{I}} \eta_N^{(k)} C_N^{(k)} \left\| \psi_k^{\frac{1}{2}} \nabla v \right\|^2_{L^2(\Omega)} \le \max_{k\in\mathcal{I}} \left(\eta_N^{(k)} C_N^{(k)} \right) \sum_{j\in\mathcal{I}} \left\| \psi_j^{\frac{1}{2}} \nabla v \right\|^2_{L^2(\Omega)} \le \max_{k\in\mathcal{I}} \left(\eta_N^{(k)} C_N^{(k)} \right) \|\nabla v\|^2_{L^2(\Omega)}, \end{aligned}$$

so a sufficient condition for (10) to hold is $1/\eta_N^{(k)} > C_N^{(k)}$ for all $k \in \mathcal{I}$. Additionally, from the convexity of the inverse function in $\mathbb{R}_{>0}$ it follows that

$$\gamma_N = \frac{1}{\eta_N} = \frac{1}{\sum_{k\in\mathcal{I}} \psi_k|_{\partial\Omega}\, \eta_N^{(k)}} \le \sum_{k\in\mathcal{I}} \psi_k|_{\partial\Omega}\, \frac{1}{\eta_N^{(k)}},$$

and thus the alternative construction

$$\gamma_N := \sum_{k\in\mathcal{I}} \psi_k|_{\partial\Omega}\, \gamma_N^{(k)}, \quad \gamma_N^{(k)} := \frac{1}{\eta_N^{(k)}} > C_N^{(k)}$$

also satisfies (10).

In particular, we may choose $\gamma_N^{(k)} = \beta C_N^{(k)}$ for some $\beta > 1$, in which case, as before, (10) holds with $\varrho = \beta^{-1/2}$ (a usual choice is again $\beta = 2$). In this case, the GEP having $C_N^{(k)}$ as largest eigenvalue is $A^k x = \lambda B^k x$ with

$$A_{ij}^k = \int_{\partial\Omega} \psi_k (\nabla\phi_i \cdot \mathbf{n})(\nabla\phi_j \cdot \mathbf{n}) \mathrm{d}\Gamma, \quad B_{ij}^k = \int_{\Omega} \psi_k \nabla\phi_i \cdot \nabla\phi_j \mathrm{d}\Omega,$$

where $V_N = \mathrm{span}\langle\{\phi_i\}\rangle$.

Note that γ_N inherits the regularity of the PU functions ψ_k. In general, $\gamma_N(x)$ is a convex combination of the coefficients $\gamma_N^{(k)}$ associated to patches ω_k overlapping x, i.e.

$$\gamma_N(x) := \sum_{k\in\mathcal{I}} \gamma_N^{(k)} \psi_k(x), \quad x \in \partial\Omega,$$

but within any flat-top region, i.e. for $x \in \omega_{\mathrm{FT},i}$, we have that $\gamma_N(x) = \gamma_N^{(i)}$. In the particular case where the $\{\psi_k\}$ are piecewise-constant, γ_N will be piecewise-constant too.

A careful choice of the partition of unity $\{\psi_k\}$ can make these GEPs much smaller than the one previously considered, thus replacing the solution of one global GEP (only for the largest eigenvalue) by that of multiple local GEPs. In addition, by allowing the inequality (10) to be satisfied locally, the stabilization function will be better adapted to the local properties of the problem and its approximation near the boundary (e.g. the local behavior of the elliptic operator, or the intersection of $\partial\Omega$ with the cover used to construct V_N). On the contrary, a global stabilization constant forcibly adapts to those local properties that would lead to the highest value of the stabilization function in the local case, possibly yielding unnecessarily large values in other parts of the boundary.

In the FEM, given a triangulation $\mathcal{T} = \{T_k\}$ of Ω, the choice of the partition of unity associated to the triangulation $\psi_k = \chi_{T_k}$ leads to a stabilization function γ_N that is constant on every element that intersects the Dirichlet boundary, which is the case that we generalize. For every triangle T_k intersecting the boundary, the constant $C_N^{(k)} > 0$ is such that

$$\|\nabla v \cdot \mathbf{n}\|^2_{L^2(\partial\Omega\cap T_k)} \leq C_N^{(k)} \|\nabla v\|^2_{L^2(\Omega\cap T_k)}, \quad \forall v \in V_N,$$

so $C_N^{(k)}$ can be obtained as the largest eigenvalue of the GEP $A^k x = \lambda B^k x$ with

$$A^k_{ij} = \int_{\partial\Omega\cap T_k} (\nabla\phi_i \cdot \mathbf{n})(\nabla\phi_j \cdot \mathbf{n}) \mathrm{d}\Gamma, \quad B^k_{ij} = \int_{\Omega\cap T_k} \nabla\phi_i \cdot \nabla\phi_j \mathrm{d}\Omega,$$

which involves only those basis functions ϕ_i with support at T_k. For a boundary-fitted triangulation, and with the assumption of shape-regularity, the constants $C_N^{(k)}$ (or the global constant C_N) behave as $\mathcal{O}(h^{-1})$. This, together with Céa's Lemma (12), implies optimal convergence in the $H^1(\Omega)$-norm [15, Chapter 2]. However, this does not hold in general for FEM methods where the mesh is not fitted to the boundary (e.g. the fictitious domain FEM [6]).

In the PUM, for a discretization space $V_N = \sum_{i=1}^n \varphi_i V_i = \mathrm{span}\langle\{\varphi_i \vartheta_i^l\}\rangle$, the matrices for the local GEPs are

$$\begin{aligned} A^k_{(i,l),(j,m)} &= \int_{\partial\Omega} \psi_k (\nabla(\varphi_i\vartheta_i^l)\cdot\mathbf{n})(\nabla(\varphi_j\vartheta_j^m)\cdot\mathbf{n})\mathrm{d}\Gamma, \\ B^k_{(i,l),(j,m)} &= \int_{\Omega} \psi_k \nabla(\varphi_i\vartheta_i^l)\cdot\nabla(\varphi_j\vartheta_j^m)\mathrm{d}\Omega. \end{aligned} \tag{15}$$

Note that, if the PU $\{\varphi_i\}_{i=1}^n$ is associated to a cover $C_\Omega = \{\omega_i\}_{i=1}^n$, we can choose $\{\psi_k\}_{k=1}^n$ as a (non-negative) PU associated to the same cover, in which case the matrices above involve only the basis functions coming from local spaces V_l with $\mu_{d-1}(\omega_l\cap\omega_k\cap\partial\Omega) > 0$, including but not limited to V_k itself.

Compared with the FEM case, the PUM approach produces larger local GEPs, by a factor that depends on the physical dimension of the domain. For 2D domains, the number of boundary neighbors of a given boundary patch is usually 2, in which case the local GEP is 3 times larger than its FEM counterpart. For 3D domains, however, boundary neighbors span over a surface and their number is usually 8, leading to a 9 times larger GEP. Of course, there is an associated increased cost not only for solving such GEPs but also for assembling the necessary matrices. To reduce this increased computational effort in practice, we will present a cheaper procedure in one of the numerical examples of Section 4, evaluating the suitability of the resulting stabilization functions.

3.4. Generalizations

Some generalizations can be applied to the thus far introduced method for the construction of stabilization functions.

3.4.1. Non-symmetric methods

The first generalization corresponds to the non-symmetric variants of Nitsche's method, resulting from multiplying the symmetry terms in (6) with a parameter $\theta \in [-1, 1]$, leading to

$$\begin{aligned} \int_\Omega \nabla u_N \cdot \nabla v \mathrm{d}\Omega - \int_{\partial\Omega} \left((\nabla u_N\cdot\mathbf{n})v + \theta(\nabla v\cdot\mathbf{n})u_N\right)\mathrm{d}\Gamma + \int_{\partial\Omega}\gamma_N u_N v \mathrm{d}\Gamma = \\ = \int_\Omega f v \mathrm{d}\Omega - \theta\int_{\partial\Omega}(\nabla v\cdot\mathbf{n})g\mathrm{d}\Gamma + \int_{\partial\Omega}\gamma_N g v\mathrm{d}\Gamma, \quad \forall v\in V_N. \end{aligned} \tag{16}$$

It can be shown that, for $\theta \in (-1, 1]$, the sufficient condition which generalizes (10) for existence and uniqueness of solutions is

$$\exists \varrho \in (0,1) \quad : \quad \left\| \gamma_N^{-\frac{1}{2}} (\nabla v\cdot\mathbf{n}) \right\|^2_{L^2(\partial\Omega)} \leq \left(\frac{2\varrho}{1+\theta}\right)^2 \|\nabla v\|^2_{L^2(\Omega)}, \quad \forall v \in V_N,$$

so for a PU-defined Nitsche stabilization parameter we need $\gamma_N^{(k)} > \left(\frac{1+\theta}{2}\right)^2 C_N^{(k)}$, with the same trace inequalities (14). The method with $\theta = -1$ has a unique solution regardless of the value of $\gamma_N > 0$. The choice $\gamma_N = 0$, which corresponds to the so-called penalty-free Nitsche method, has been explored by Schillinger et al. within the unfitted FEM [16].

3.4.2. Interface constraints

A second generalization is to consider interface constraints, and not simply Dirichlet boundary conditions.

Let us consider two non-overlapping open Lipschitz domains $\Omega_0, \Omega_1 \subset \mathbb{R}^d$, $\Omega_0 \cap \Omega_1 = \emptyset$, sharing an interface $\Gamma = \partial\Omega_0 \cap \partial\Omega_1 \neq \emptyset$. We will impose Dirichlet boundary conditions at the rest of the boundary, denoting $\Gamma_i^D = \partial\Omega_i \setminus \Gamma$. Given two discrete spaces $V_i \subset H^1(\Omega_i)$, $i \in \{0, 1\}$, the Nitsche formulation is: find $(u_0, u_1) \in V_0 \times V_1$ such that

$$\begin{aligned} \sum_{i=0}^1 \left(\int_{\Omega_i} \nabla u_i \cdot \nabla v_i \mathrm{d}\Omega + \int_{\Gamma_i^D} \left(\gamma_i u_i v_i - (\nabla u_i\cdot\mathbf{n})v_i - (\nabla v_i\cdot\mathbf{n})u_i\right)\mathrm{d}\Gamma\right) + \\ + \int_\Gamma \left(\{\gamma\}_\lambda [\![u]\!][\![v]\!] - [\![v]\!]\{\nabla u\}_\lambda\cdot\mathbf{n}_\Gamma - [\![u]\!]\{\nabla v\}_\lambda\cdot\mathbf{n}_\Gamma\right)\mathrm{d}\Gamma = \\ = \sum_{i=0}^1 \left(\int_{\Omega_i} f_i v_i \mathrm{d}\Omega + \int_{\Gamma_i^D} g_i(\gamma_i v_i - \nabla v_i\cdot\mathbf{n})\mathrm{d}\Gamma\right), \quad \forall (v_0, v_1) \in V_0\times V_1, \end{aligned} \tag{17}$$

with the jump $[\![\cdot]\!]$ defined as $[\![w]\!] := w_0 - w_1$, and the interface normal $\mathbf{n}_\Gamma$ as pointing from Ω_0 to Ω_1 (note that it is possible to define both objects in the opposite sense at the same time, i.e. to exchange the role of indices 0 and 1). $\gamma_i \in L^2(\Gamma_i^D \cup \Gamma)$ are the stabilization functions associated to each part, and given $\lambda : \Gamma \to [0,1]$, $\lambda \in L^2(\Gamma)$, the averaging operator $\{\cdot\}_\lambda$ is defined as

$$\{w\}_\lambda := \lambda w_0 + (1-\lambda) w_1.$$

By a slight modification of (11), namely

$$2\int_\Gamma \beta_i(\nabla v \cdot \mathbf{n}_\Gamma) v \mathrm{d}\Gamma \le 2 \left\| \gamma_i^{-\frac{1}{2}} \nabla(v\cdot\mathbf{n}) \right\|_{L^2(\Gamma)} \left\| (\beta_i\gamma_i)^{\frac{1}{2}} v \right\|_{L^2(\Gamma)} \le \frac{1}{\epsilon} \left\| \gamma_i^{-\frac{1}{2}} \nabla(v\cdot\mathbf{n}) \right\|^2_{L^2(\Gamma)} + \epsilon \left\| (\beta_i\gamma_i)^{\frac{1}{2}} v \right\|^2_{L^2(\Gamma)}, \quad \forall v \in V_i$$

with $\beta_0 = \lambda$ and $\beta_1 = 1-\lambda$, it can be shown that, independently of the choice of λ, the bilinear form is coercive provided that

$$\exists \varrho \in (0,1) \quad : \quad \left\| \gamma_i^{-\frac{1}{2}} \nabla v \cdot \mathbf{n} \right\|^2_{L^2(\partial\Omega_i)} \le \varrho^2 \left\| \nabla v \right\|^2_{L^2(\Omega_i)}, \quad \forall v \in V_i, \quad i \in \{0,1\},$$

which is simply a version of (10) for each γ_i with respect to its corresponding part of the domain. The energy norm with respect to which coercivity can be shown is defined as

$$|||v|||^2 := \sum_{i=0}^{1} \left(\left\| \nabla v_i \right\|^2_{L^2(\Omega_i)} + \left\| \gamma_i^{-\frac{1}{2}} (\nabla v_i \cdot \mathbf{n}) \right\|^2_{L^2(\partial\Omega_i)} + \left\| \gamma_i^{\frac{1}{2}} v_i \right\|^2_{L^2(\Gamma_i^D)} \right) + \left\| \{\gamma\}_\lambda^{\frac{1}{2}} [\![v]\!] \right\|^2_{L^2(\Gamma)}$$

which does depend on the choice of λ. Usual choices for this function are the constant values $\lambda \in \{0,1\}$ (unbalanced), $\lambda \equiv 0.5$ (balanced), and the γ-weighted average

$$\lambda = \frac{\gamma_0^{-1}}{\gamma_0^{-1} + \gamma_1^{-1}} = \frac{1}{1 + \gamma_0/\gamma_1}, \tag{18}$$

as proposed in [17] for cases with possibly a high contrast between γ_0 and γ_1. The γ-weighted average of the stabilization functions themselves is $2\gamma_0\gamma_1/(\gamma_0+\gamma_1)$, which approximates to $2\gamma_0$ if $\gamma_1 \gg \gamma_0$, and vice versa.

Of course, the Nitsche formulation (17) can be generalized to non-homogeneous jump constraints in the function value and/or in the normal derivative, by properly adding the necessary terms to the right hand side of the equation.

3.4.3. Loaded cracks

A special case of interface is a crack in the domain of some elasticity problem. Traction-free cracks are not of interest here, because in that case Nitsche's method itself is not needed at the crack.

The method may still be used for the imposition of contact constraints involving both sides of a crack, or constraints on the displacement independently on each side, e.g. in hydraulic fracturing. If a crack splits the whole domain into 2 subdomains, we can treat it as in the previous subsection, with one function space associated to each subdomain. When that is not the case, we need to enrich some local spaces V_i: those whose support overlaps the crack front (or the crack tip in 2D) with certain special functions for the crack opening, and those completely split by the crack, with the Heaviside function H_C associated to the crack itself, in this case as

$$V_i = \mathcal{P}_i + H_C \mathcal{P}_i,$$

where H_C takes values 0 and 1 on each side of the crack, respectively.

This, however, can be also understood as splitting the $\varphi_i V_i$ contribution to the PU space into 2 as

$$\varphi_i V_i = \varphi_i(\mathcal{P}_i + H_C\mathcal{P}_i) = \varphi_i((1-H_C)\mathcal{P}_i + H_C\mathcal{P}_i) = \varphi_i^- \mathcal{P}_i + \varphi_i^+ \mathcal{P}_i,$$

where $\varphi_i^- = (1-H_C)\varphi_i$ and $\varphi_i^+ = H_C\varphi_i$ are the restrictions of φ_i to each side of the crack. In this reinterpretation, it is possible for either φ_i^+ or φ_i^- to have a very small flat-top region, in which case an analogous patch-aggregation procedure is needed, this time not for the original PU $\{\varphi_i\}$ (or for the original domain cover $\{\omega_i\}$), but for the split-patches and split-PU-functions generated by the crack. Mathematically, this is no different to the patch-aggregation performed for each PU space in the case of 2 subdomains, but in practice it presents more complications. For this reason, we have not considered cracks in our numerical examples (while still covering the interface case).

3.4.4. Different PDEs

One more generalization consists in considering different and higher-order PDEs than simply the second-order Poisson equation. As the abstract framework developed by Benzaken et al. in [4] shows, this always amounts to formulating the proper trace inequalities analogous to (10) for the problem at hand and its own kind(s) of essential boundary conditions.

For a given essential boundary condition $\mathcal{T}u = g$ (following the notation from [4]), the trace inequality will look like

$$\langle \mathcal{B}v, \gamma_N^{-1}\mathcal{B}v \rangle \le \varrho^2 a(v,v), \quad \forall v \in V_N, \quad \varrho \in (0,1), \tag{19}$$

where $a(\cdot,\cdot)$ is the quadratic form of the corresponding minimization problem (i.e. the bilinear form of the weak formulation without Nitsche's method), $\mathcal{B}$ encodes the natural boundary condition conjugate to $\mathcal{T}$, and $\langle\cdot,\cdot\rangle$ is some inner product (in the abstract setting, a

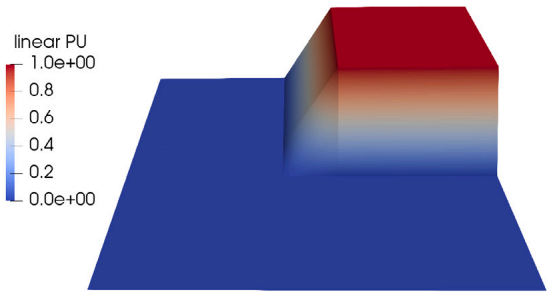


Fig. 6. PU function constructed from piecewise-linear C^0 weights.

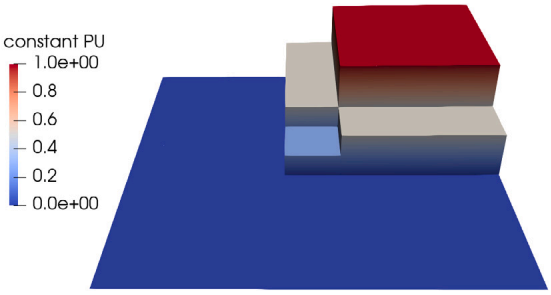


Fig. 7. PU function constructed from piecewise-constant discontinuous weights.

duality pairing, see [4, Remark 10]). Note that the factor ϱ, which we include in the trace inequality, encodes the same information as the inverse of $\gamma \in (1, \infty)$ in the generalized Cauchy–Schwarz inequality from [4, Assumption 2]. In the presence of multiple overlapping essential boundary conditions $\mathcal{T}_i$, a different stabilization function $\gamma_{N,i}$ might be used for each of them, leading to

$$\sum_{i=1}^{K} \langle \mathcal{B}_i v, \gamma_{N,i}^{-1} \mathcal{B}_i v \rangle_i \leq \varrho^2 a(v, v),$$

which we can reduce to

$$\langle \mathcal{B}_i v, \gamma_{N,i}^{-1} \mathcal{B}_i v \rangle_i \leq \frac{1}{K} \varrho^2 a(v, v), \quad i = 1, \dots, K$$

(or with other positive weights that add up to 1, instead of simply $1/K$). In order for a PU-constructed stabilization function to satisfy (19), we need a generalization of (14), i.e.

$$\langle \mathcal{B} v, \psi_k \mathcal{B} v \rangle \leq C_N^{(k)} a_k(v, v), \quad \forall v \in V_N,$$

where $a_k(\cdot, \cdot)$ is derived from $a(\cdot, \cdot)$ by introducing the weight ψ_k into its integrals.

4. Numerical examples

In this section, we will perform some numerical tests for the evaluation of the previously presented PU-construction of a stabilization function for Nitsche's method.

We will construct our PU spaces with a stretch factor $\alpha = 1.3$ and local spaces of bilinear ($V_i = \text{span}\langle\{1, x, y\}\rangle$) or trilinear ($V_i = \text{span}\langle\{1, x, y, z\}\rangle$) polynomials, unless explicitly stated otherwise. By h we will refer to the side-length of the (unstretched) patches in the cover. The PU functions will be constructed by Shepard's method as described in Section 2.2, enforcing the required regularity for each problem by choosing the appropriate B-Spline (see Fig. 6 for an example of C^0 functions). The PU basis for the stabilization functions will be derived from constant Shepard weights (see Fig. 7).

In all examples, essential boundary conditions will be imposed in the whole boundary. This means Dirichlet boundary conditions for the Poisson problem, and a larger set of boundary conditions for the higher-order problems we will consider.

4.1. Poisson problem in the unit square

Let us begin this section with the simplest example: the Poisson problem on a unit square domain $\Omega = (-1, 1)^2$, with the prescribed solution $u(x, y) = \sin(\pi x) + \sin(\pi y)$, that will allow us to evaluate the convergence behavior of our proposed method through uniform

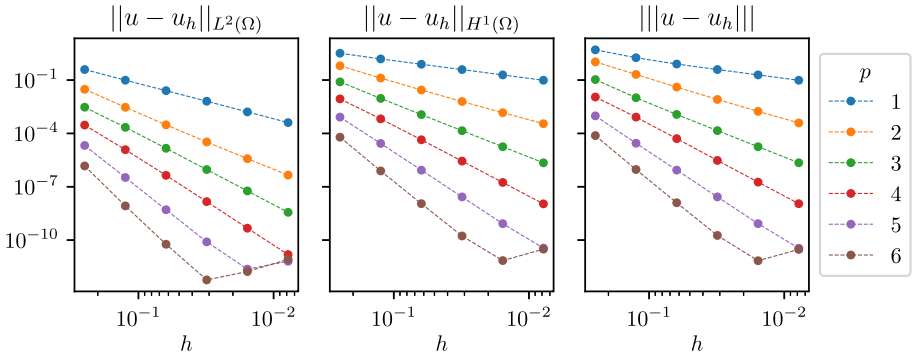


Fig. 8. Convergence plots from Example 4.1.

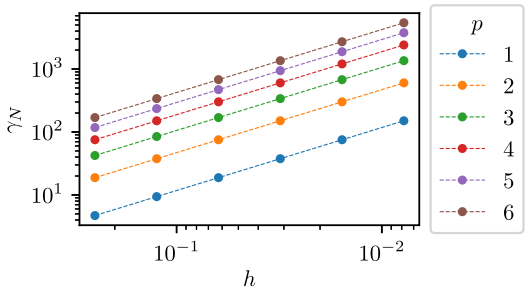


Fig. 9. Values of the stabilization function from Example 4.1.

refinement. In this case, we use local polynomial spaces with degrees from 1 to 6, and the resulting convergence plots can be found in Fig. 8. Apart from the stagnation and round-off errors arising for small h and large p (due to ill-conditioning of the matrices), we observe optimal convergence $\mathcal{O}(h^{p+1})$ for the L^2-norm, as well as $\mathcal{O}(h^p)$ for the H^1 and energy norms. In this simple domain, the resulting stabilization functions are constant over $\partial\Omega$, and the values are shown in Fig. 9, from which we can extract the behavior $\gamma_N = \mathcal{O}(p^2 h^{-1})$.

4.2. Arbitrarily large stabilization function

The following example is based on [6, Example 5.1]. We will again solve the Poisson problem with the same solution as before, but now replacing the fixed domain Ω by the varying domain

$$\Omega_\epsilon = (-1-\epsilon, 1+\epsilon)^2, \quad \epsilon \searrow 0.$$

We will use a fixed background cover that produces patches at the boundary with

$$\frac{|\omega_{\mathrm{FT}} \cap \Omega_\epsilon|}{|\omega_{\mathrm{FT}} \cap \partial\Omega_\epsilon|} = \mathcal{O}(\epsilon),$$

to illustrate the problem that the patch-aggregation technique should address. A visualization of the domain and the cover is included in Fig. 10.

The resulting errors in different norms with decreasing ϵ are shown in Fig. 11. Contrary to the results from [6], we do not observe an increasing error in the $H^1(\Omega_\epsilon)$-norm as $\epsilon \to 0$. The reason for this can be understood by comparing our PUM discretization space to their FE discretization with quadrilateral elements.

First of all, as it is explained there, minimizing the energy norm of the error $|||u_h - u|||$ for $\gamma_N \gg 0$ amounts to first minimizing $\left\| \gamma_N^{\frac{1}{2}}(u_h - u) \right\|_{L^2(\partial\Omega_\epsilon)}$, and then minimizing $\|\nabla(u_h - u)\|_{L^2(\Omega_\epsilon)}$ in the remaining d.o.f.'s not fixed by the previous minimization. In our PUM space, every local space V_i coming from a patch intersecting the boundary (but not the corners) has 2 d.o.f.'s with support at the boundary and 1 that is equally 0 there, which can thus be used to approximate $\nabla u \cdot \mathbf{n}$. For patches intersecting the corners, however, all 3 d.o.f.'s are fixed by the boundary condition, but the volume of their flat-top region is ϵ^2. So far this is all similar to the FE discretization with quadrilateral elements from [6, Example 5.1]. The difference comes from the fact that FE basis functions fixed by the boundary conditions have gradients of magnitude ϵ^{-1}, because they have a value of order ~ 1 at a boundary node and a value 0 at some neighboring interior node, which is at a distance $\sim \epsilon$; this is not the case with our PUM basis functions: since

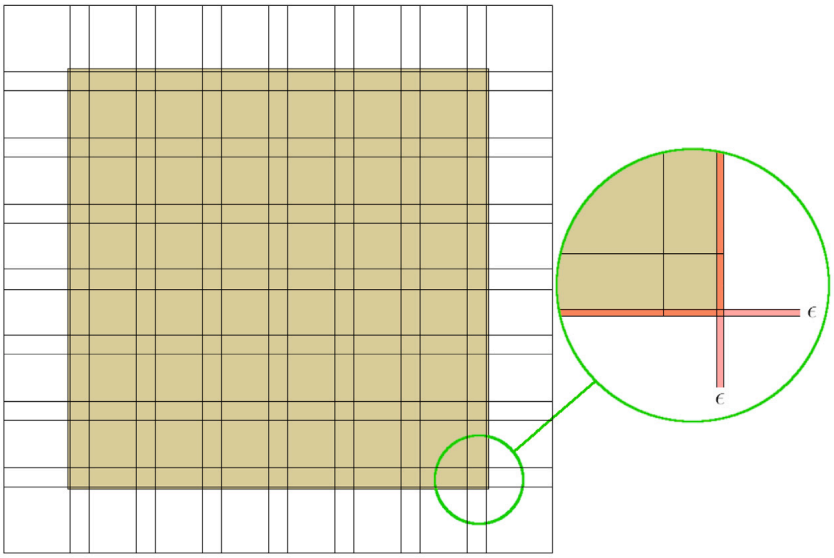


Fig. 10. Domain and cover of Example 4.2, showing the influence of $0 < \epsilon \ll 1$ at the boundary.

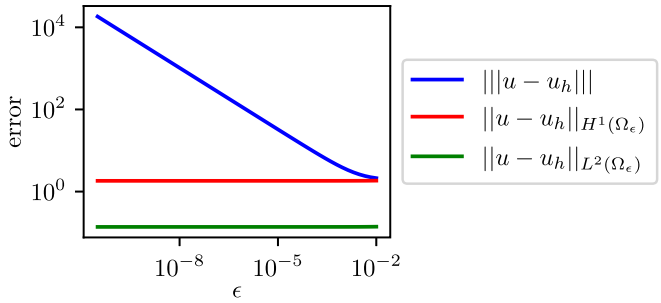


Fig. 11. Errors for the problem from Example 4.2.

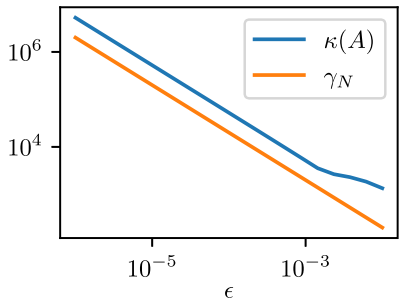


Fig. 12. Value of the stabilization function and condition number of the resulting stiffness matrix from Example 4.2.

patches overlap with one another and the volume of such overlap is independent of ϵ, the gradients of basis functions fixed by the boundary conditions are also independent of ϵ. This is the reason why we do not observe an increasing $H^1(\Omega_\epsilon)$-error as $\epsilon \to 0$.

Nevertheless, the stabilization function γ_N (which, as in the previous example, has the same value on every patch), does grow arbitrarily large (which is the driving force behind the increasing error in the energy norm), and as a consequence, the condition number of the matrix also grows arbitrarily. Both values are shown in Fig. 12 with respect to ϵ.

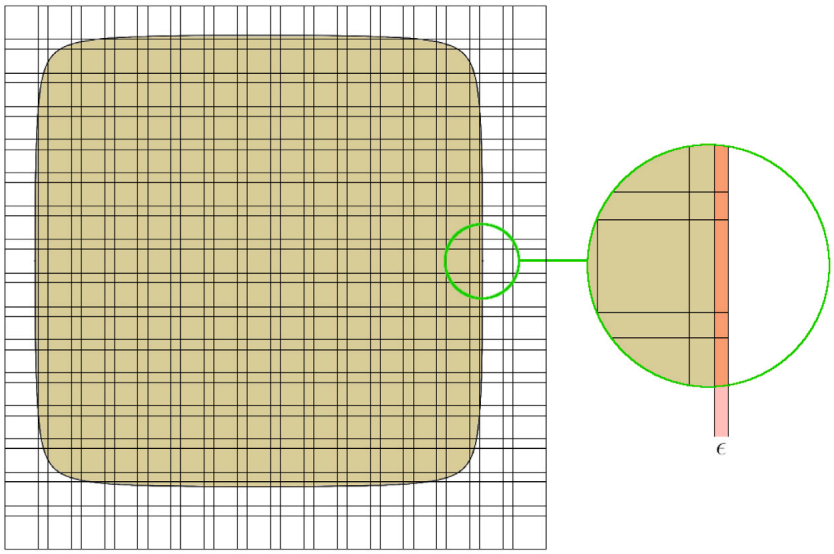


Fig. 13. Domain and cover of Example 4.3, showing the influence of $0 < \epsilon \ll 1$ at the boundary.

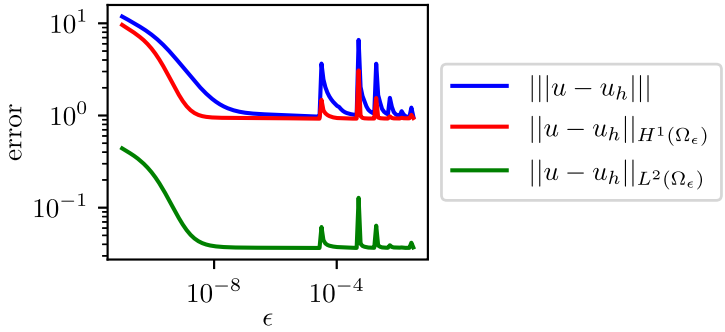


Fig. 14. Errors for the rounded-square problem from Example 4.3.

4.3. A rounded square domain

This example is based on [6, Example 5.4]. The underlying analytical solution is the same as in the previous two examples, but we now choose a domain with a curved boundary. In the cited reference, the domain is $\Omega_\epsilon = \{(x, y) \in \mathbb{R}^2 : x^8 + y^8 < 1 + \epsilon\}$ (defined by a polynomial curve), but here we will use a slightly different one, whose boundary in the first quadrant is given by the rational Bézier curve of degree 2 with control points $\{(1 + \epsilon, 0), (1 + \epsilon, 1 + \epsilon), (0, 1 + \epsilon)\}$ and associated weights $\{1, 5, 1\}$. The whole boundary is completed by properly rotating this curve around the origin. The domain and its cover are shown in Fig. 13.

The resulting errors are plotted in Fig. 14, where we observe similar phenomena to those from [6]. We can observe 6 intermediate peaks in each error norm, associated to the values of ϵ at which 6 different sets of patches at the boundary lose their flat-top regions. At the end, only one set of 4 patches with very small flat-top region remains, and these are responsible for the increasing error as $\epsilon \to 0$. In Fig. 15 we plot two measures of the error at the boundary, to illustrate the driving force of the error as $\epsilon \to 0$: as expected, it is the approximation of the normal derivative which deteriorates with decreasing ϵ, not the one of the boundary value itself.

To avoid this, we consider two approaches: first, increasing the polynomial degree of the local spaces associated to patches at the boundary, and secondly, applying the patch aggregation approach from Section 2.3. The results for each case are shown in Fig. 16(a) and Fig. 16(b), respectively. As it can be seen, by increasing the polynomial degree we can significantly reduce the errors for very small ϵ, as well as for the peaks in intermediate values. Nevertheless, patch aggregation eliminates any dependency with ϵ and, as a result, the graph is trivial. The $L^\infty(\partial\Omega_\epsilon)$-norm of the stabilization parameter with respect to ϵ (with and without patch aggregation) is shown in Fig. 17.

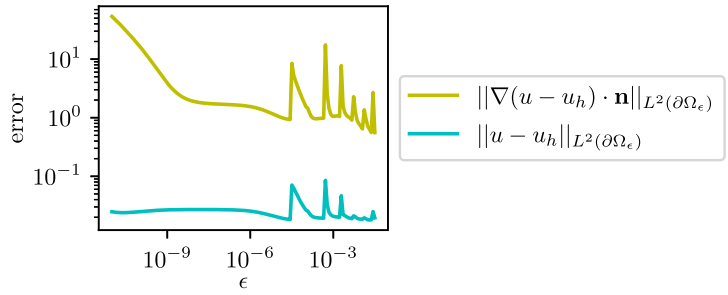
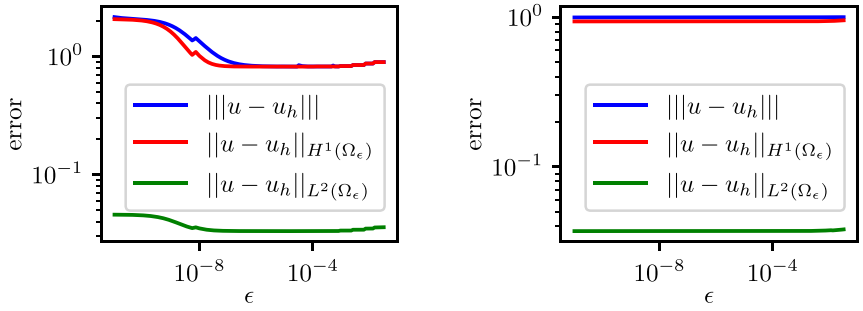
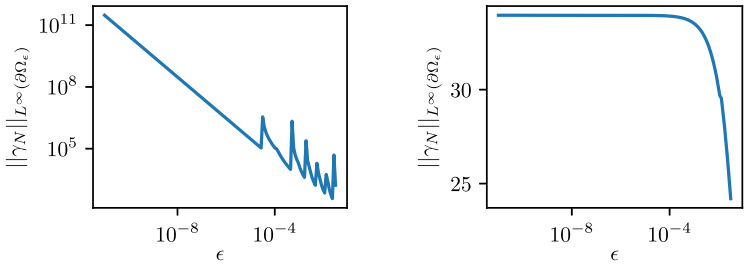


Fig. 15. Boundary errors for the rounded-square problem from Example 4.3.



(a) With local spaces of degree 2 at boundary patches. (b) With patch aggregation.

Fig. 16. Errors for the problem from Example 4.3.



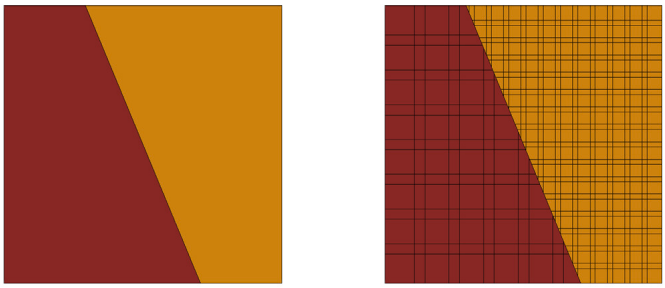
(a) Without patch aggregation. (b) With patch aggregation.

Fig. 17. Stabilization function for the problem from Example 4.3.

4.4. Interface problem

Let us now consider an interface problem taken from [16]. Once again, it will be the Poisson problem, this time in the domain $\Omega = (0,1)^2$ with the exact solution $u(x,y) = [\cosh(\pi y) - \coth(\pi)\sinh(\pi y)]\sin(\pi y)$. Just like there, we split the domain into two parts with a straight line that goes through the center $(0.5, 0.5)$ and is rotated by $\pi/8$ (see Fig. 18(a)). The covers for each part will both be constructed from the bounding box $(0,1)^2$. Instead of using the same refinement level for the two parts, we will couple a refinement level ℓ for the left part with $\ell + 1$ for the right part. One such configuration is shown in Fig. 18(b).

In Fig. 19 we show the convergence of our approach when using patch aggregation on every part and the γ-based averaging (18) (which in this case does not make a big difference, since the only source of contrast is the different discretization levels on each side, but the patches of the right part are only half as big as those from the left side). Similarly as in Fig. 8, the graph for the energy norm is almost identical to that of the H^1-norm, and therefore not shown here. Optimal convergence is observed in all norms for all the considered polynomial degrees, up to round-off errors.



(a) Partitioned domain Ω. (b) Covers for each part of Ω.

Fig. 18. Domain from Example 4.4, with one of the considered pairs of covers.

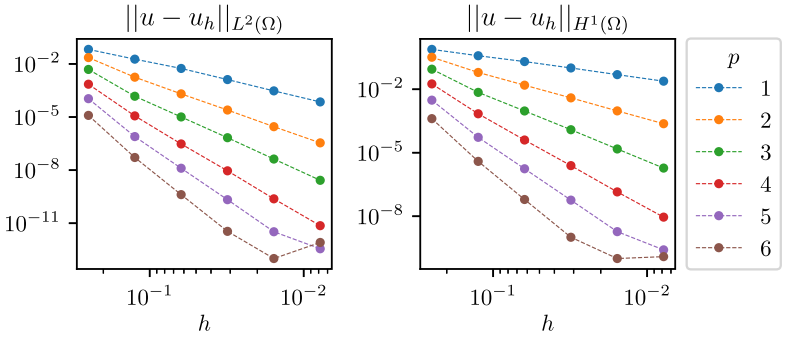


Fig. 19. Convergence graph for the interface problem from Example 4.4.

4.5. Shell problem

Let us now consider [5, Example 5.1], a Kirchhoff–Love plate problem on an annular domain, defined by a NURBS parametrization on $\Xi = (0,1)^2$. The exact solution is chosen as

$$\mathbf{u}(\xi,\eta) = \xi \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|} + \left(e^{\xi} - 1\right) \xi\, \mathbf{a}_3, \quad (\xi,\eta) \in \bar{\Xi}, \tag{20}$$

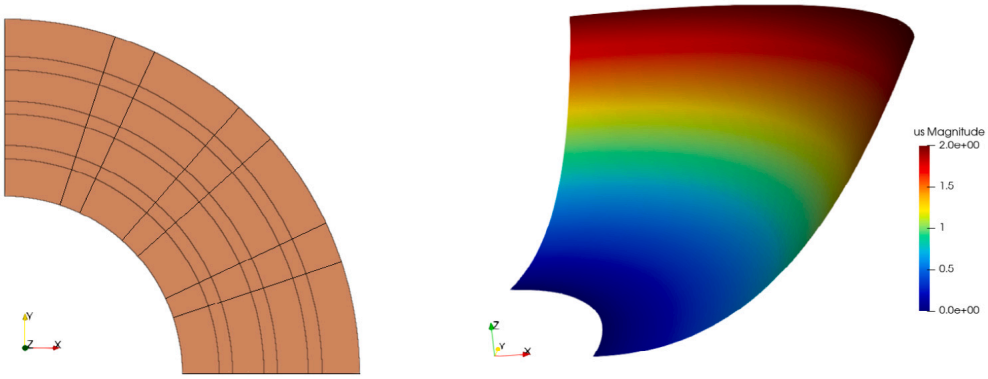
where $\mathbf{a}_1 \equiv \mathbf{a}_1(\xi,\eta) \in \mathbb{R}^3$ is the first element of the covariant basis given by the parametrization, and $\mathbf{a}_3 = (0,0,1)$ is the surface normal. In Fig. 20 we show the physical domain Ω with a cover, as well as the solution. For more details, we refer to [5] and the repository where the authors have gathered their data.[2]

Contrary to what is done in the paper we take as reference, we impose the essential boundary conditions in the whole boundary, and for simplicity we use $E = 1$ (since it is only a global scaling constant). In this problem, there are 3 kinds of essential boundary conditions: displacements, normal rotations and normal displacements at the corners. The conjugate quantity to the boundary displacement is the *ersatz force* $\mathbf{T}$, which can be split into *in-plane* and *out-of-plane* components, $\mathbf{T}(\mathbf{v}) = \underline{\mathrm{T}}(\mathbf{v}) + \mathrm{T}_3(\mathbf{v})\mathbf{a}_3$, each one of which leads to a different scaling in the stabilization function. Thus, we use 4 different stabilization functions, constructed from the following trace inequalities:

$$\left\| \gamma_I^{-\frac{1}{2}} \underline{\mathrm{T}}(\mathbf{v}) \right\|^2_{L^2(\partial\Omega)} \leq \frac{\rho^2}{4} a^S(\mathbf{v},\mathbf{v}), \qquad \left\| \gamma_O^{-\frac{1}{2}} \mathrm{T}_3(\mathbf{v}) \right\|^2_{L^2(\partial\Omega)} \leq \frac{\rho^2}{4} a^S(\mathbf{v},\mathbf{v}),$$
$$\left\| \gamma_\theta^{-\frac{1}{2}} B_{nn}(\mathbf{v}) \right\|^2_{L^2(\partial\Omega)} \leq \frac{\rho^2}{4} a^S(\mathbf{v},\mathbf{v}), \qquad \left\| \gamma_C^{-\frac{1}{2}} [\![B_{nt}(\mathbf{v})]\!] \right\|^2_{L^2(\chi_D)} \leq \frac{\rho^2}{4} a^S(\mathbf{v},\mathbf{v}),$$

where $a^S(\cdot,\cdot)$ is the bilinear form containing the membrane and bending interior terms, $B_{nn}(\mathbf{v}) = \underline{n} \cdot \underline{\underline{B}}(\mathbf{v}) \cdot \underline{n}$ is the bending moment, $B_{nt}(\mathbf{v}) = \underline{n} \cdot \underline{\underline{B}}(\mathbf{v}) \cdot \underline{t}$ is the twisting moment, χ_D is the set of corners and $[\![B_{nt}(\mathbf{v})]\!]$ is the jump in the twisting moment at a given corner. For simplicity, we write $\|q\|^2_{L^2(\chi_D)} = \sum_{C_i \in \chi_D} q^2|_{C_i}$. Once again, for the specific details, we refer to [5], from which we borrow (most of) the notation.

[2] https://github.com/wdas/shell-obstacle-course



(a) Covered domain Ω. (b) Manufactured solution $\mathbf{u}$ given by (20).

Fig. 20. Physical domain from Example 4.5 and exact solution of the problem.

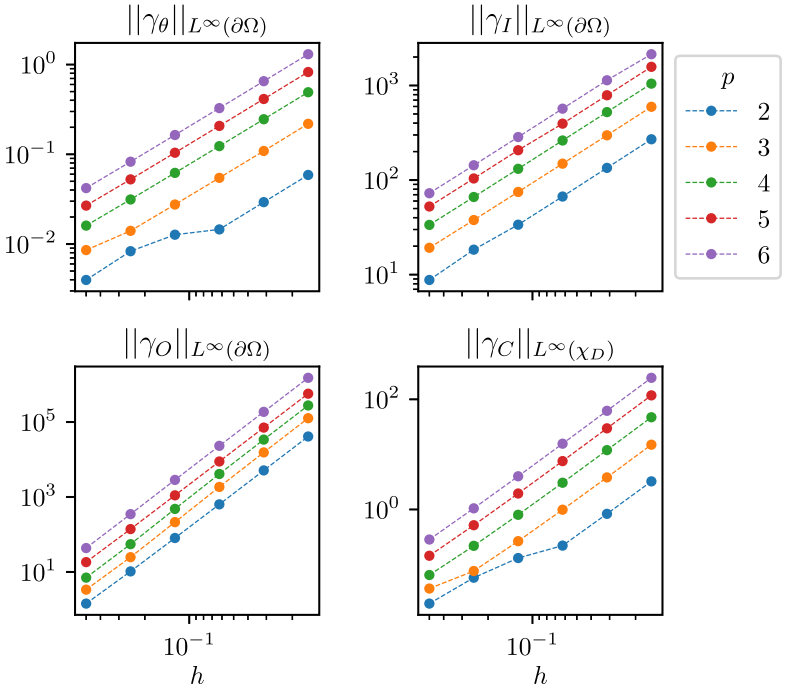


Fig. 21. Stabilization values from Example 4.5.

In Fig. 21 we show the maximum values of the different stabilization functions for each refinement level and different polynomial degrees. As expected, we observe $\gamma_I, \gamma_\theta \sim \mathcal{O}(h^{-1})$, $\gamma_C \sim \mathcal{O}(h^{-2})$ and $\gamma_O \sim \mathcal{O}(h^{-3})$. The relationship with p is not as simple as in the Poisson case.

In Fig. 22 we show the behavior of 4 norms of the error for several polynomial degrees. We again observe optimal convergence rates, i.e. $\mathcal{O}(h^{p+1})$ for the L^2-norm, $\mathcal{O}(h^p)$ for the H^1-norm, and $\mathcal{O}(h^{p-1})$ for the H^2-norm, up to round-off errors. The only suboptimal case is that of the L^2-norm for $p = 2$, where we observe $\mathcal{O}(h^2)$ convergence, which is however expected, following [5, Theorem 5]. In the case of the energy norm, however, we observe super-optimal convergence rates, lying somewhere between $\mathcal{O}(h^{p-1})$ and $\mathcal{O}(h^p)$. The reason for this that the bending component of the energy error, which does converge with the H^2-rate, is shadowed by the membrane component, which converges with the H^1-rate, but is larger than the former. For completeness, let us state the definition

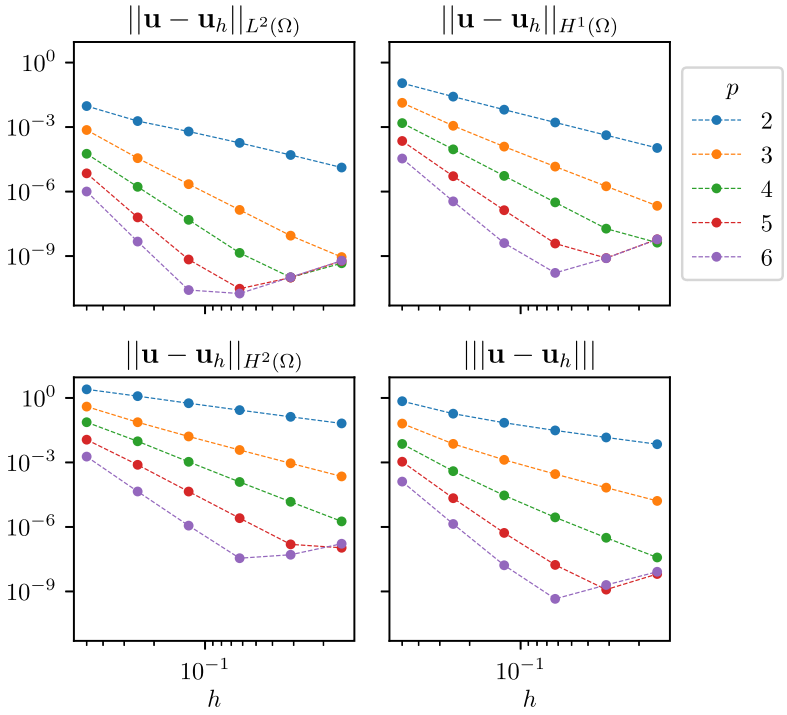


Fig. 22. Convergence plots from Example 4.5.

of the energy norm

$$|||\mathbf{v}||| := a^S(\mathbf{v},\mathbf{v}) + \left\|\gamma_I^{-\frac{1}{2}}\underline{\mathrm{T}}(\mathbf{v})\right\|^2_{L^2(\partial\Omega)} + \left\|\gamma_I^{\frac{1}{2}}\underline{v}\right\|^2_{L^2(\partial\Omega)} + \left\|\gamma_O^{-\frac{1}{2}}\mathrm{T}_3(\mathbf{v})\right\|^2_{L^2(\partial\Omega)} + \left\|\gamma_O^{\frac{1}{2}}v_3\right\|^2_{L^2(\partial\Omega)} + \\ + \left\|\gamma_\theta^{-\frac{1}{2}}B_{nn}(\mathbf{v})\right\|^2_{L^2(\partial\Omega)} + \left\|\gamma_\theta^{\frac{1}{2}}\theta_n(\mathbf{v})\right\|^2_{L^2(\partial\Omega)} + \left\|\gamma_C^{-\frac{1}{2}}[\![B_{nt}(\mathbf{v})]\!]\right\|^2_{L^2(\chi_D)} + \left\|\gamma_C^{\frac{1}{2}}v_3\right\|^2_{L^2(\chi_D)}.$$

4.6. Purely-local stabilization

Finally, let us go back to a comment made at the end of Section 3.3. As mentioned there, the local GEPs required in the PUM are considerably larger than those from the FEM, especially in three dimensions. This is due to the fact that the local GEP for computing the coefficient $\gamma_N^{(k)}$ relies not only on the local space V_k, but also on the local spaces associated to neighboring patches.

Nevertheless, we will compute the stabilization functions with purely local GEPs, and evaluate the suitability of the resulting $\hat{\gamma}_N$. In the case of the Poisson problem, instead of the matrices from (15), we may use

$$\begin{aligned}\hat{A}^k_{(l,m)} &:= A^k_{(k,l),(k,m)} = \int_{\partial\Omega}\psi_k(\nabla(\varphi_k\vartheta^l_k)\cdot\mathbf{n})(\nabla(\varphi_k\vartheta^m_k)\cdot\mathbf{n})\mathrm{d}\Gamma,\\ \hat{B}^k_{(l,m)} &:= B^k_{(k,l),(k,m)} = \int_{\Omega}\psi_k\nabla(\varphi_k\vartheta^l_k)\cdot\nabla(\varphi_k\vartheta^m_k)\mathrm{d}\Omega,\end{aligned} \tag{21}$$

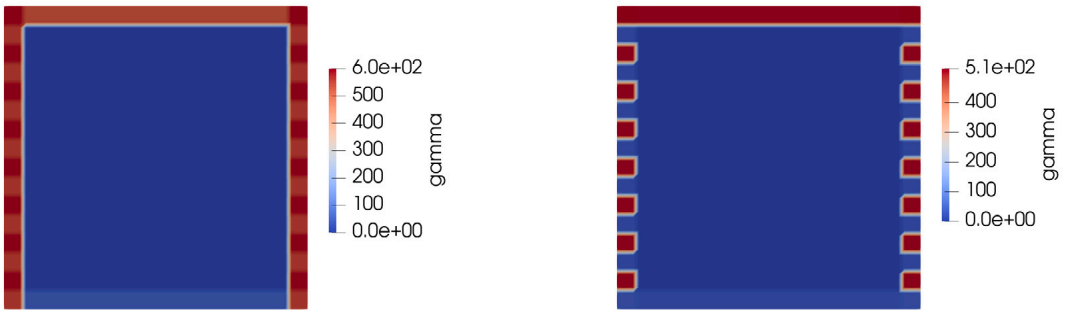
which amounts to a reduction of the dimension of each local GEP to the corresponding local space of the PUM.

In Fig. 23 we show the different results obtained for the Poisson problem in the unit square $\Omega = (0,1)^2$ with polynomial local spaces of alternating degree in the y direction. That means that half of the local spaces have degree 1 (in particular, those from patches at the bottom), and half have degree 4 (in particular, those from patches at the top). As we can see (and as expected), the purely-local stabilization function is never larger than the rigorously-computed stabilization function. Even more, for the patches with degree 1, a rigorous computation yields significantly larger coefficients if they are surrounded by patches of degree 4 than if they are surrounded by patches of degree 1.

These results, however, do not imply that the stabilization function $\hat{\gamma}_N$ resulting from a purely local computation is not valid. In order to properly assess the validity of a stabilization function, we can use the trace inequality (10): given some discrete space $V_N \in H^1(\Omega)$ and a tentative stabilization function $\hat{\gamma}_N \in L_2(\Omega)$, $\hat{\gamma}_N > 0$, let $\beta > 0$ be such that

$$\left\|\hat{\gamma}_N^{-\frac{1}{2}}\nabla v\cdot\mathbf{n}\right\|^2_{L^2(\partial\Omega)} \leq \beta\,\|\nabla v\|^2_{L^2(\Omega)}, \quad \forall v \in V_N.$$

92



(a) Rigorously computed γ_N. (b) Purely-local stabilization function $\hat{\gamma}_N$.

Fig. 23. Comparison of the stabilization function resulting from a purely local computation with that resulting from a rigorous computation. The discretization level is 4 (16 patches in each dimension) and the polynomial degree of the local spaces alternates between 1 and 4 in the y direction.

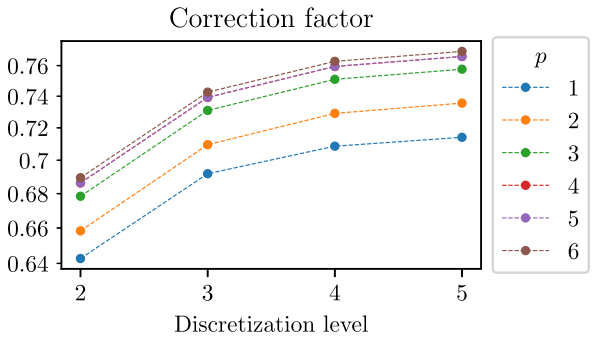


Fig. 24. Correction factors for purely-local stabilization functions in the unit square $\Omega = (0,1)^2$ with $(1,p)$-alternating polynomial degrees in the y direction.

If $\beta < 1$, then $\hat{\gamma}_N$ is a valid stabilization function, otherwise $\hat{\beta}\hat{\gamma}$ is, for any $\hat{\beta} > \beta$. We can thus interpret β, which can be computed through a global GEP, as a minimum *correction factor*.

Let us now generalize the problem from Fig. 23 to different discretization levels and different polynomial degrees (instead of simply 4): we alternate local spaces of degree 1 with local spaces of degree $p \in \{1, \dots, 6\}$ (the case $p = 1$ is of course homogeneous). The obtained correction factors are shown in Fig. 24, where we observe in all cases values smaller than 1, and limited with growing discretization level and polynomial degree.

In what follows, we will compute the correction factors associated to purely-local stabilization functions in a variety of cases. In each of our experiments, we will set the polynomial degrees of local spaces at the boundary at random from $\{p, P\}$. We then produce K different configurations ($K = 100$ for the first 3 problems, $K = 30$ for the last 3), and gather the resulting correction factors into a histogram. The results are shown in Fig. 25, where each experiment is designed as follows:

(a) Square: Poisson equation in the unit square $\Omega = (0,1)^2$, at discretization level 5, with $p = 1$ and $P = 10$.
(b) Circle: Poisson equation in the 2-dimensional ball $\Omega = B((0.5, 0.5), 0.5)$, which we discretize at level 6 based on the bounding box $(0,1)^2$ with, as before, $p = 1$ and $P = 10$. We use the patch-aggregation technique to avoid having patches with too small or even empty flat-top region.
(c) Cube: Poisson equation in the unit cube $\Omega = (0,1)^3$, at discretization level 3, with $p = 1$ and $P = 5$.
(d) Shell: Kirchhoff–Love shell problem from Section 4.5, at discretization level 4, with $p = 2$ and $P = 6$. We compute a correction factor associated to each one of the 4 stabilization functions, following the trace inequalities

$$\left\| \gamma_I^{-\frac{1}{2}} \underline{\mathrm{T}}(\mathbf{v}) \right\|^2_{L^2(\partial\Omega)} \leq \beta_I a^S(\mathbf{v},\mathbf{v}), \qquad \left\| \gamma_O^{-\frac{1}{2}} \mathrm{T}_3(\mathbf{v}) \right\|^2_{L^2(\partial\Omega)} \leq \beta_O a^S(\mathbf{v},\mathbf{v}),$$
$$\left\| \gamma_\theta^{-\frac{1}{2}} B_{nn}(\mathbf{v}) \right\|^2_{L^2(\partial\Omega)} \leq \beta_\theta a^S(\mathbf{v},\mathbf{v}), \qquad \left\| \gamma_C^{-\frac{1}{2}} [\![B_{nt}(\mathbf{v})]\!] \right\|^2_{L^2(\chi_D)} \leq \beta_C a^S(\mathbf{v},\mathbf{v}).$$

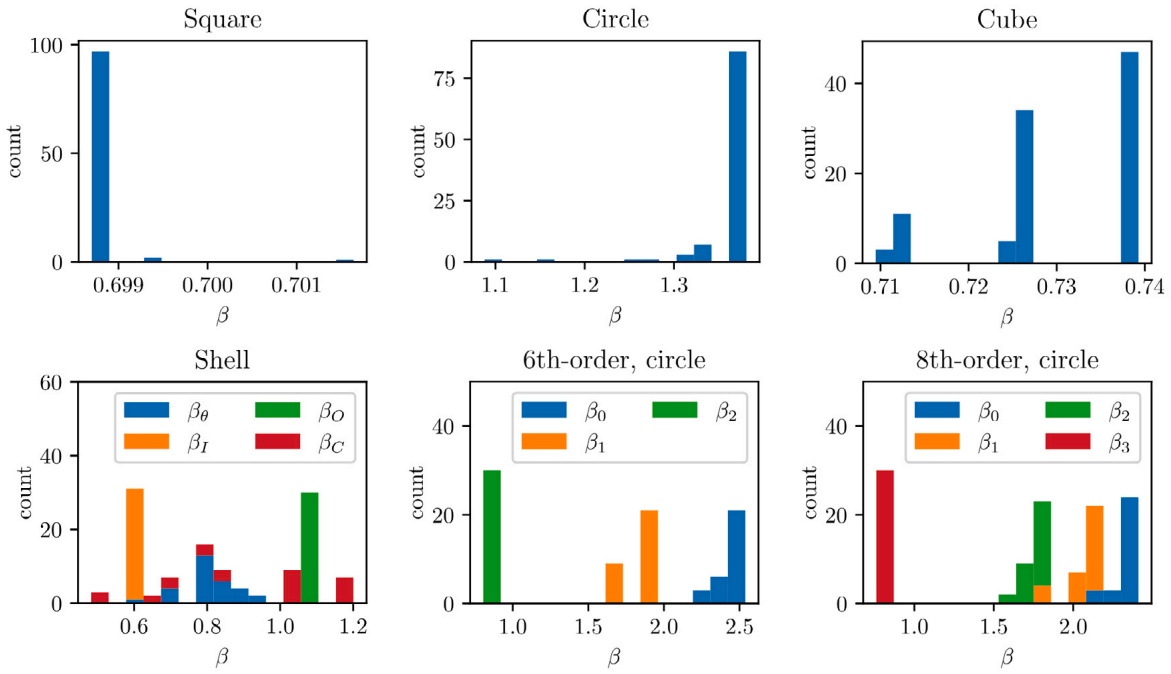


Fig. 25. Correction factors of the purely-local stabilization functions computed in 6 different examples.

(e) 6th-order: $-\Delta^3 u = f$ in the 2-dimensional ball $\Omega = B((0.5, 0.5), 0.5)$, discretized as previously, with $p = 3$ and $P = 8$. The problem has 3 different kinds of essential boundary conditions, with associated trace inequalities

$$\left\| \gamma_0^{-\frac{1}{2}} \nabla(\Delta^2 v) \cdot \mathbf{n} \right\|^2_{L^2(\partial\Omega)} \leq \beta_0 \left\| \nabla(\Delta v) \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } u,$$

$$\left\| \gamma_1^{-\frac{1}{2}} \Delta^2 v \right\|^2_{L^2(\partial\Omega)} \leq \beta_1 \left\| \nabla(\Delta v) \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } \nabla u \cdot \mathbf{n},$$

$$\left\| \gamma_2^{-\frac{1}{2}} \nabla(\Delta v) \cdot \mathbf{n} \right\|^2_{L^2(\partial\Omega)} \leq \beta_2 \left\| \nabla(\Delta v) \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } \Delta u.$$

(f) 8th-order: $-\Delta^4 u = f$ in the 2-dimensional ball $\Omega = B((0.5, 0.5), 0.5)$, discretized as in the previous case, with $p = 4$ and $P = 8$. The problem has 4 different kinds of essential boundary conditions, with associated trace inequalities

$$\left\| \gamma_0^{-\frac{1}{2}} \nabla(\Delta^3 v) \cdot \mathbf{n} \right\|^2_{L^2(\partial\Omega)} \leq \beta_0 \left\| \Delta^2 v \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } u, \qquad \left\| \gamma_1^{-\frac{1}{2}} \Delta^3 v \right\|^2_{L^2(\partial\Omega)} \leq \beta_1 \left\| \Delta^2 v \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } \nabla u \cdot \mathbf{n},$$

$$\left\| \gamma_2^{-\frac{1}{2}} \nabla(\Delta^2 v) \cdot \mathbf{n} \right\|^2_{L^2(\partial\Omega)} \leq \beta_2 \left\| \Delta^2 v \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } \Delta u, \qquad \left\| \gamma_3^{-\frac{1}{2}} \Delta^2 v \right\|^2_{L^2(\partial\Omega)} \leq \beta_3 \left\| \Delta^2 v \right\|^2_{L^2(\Omega)}, \quad \text{b.c. } \nabla(\Delta u) \cdot \mathbf{n}.$$

We observe in all cases correction factors that are either smaller than 1, or not much larger. Factors contributing to larger correction factors are irregular intersections of the patches with the domain, and presence of higher order derivatives in the trace inequality. In the 6th-order problem, where we combine both circumstances, we observe $\beta_0 \in (2, 2.5)$ (whose trace inequality involves 5th-order derivatives), and similar results also for β_0 and β_1 in the 8th-order problem.

Note that we do not consider interface problems because, as we already mentioned, the stabilization function for each side of the interface can be computed independently, meaning that, for the purpose of computing correction factors, one interface problem is equivalent to two simple boundary problems.

In order to reduce (or even avoid) the need for a re-scaling, we can go one step further by reducing the r.h.s. matrices $\hat{B}^k$ from (21) to the flat top region of the corresponding patch, i.e.

$$\hat{B}^k_{(l,m)} = \int_{\Omega \cap \omega_{k,\mathrm{FT}}} \nabla \vartheta^l_k \cdot \nabla \vartheta^m_k \mathrm{d}\Omega. \tag{22}$$

94

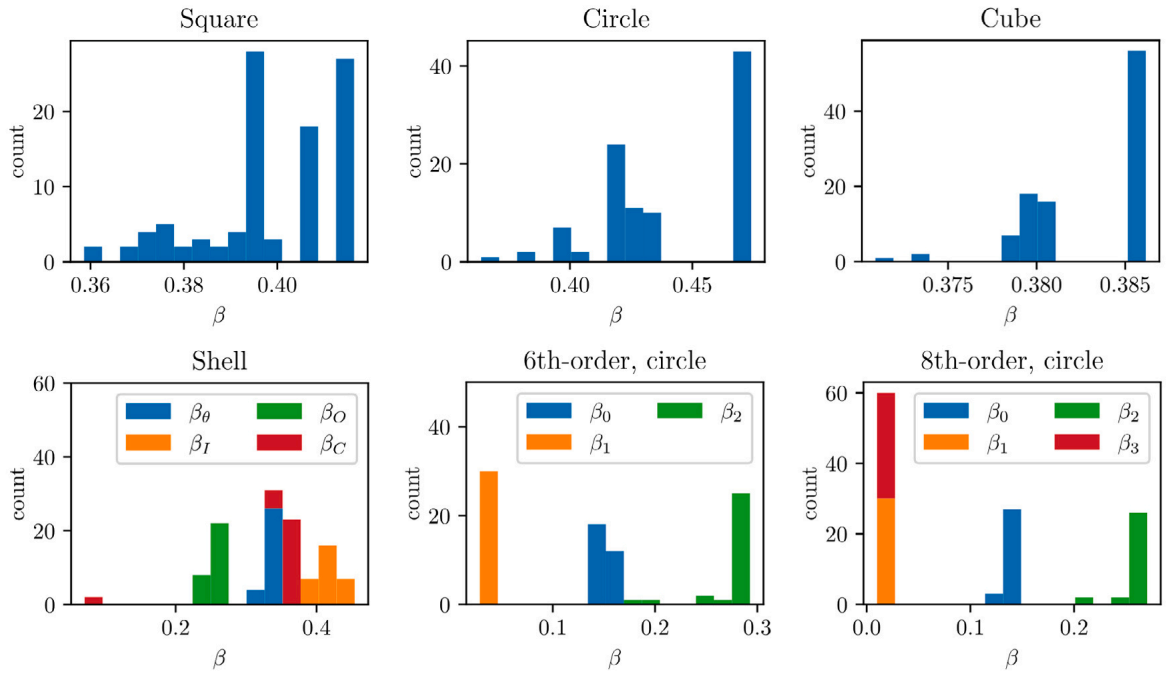


Fig. 26. Correction factors of the purely-local stabilization functions computed in 6 different examples, with the r.h.s. term reduced to the FT-region of each patch as in (22).

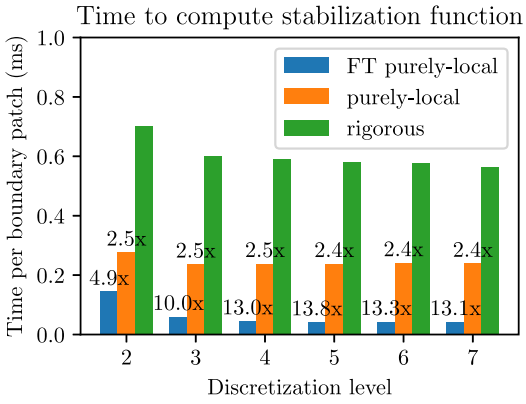


Fig. 27. Time measurements of purely-local and rigorously computed stabilization computations, for the Poisson problem in the unit cube $\Omega = (0,1)^3$, with homogeneous polynomial degree 1. Computations are made on a single thread, and we include the speedup of each purely-local version with respect to the rigorous one in each case.

It is easy to see that the largest eigenvalue in this case is guaranteed to be larger or equal than the one associated to the matrices from (21). Nevertheless, this approach is only practical as long as $\Omega \cap \omega_{k,\mathrm{FT}}$ is not too small compared with ω_k. We now replicate the previous tests with the new r.h.s. term for the local GEPs, and show the resulting correction factors in Fig. 26. As we can see, all values are now below 1, and in fact, below 0.5, which is the upper bound for rigorously computed stabilization functions.

As additional argument in favor of the purely-local computation, in Fig. 27 we show a time-comparison with respect to the complete procedure, for the Poisson problem in the unit cube $\Omega = (0,1)^3$, with our initially proposed purely-local approach, and with the FT-reduced r.h.s. matrix (which we refer to as "FT purely-local" approach). The reasons behind the speedup are the cheaper assembly of the matrices in (21) with respect to those from (15) (because we consider only the corresponding local space), and the cheaper assembly also of (22) with respect to the r.h.s. matrix from (21) (because we only integrate on flat top regions).

5. Conclusions

We have successfully developed a local construction of the Nitsche stabilization function compatible with PUM discretization spaces, thus avoiding the need for a constant stabilization parameter. Together with that, we have also introduced a patch aggregation procedure within the PUM framework to avoid the appearance of patches with arbitrarily small flat-top regions, which, as we have seen in some of our numerical examples, produce arbitrarily large coefficients for the stabilization function.

Our construction, which has been designed with the FT-PUM in mind but, as already mentioned, also encompasses the element-wise constant stabilization function for the Nitsche-FEM, can in principle be extended to any other method relying on partition-of-unity functions for its basis functions. The key characteristic of the method is that, given some discrete function space V and a PU-definition of the stabilization function $\gamma_N = \sum_k \gamma_N^{(k)} \psi_k$, each constant coefficient $\gamma_N^{(k)}$ can be derived from a local GEP involving only the basis functions from V whose support intersects $\mathrm{supp}(\psi_k)$.

As mentioned at the end of Section 3.3, the computational cost associated to constructing a stabilization function within the PUM is larger than within the FEM, specially in 3D domains. However, in our numerical experiments section we have suggested a cheaper construction of the stabilization function which, even if it cannot be guaranteed to produce solvable systems, has been shown to do so in a variety of problems.

This work also paves the way for using Nitsche's method in combination with the PUM in more complicated scenarios, e.g. nonlinear problems, or shell-shell and solid-shell coupling. With respect to solid-shell coupling, the PUM offers the possibility to discretize both components in essentially the same way, and a paper on this topic might also be published in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Griebel, M.A. Schweitzer, A particle-partition of unity method for the solution of elliptic, parabolic, and hyperbolic PDEs, SIAM J. Sci. Comput. 22 (3) (2000) 853–890, http://dx.doi.org/10.1137/S1064827599355840.
[2] M.A. Schweitzer, A parallel multilevel partition of unity method for elliptic partial differential equations, in: Lect. Notes Comput. Sci. Eng., vol. 29, Springer, 2003, http://dx.doi.org/10.1007/978-3-642-59325-3.
[3] J. Nitsche, Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, in: Abhandlungen aus dem mathematischen Seminar der Universität Hamburg, vol. 36, Springer, 1971, pp. 9–15.
[4] J. Benzaken, J.A. Evans, R. Tamstorf, Constructing Nitsche's method for variational problems, 2022, http://dx.doi.org/10.48550/ARXIV.2203.02603, URL https://arxiv.org/abs/2203.02603.
[5] J. Benzaken, J.A. Evans, S.F. McCormick, R. Tamstorf, Nitsche's method for linear Kirchhoff–Love shells: Formulation, error analysis, and verification, Comput. Methods Appl. Mech. Engrg. 374 (2021) 113544, http://dx.doi.org/10.1016/j.cma.2020.113544, URL https://www.sciencedirect.com/science/article/pii/S0045782520307295.
[6] F. de Prenter, C. Lehrenfeld, A. Massing, A note on the stability parameter in Nitsche's method for unfitted boundary value problems, Comput. Math. Appl. 75 (12) (2018) 4322–4336, http://dx.doi.org/10.1016/j.camwa.2018.03.032.
[7] F. Chouly, M. Fabre, P. Hild, R. Mlika, J. Pousin, Y. Renard, An overview of recent results on Nitsche's method for contact problems, in: S. Bordas, E. Burman, M. Larson, M. Olshanskii (Eds.), UCL Workshop 2016, in: Geometrically Unfitted Finite Element Methods and Applications, vol. 121, UCL (University College London), Springer, London, United Kingdom, 2016, pp. 93–141, http://dx.doi.org/10.1007/978-3-319-71431-8_4, URL https://hal.archives-ouvertes.fr/hal-01403003.
[8] M.A. Schweitzer, An algebraic treatment of essential boundary conditions in the particle-partition of unity method, SIAM J. Sci. Comput. 31 (2) (2009) 1581–1602, http://dx.doi.org/10.1137/080716499.
[9] M. Griebel, M.A. Schweitzer, A Particle-Partition of Unity Method–Part V: Boundary Conditions, in: Geometric Analysis and Nonlinear Partial Differential Equations, Springer, 2003, pp. 519–542.
[10] J. Dolbow, I. Harari, An efficient finite element method for embedded interface problems, Internat. J. Numer. Methods Engrg. 78 (2) (2009) 229–252, http://dx.doi.org/10.1002/nme.2486.
[11] A. Embar, J. Dolbow, I. Harari, Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements, Internat. J. Numer. Methods Engrg. (2010) n/a–n/a, http://dx.doi.org/10.1002/nme.2863.
[12] M. Griebel, M.A. Schweitzer, A particle-partition of unity method–Part II: Efficient cover construction and reliable integration, SIAM J. Sci. Comput. 23 (5) (2001) 1655–1682, http://dx.doi.org/10.1137/S1064827501391588.
[13] A. Johansson, M.G. Larson, A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary, Numer. Math. 123 (4) (2012) 607–628, http://dx.doi.org/10.1007/s00211-012-0497-1.
[14] M. Fabre, J. Pousin, Y. Renard, A fictitious domain method for frictionless contact problems in elasticity using Nitsche's method, SMAI J. Comput. Math. 2 (2016) 19–50, http://dx.doi.org/10.5802/smai-jcm.8, URL https://smai-jcm.centre-mersenne.org/item/SMAI-JCM_2016__2__19_0.
[15] V. Thomée, Galerkin Finite Element Methods for Parabolic Problems, Springer Berlin Heidelberg, 2006, http://dx.doi.org/10.1007/3-540-33122-0.
[16] D. Schillinger, I. Harari, M.-C. Hsu, D. Kamensky, S.K. Stoter, Y. Yu, Y. Zhao, The non-symmetric Nitsche method for the parameter-free imposition of weak boundary and coupling conditions in immersed finite elements, Comput. Methods Appl. Mech. Engrg. 309 (2016) 625–652, http://dx.doi.org/10.1016/j.cma.2016.06.026.
[17] W. Jiang, C. Annavarapu, J.E. Dolbow, I. Harari, A robust Nitsche's formulation for interface problems with spline-based finite elements, Internat. J. Numer. Methods Engrg. 104 (7) (2015) 676–696, http://dx.doi.org/10.1002/nme.4766.

Chebyshev smoothing with adaptive block-FSAI preconditioners for the multilevel solution of higher-order problems

Pablo Jiménez Recio[1,2] and Marc Alexander Schweitzer[1,2]

[1] *Institute for Numerical Simulation, Universität Bonn, 53111 Bonn, Germany*
[2] *Fraunhofer SCAI, 53757 Sankt Augustin, Germany*

Abstract

In this paper, we assess the performance of adaptive and nested factorized sparse approximate inverses as smoothers in multilevel V-cycles, when smoothing is performed following the Chebyshev iteration of the fourth kind. For our test problems, we rely on the partition of unity method to discretize the biharmonic and triharmonic equations in a multilevel manner. Inspired by existing algorithms, we introduce a new adaptive algorithm for the construction of sparse approximate inverses, based on the block structure of matrices arising in the partition of unity method. Additionally, we also present a new (and arguably simpler) formulation of the Chebyshev iteration of the fourth kind.

1 Introduction

The use of the Chebyshev iteration for concatenating smoothing steps in multigrid/multilevel iterations has been redefined by the significant work of Lottes [17], which has allowed dispensing with the estimate of the lower end of the fine spectrum of the preconditioned matrix. The new Chebyshev iteration of the fourth kind was quickly adopted in PETSc and the deal.II library [3]. In this work, we combine this new Chebyshev iteration with underlying Factorized Sparse Approximate Inverse (FSAI) preconditioners, as developed in recent years by Janna et al. [11, 13, 14], and which have already been used as smoothers in multilevel approaches [12, 20]. We will put these preconditioners into practice for a variety of problems generated with the Partition of Unity Method (PUM) of Schweitzer and Griebel [7, 23].

The paper is organized as follows. In Section 2 we introduce the block-FSAI preconditioner, where our focus on block matrices follows from the natural block structure of matrices arising from the PUM. We pay special attention to developing our own adaptive algorithm for the construction of a suitable sparsity pattern for PUM block matrices, based on the one from [11], and also describe the possibility of nesting FSAI preconditioners.

In Section 3 we present the Chebyshev iteration, explain its *raison d'être* and delve into its use as a way to optimally concatenate smoothing steps in multilevel iterations, paying special attention to the work of Lottes [17]. Given the simplicity of the new Chebyshev iteration of the fourth kind (which we simplify even further),

we choose it for our work instead of the optimized (but more involved) version also introduced by Lottes.

In Section 4 we provide a brief description of the PUM, whose discrete function spaces possess a natural geometric multilevel structure, as well as basis functions of any desired regularity.

Finally, in Section 5 we present some numerical experiments to attest the effectiveness and efficiency of the (adaptive and nested) block-FSAI preconditioners, which we embed within a Chebyshev iteration. The PUM allows us to test our multilevel solvers on higher-order problems such as the biharmonic and triharmonic equations, which provide a challenging scenario, since the usual Jacobi and Gauss-Seidel iterations are not particularly effective smoothers in this case, or at least not for PUM discretizations. We note that the multilevel solution of the biharmonic equation has already been investigated by other authors, see e.g. [25] and the references therein.

2 The FSAI preconditioner

For an invertible matrix $A \in \mathbb{R}^{N\times N}$ and some predefined sparsity pattern $\mathcal{P} \subset \{1,\dots,N\}^2$, a *sparse approximate inverse* (SAI) of A based on $\mathcal{P}$ is some matrix $M \in \mathbb{R}^{N\times N}$ with sparsity pattern $\mathcal{P}(M) \coloneqq \{(i,j) : M_{ij} \neq 0\} \subset \mathcal{P}$ which approximates A^{-1}, usually by solving

$$M \coloneqq \underset{\mathcal{P}(\tilde{M})\subset\mathcal{P}}{\arg\min} \|I - \tilde{M}A\|_F,$$

with $\|\cdot\|_F$ the Frobenius norm, defined by $\|B\|_F^2 = \operatorname{tr}(B^\top B)$. This constitutes the basis of the SAI preconditioner [24].

Note however that, even if the matrix A is symmetric positive definite (s.p.d.), the resulting matrix M will not be s.p.d., and thus the preconditioned matrix will also not be s.p.d. The Factorized Sparse Approximate Inverse (FSAI) preconditioner overcomes this issue, and (following the construction of Janna et al. [14]) it does so by constructing a sparse approximation $G \approx L^{-1}$ to the inverse of the (unknown) triangular Cholesky factor L of $A = LL^\top$, leading to $M \coloneqq G^\top G \approx L^{-\top}L^{-1} = A^{-1}$, with the preconditioned matrix being $GAG^\top$.

Let us first introduce the notation that we will use for working with block-structured matrices as those arising in the PUM.

Definition 2.1. Let $n \in \mathbb{N}$ and $\mathfrak{B} \coloneqq \{m_i\}_{i=1}^n$ a set of n positive integers with $N = \sum_{i=1}^n m_i$. By saying that a matrix $A \in \mathbb{R}^{N\times N}$ has the block structure $\mathfrak{B}$, we allow ourselves to interpret it as

$$A = (A_{ij}), \quad i,j \in \{1,\dots,n\},$$
$$A_{ij} = (a_{kl}) \in \mathbb{R}^{m_i\times m_j}.$$

We further denote

$$\mathcal{P}(A) \coloneqq \{(i,j) : A_{ij} \neq \mathbb{O}_{m_i\times m_j}\} \subset \{1,\dots,n\}^2$$

the sparsity pattern of such a matrix, and call a matrix block-diagonal if

$$\mathcal{P}(A) \subset \{(i,i) : i \in \{1,\dots,n\}\}.$$

Finally, we denote $\operatorname{diag}_{\mathfrak{B}}(A)$ the block-diagonal component of a matrix A, i.e. the matrix with diagonal blocks A_{ii} and $\mathbb{O}_{m_i\times m_j}$ off-diagonal blocks.

For any block matrix B as in Definition 2.1 and sets of indices $\mathcal{I}, \mathcal{J} \subset \{1, \dots, n\}$, we will denote $B[\mathcal{I}, \mathcal{J}]$ the restriction of B to row indices in $\mathcal{I}$ and column indices in $\mathcal{J}$. We may now define the block-FSAI preconditioner.

Definition 2.2. Let A be a s.p.d. matrix with block structure $\mathfrak{B} \coloneqq \{m_i\}_{i=1}^n$ as in Definition 2.1. Let $\mathcal{P} \subset \{1, \dots, n\}^2$ be some lower triangular sparsity pattern containing the diagonal, i.e.

$$(i,i) \in \mathcal{P} \quad \forall i \in \{1, \dots, n\}, \qquad \text{and} \qquad i \geq j \quad \forall (i,j) \in \mathcal{P}, \tag{1}$$

Let us denote its rows, with and without the corresponding row index, by

$$\mathcal{P}_i \coloneqq \{j : (i,j) \in \mathcal{P}\}, \quad \tilde{\mathcal{P}}_i \coloneqq \mathcal{P}_i \setminus \{i\}, \qquad i \in \{1, \dots, n\}.$$

We define the *block-FSAI matrix for A based on $\mathcal{P}$* as the matrix F with block structure $\mathfrak{B}$, sparsity pattern $\mathcal{P}(F) \subset \mathcal{P}$ and non-zero block-entries given by

$$F_{ii} = \mathbb{I}_{m_i \times m_i}, \quad F[\{i\}, \tilde{\mathcal{P}}_i] = -A[\{i\}, \tilde{\mathcal{P}}_i]\, A[\tilde{\mathcal{P}}_i, \tilde{\mathcal{P}}_i]^{-1}, \qquad i \in \{1, \dots, n\}. \tag{2}$$

The *block-FSAI preconditioning matrix for A based on $\mathcal{P}$* is then $M \coloneqq F^\top S^{-1} F$, where S is the block-diagonal matrix (also with block structure $\mathfrak{B}$) given by

$$S_{ii} \coloneqq A_{ii} - A[\{i\}, \tilde{\mathcal{P}}_i] A[\tilde{\mathcal{P}}_i, \tilde{\mathcal{P}}_i]^{-1} A[\tilde{\mathcal{P}}_i, \{i\}], \qquad i \in \{1, \dots, n\}. \tag{3}$$

We may also write $M = G^\top G$ for $G = S^{-1/2} F$, with the preconditioned matrix then being $GAG^\top$.

Remark 2.1. From now on, unless explicitly stated otherwise, every reference to a "lower triangular sparsity pattern" will mean also including the diagonal, i.e. a sparsity pattern fulfilling both conditions in (1).

Lemma 2.1. *Let A be a s.p.d. matrix with block structure $\mathfrak{B}$ and F be defined as in* (2) *for some lower triangular sparsity pattern $\mathcal{P}$. Let S be the block-diagonal matrix with blocks given by* (3)*. It holds that*

$$\operatorname{diag}_{\mathfrak{B}}(FA) = S = \operatorname{diag}_{\mathfrak{B}}(FAF^\top),$$

and in particular it follows that the block-diagonal of the block-FSAI preconditioned matrix, $\operatorname{diag}_{\mathfrak{B}}(GAG^\top)$*, is the identity matrix.*

Proof. The first identity follows directly from (2) and (3). Indeed,

$$(FA)_{ii} = F[\{i\}, \mathcal{P}_i]\, A[\mathcal{P}_i, \{i\}] = S_{ii}, \quad \forall i \in \{1, \dots, n\}.$$

The second one can also be easily checked relying on the first one and writing

$$(FAF^\top)_{ii} = \begin{pmatrix} I_{ii} & F[\{i\}, \tilde{\mathcal{P}}_i] \end{pmatrix} \begin{pmatrix} A_{ii} & A[\{i\}, \tilde{\mathcal{P}}_i] \\ A[\tilde{\mathcal{P}}_i, \{i\}] & A[\tilde{\mathcal{P}}_i, \tilde{\mathcal{P}}_i] \end{pmatrix} \begin{pmatrix} I_{ii} \\ F^\top[\tilde{\mathcal{P}}_i, \{i\}] \end{pmatrix} =$$
$$= \begin{pmatrix} I_{ii} & F[\{i\}, \tilde{\mathcal{P}}_i] \end{pmatrix} \begin{pmatrix} S_{ii} \\ 0 \end{pmatrix} = S_{ii}.$$

The final statement follows trivially for $G = S^{-1/2} F$. □

Remark 2.2. For the simplest choice of a diagonal sparsity pattern

$$\mathcal{P} = \{(i,i) : \, i \in \{1, \dots, n\}\},$$

the resulting matrix F is the identity matrix and $S = \mathrm{diag}_{\mathfrak{B}}(A)$, so the block-FSAI preconditioner reduces to a block-Jacobi preconditioner.

Let us now introduce an important concept which, as we will show, is deeply connected to the FSAI preconditioner.

Definition 2.3. For a s.p.d. matrix $B \in \mathbb{R}^{N \times N}$, the *Kaporin number* is defined as

$$\beta(B) \coloneqq \frac{\mathrm{tr}(B)}{N \, \det(B)^{1/N}},$$

that is, the ratio between the algebraic and the geometric means of the eigenvalues of B.

Remark 2.3. Some trivial properties of the Kaporin number $\beta(\cdot)$ are that $\beta(B) \geq 1$, and that $\beta(B) = 1$ if and only if all eigenvalues of B coincide. Additionally, from this last property it follows that, for any full-rank matrix C, $\beta(CBC^\top) = 1$ if and only if $C^\top C = \alpha B^{-1}$ for some $\alpha > 0$.

We now reproduce a result from Janna et al. [11], which will be useful for the subsequent theorem.

Lemma 2.2. *Let $B \in \mathbb{R}^{N \times N}$ be a s.p.d. matrix with block structure $\mathfrak{B}$. Then $\beta(JBJ^\top)$ is minimized over full-rank block-diagonal matrices J if and only if there exists $\alpha > 0$ such that $(J^\top J)_{kk} = \alpha (B_{kk})^{-1}$.*

Proof. Let $D = \mathrm{diag}_{\mathfrak{B}}(B)$, so that in particular, $\mathrm{tr}(JBJ^\top) = \mathrm{tr}(JDJ^\top)$ and also $\mathrm{tr}(D^{-1}B) = N$. Since $\det(JBJ^\top) = \det(JDJ^\top)\det(D^{-1}B)$, it can easily be shown that

$$\beta(JBJ^\top) = \frac{\mathrm{tr}(JBJ^\top)}{N \det(JBJ^\top)^{1/N}} = \frac{\mathrm{tr}(JDJ^\top)}{N \det(JDJ^\top)^{1/N}} \, \frac{N}{N \det(D^{-1}B)^{1/N}} = \\ = \beta(JDJ^\top) \, \beta(D^{-1/2} B D^{-1/2}).$$

The problem has thus been reduced to minimizing $\beta(JDJ^\top)$, and the claim follows by noting that $\beta(JDJ^\top) \geq 1$ and that the value of 1 is attained if and only if $J^\top J = \alpha D^{-1}$ for some $\alpha > 0$ (recall Remark 2.3). □

We may now introduce the well-known optimality property of the FSAI preconditioner with respect to the Kaporin number (see the work of Kaporin himself [16]).

Theorem 2.1 (FSAI optimality)**.** *Let A be a s.p.d. matrix with block structure $\mathfrak{B}$, $\mathcal{P}$ be some lower triangular sparsity pattern, and F, S be the block-FSAI matrices based on $\mathcal{P}$ as in Definition 2.2.*

Then $G \coloneqq S^{-1/2} F$ minimizes the Kaporin number of the preconditioned matrix $\beta(\tilde{G} A \tilde{G}^\top)$ over all full-rank matrices $\tilde{G}$ with sparsity pattern $\mathcal{P}$.

Proof. We proceed by characterizing the minimizers of $\beta(\tilde{G} A \tilde{G}^\top)$ over the considered matrices.

First of all, any full-rank matrix with (lower triangular) sparsity pattern $\mathcal{P}$ can be written as $\tilde{G} = J(I + \tilde{F})$, for some non-singular block-diagonal matrix J and some (also lower triangular) $\tilde{F}$ having $\mathbb{O}_{m_i \times m_i}$ as diagonal blocks (i.e. being strictly lower triangular). As a result, and since $\det(I + \tilde{F}) = 1$, we can write

$$\beta(\tilde{G}A\tilde{G}^\top) = \frac{\operatorname{tr}\left(J(I + \tilde{F})A(I + \tilde{F}^\top)J^\top\right)}{N\left[\det(A)\det(J^\top J)\right]^{1/N}},$$

where only the numerator depends on $\tilde{F}$. Moreover,

$$\operatorname{tr}\left(J(I + \tilde{F})A(I + \tilde{F}^\top)J^\top\right) = \sum_{k=0}^{n} \operatorname{tr}\left(((I + \tilde{F})A(I + \tilde{F})^\top)_{kk}(J^\top J)_{kk}\right),$$

so when minimizing $\beta(\tilde{G}A\tilde{G}^\top)$ with respect to $\tilde{F}$, the non-zero entries in the kth row of $\tilde{F}$, i.e. $\tilde{F}[\{k\}, \tilde{\mathcal{P}}_k]$, must minimize

$$\begin{aligned}\operatorname{tr}\left(((I + \tilde{F})A(I + \tilde{F})^\top)_{kk}(J^\top J)_{kk}\right) = \operatorname{tr}\left(A_{kk}(J^\top J)_{kk}\right) + \\ + \operatorname{tr}\left(A[\{k\}, \tilde{\mathcal{P}}_k]\tilde{F}^\top[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk}\right) + \\ + \operatorname{tr}\left(\tilde{F}[\{k\}, \tilde{\mathcal{P}}_k]A[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk}\right) + \\ + \operatorname{tr}\left(\tilde{F}[\{k\}, \tilde{\mathcal{P}}_k]A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]\tilde{F}^\top[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk}\right),\end{aligned}$$

or equivalently

$$2\operatorname{tr}\left(A[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk}\right) + \operatorname{tr}\left(\tilde{F}[\{k\}, \tilde{\mathcal{P}}_k]A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]\tilde{F}^\top[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk}\right).$$

Differentiating the above expression with respect to $\tilde{F}[\{k\}, \tilde{\mathcal{P}}_k]$ (cf. [21, Section 2.5]) and setting the result to 0 yields

$$2A[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk} + 2A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]\tilde{F}^\top[\tilde{\mathcal{P}}_k, \{k\}](J^\top J)_{kk} = \mathbb{O}_{m_k \times m_k},$$

and since $(J^\top J)_{kk}$ is a s.p.d. block, we arrive at

$$\tilde{F}[\{k\}, \tilde{\mathcal{P}}_k] = -A[\{k\}, \tilde{\mathcal{P}}_k]A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1},$$

which means that $I + \tilde{F}$ coincides with the FSAI matrix F from (2), independently of J.

Having fixed $\tilde{F}$, it remains to minimize $\beta\left(J(FAF^\top)J^\top\right)$ with respect to J. This last step is quite simple, since from Lemma 2.2 it follows that $J^\top J = \alpha \operatorname{diag}_{\mathfrak{B}}(FAF^\top)$ for some $\alpha > 0$, which, recalling Lemma 2.1, is fulfilled by $J = S^{-1/2}$ with $\alpha = 1$. □

2.1 Adaptive block-FSAI

It remains to address the question about how to properly choose the triangular sparsity pattern $\mathcal{P}$ for the computation of F. The simplest case is to fix it a priori, e.g. as the triangular pattern extracted from the original matrix A, or from some power of it: A^2, A^3, etc.

That approach, however, does not rely on the actual entries of A, but only on its structure. A more involved approach is to adaptively construct a sparsity pattern relying specifically on the entries of A. Following the work Janna and Ferronato [11], we will develop an adaptive algorithm with the aim of minimizing the Kaporin number of the preconditioned matrix, $\beta(GAG^\top)$. It will rely on the following result, also taken from [11].

Lemma 2.3. *Let $A \in \mathbb{R}^{N\times N}$ be a s.p.d. matrix with block structure $\mathfrak{B} = \{m_i\}_{i=1}^n$, $\mathcal{P}$ be some lower triangular sparsity pattern, and F and S be the FSAI matrices based on $\mathcal{P}$ as in Definition 2.2. Then, for $G \coloneqq S^{-1/2}F$, it holds that*

$$\beta(GAG^\top) = \left(\frac{\det(\mathrm{diag}_{\mathfrak{B}}(FAF^\top))}{\det(A)}\right)^{1/N} = \left(\frac{\prod_{i=1}^n \det((FAF^\top)_{ii})}{\det(A)}\right)^{1/N}.$$

Proof. From Lemma 2.1 it follows that $\mathrm{tr}(GAG^\top) = N$, and further noting that $\det(F) = 1$,

$$\beta(GAG^\top) = \frac{\mathrm{tr}(GAG^\top)}{N\,\det(GAG^\top)^{1/N}} = \left(\frac{\det(S)}{\det(A)}\right)^{1/N} = \left(\frac{\prod_{i=1}^n \det((FAF^\top)_{ii})}{\det(A)}\right)^{1/N}.$$

□

Therefore, when enlarging the kth row of the sparsity pattern, $\mathcal{P}_k$, we will aim at minimizing $\det((FAF^\top)_{kk})$. Let us now recall Lemma 2.1 in the form

$$(FA)_{kk} = (FAF^\top)_{kk} = A_{kk} - A[\{k\},\, \tilde{\mathcal{P}}_k]A[\tilde{\mathcal{P}}_k,\, \tilde{\mathcal{P}}_k]^{-1}A[\tilde{\mathcal{P}}_k,\, \{k\}],$$

and additionally note that, by construction of F,

$$(FA)[\{k\},\, \tilde{\mathcal{P}}_k] = A[\{k\},\, \tilde{\mathcal{P}}_k] - A[\{k\},\, \tilde{\mathcal{P}}_k]\, A[\tilde{\mathcal{P}}_k,\, \tilde{\mathcal{P}}_k]^{-1}\, A[\tilde{\mathcal{P}}_k,\, \tilde{\mathcal{P}}_k] \equiv 0, \tag{4}$$

so that $\mathcal{P}(FA) \cap \mathcal{P} = \{(k,k) :\, k \in \{1,\dots,n\}\}$. We now consider how

$$\Delta = \det((FAF^\top)_{kk})$$

would be affected if we were to add a single index $c \in \{0,\dots,k-1\} \setminus \mathcal{P}_k$ to the kth row of the sparsity pattern.

Lemma 2.4. *Let A be a s.p.d. matrix with block structure $\mathfrak{B} = \{m_i\}_{i=1}^n$ and F be the FSAI matrix for a given lower triangular sparsity pattern $\mathcal{P}$. Let $k \in \{1,\dots,n\}$ be some row index, and $c \in \{0,\dots,k-1\} \setminus \mathcal{P}_k$ some admissible column index. If we denote $F^{(c)}$ the FSAI matrix that would result from the expanded sparsity pattern $\mathcal{P} \cup \{(k,c)\}$, it holds that*

$$(F^{(c)}AF^{(c)\top})_{kk} = (FAF^\top)_{kk} - (FA)_{kc}W_c^{-1}(FA)_{kc}^\top,$$

where

$$W_c \coloneqq A_{cc} - A[\{c\},\, \tilde{\mathcal{P}}_k]\, A[\tilde{\mathcal{P}}_k,\, \tilde{\mathcal{P}}_k]^{-1}\, A[\tilde{\mathcal{P}}_k,\, \{c\}].$$

Additionally, if we denote $\Delta = \det((FAF^\top)_{kk})$ and $\Delta^{(c)} = \det((F^{(c)}AF^{(c)\top})_{kk})$, and let $H = FA$, the following relationship holds

$$\frac{\Delta^{(c)}}{\Delta} = \frac{\det\left(W_c - H_{kc}^\top H_{kk}^{-1}H_{kc}\right)}{\det(W_c)}.$$

Proof. For the first claim, we merely describe the otherwise tedious process for arriving at the desired expression. First we write

$$(F^{(c)}AF^{(c)\top})_{kk} =$$
$$= A_{kk} - \begin{pmatrix} A_{kc} & A[\{k\}, \tilde{\mathcal{P}}_k] \end{pmatrix} \begin{pmatrix} A_{cc} & A[\{c\}, \tilde{\mathcal{P}}_k] \\ A[\tilde{\mathcal{P}}_k, \{c\}] & A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k] \end{pmatrix}^{-1} \begin{pmatrix} A_{ck} \\ A[\tilde{\mathcal{P}}_k, \{k\}] \end{pmatrix},$$

and then we expand the inverse in terms of the Schur complement matrix W_c. The desired expression can be obtained by properly gathering all terms, while also taking into account that

$$(FA)_{kc} = A_{kc} - A[\{k\}, \tilde{\mathcal{P}}_k]\, A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1} A[\tilde{\mathcal{P}}_k, \{c\}]\,.$$

The result for $\Delta^{(c)}/\Delta$ follows by the relation of determinants of Schur complement matrices with those of the original matrix, and $H_{kk} = (FA)_{kk} = (FAF^\top)_{kk}$. □

This last result thus provides a way to compute $\Delta^{(c)}/\Delta$ without constructing the new kth row of the matrix $F^{(c)}$. One further trivial consequence is that we may ignore column indices c for which $(FA)_{kc} \equiv 0$. At this point, we are able to present our simple adaptive block-FSAI algorithm in Algorithm 1.

Compared to the algorithm from [11], ours does not use gradient descent on the minimization target (since the problem of constructing the sparsity pattern is purely discrete), relies only on a single threshold parameter and a single number of adaptive steps, and does not allow backtracking by removing pairs of indices from the sparsity pattern. Hence our claim for "simplicity", even at the cost of flexibility. Note that, since we add at most one new column index per row at each adaptive step, the t_{max} parameter determines the maximum number of off-diagonal blocks in any row of the resulting F.

For a similar analysis to ours above we refer to M. Sedlacek's PhD thesis [24, Appendix C].

2.2 Nested FSAI

Lemma 2.1 allows us to reinterpret FSAI preconditioning as applying the basis transformation F, followed by a single step of the block-Jacobi iteration for the transformed matrix $FAF^\top$, and finally a basis transformation back to the original space $F^\top$. In other words, the action of the preconditioner $F^\top S^{-1} F$ can be understood as

$$r \longmapsto v = Fr \longmapsto w = \operatorname{diag}_{\mathfrak{B}}(FAF^\top)^{-1} v \longmapsto p = F^\top w.$$

This interpretation suggests one further twist: instead of the block-Jacobi step for the FSAI-transformed matrix $FAF^\top$, we can apply any other (preferably s.p.d.) preconditioner, and in particular, an additional step of FSAI, this one constructed for the matrix $FAF^\top$ (contrary to the outer one, which is constructed simply for A). In the literature, this has been called first *recurrent* FSAI [14], and recently *nested* FSAI [12]. Ideally, one can concatenate any number of embeddings, i.e.

$$A_0 \coloneqq A, \quad A_k = F_{k-1} A_{k-1} F_{k-1}^\top, \quad k = 1, \ldots, n_\ell,$$

where F_k is the FSAI matrix for A_k (based on some sparsity pattern). At the final level n_ℓ, a standard block-Jacobi step is performed. The matrix M of the resulting

Algorithm 1 Adaptive block-FSAI

Require: A s.p.d. matrix A with the block structure $\mathfrak{B} = \{m_i\}_{i=0}^{n}$, and an initial lower triangular sparsity pattern $\mathcal{P}_0 \subset \{1, \dots, n\}^2$ on said block structure, containing all diagonal pairs of indices. The number of adaptive steps $t_{max} \geq 1$ and a threshold parameter $\tau \in (0, 1]$.

Initialize $\mathcal{P} = \mathcal{P}_0$.
Initialize $\mathcal{R} = \{1, \dots, n\}$ to be the set of row indices which may still be adapted.

for $t = 1, \dots, t_{max}$ **do**
    Compute the FSAI matrix F based on $\mathcal{P}$, according to (2).
        ▷ This requires computing $A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1}$ for every $k \in \mathcal{R}$, which we may store.

    Compute $H = FA$.
        ▷ For the last two steps, we may skip finished rows $k \notin \mathcal{R}$.

    for $k \in \mathcal{R}$ **do**
        Let $\mathcal{C} \coloneqq \{c \in \{0, \dots, k-1\} : H_{kc} \not\equiv 0\}$ be the set of admissible column indices.
            ▷ From (4), $\mathcal{C} \cap \mathcal{P}_k = \emptyset$, i.e. no index already in the row pattern is admissible.

        if $\mathcal{C} = \emptyset$ **then**
            Remove k from $\mathcal{R}$ and continue.
        end if

        Let $W_c \coloneqq A_{cc} - A[\{c\}, \tilde{\mathcal{P}}_k]\, A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1}\, A[\tilde{\mathcal{P}}_k, \{c\}]$ for $c \in \mathcal{C}$, and find

$$c^\star \in \arg\min_{c \in \mathcal{C}} \rho_c, \quad \rho_c \coloneqq \frac{\det(W_c - H_{kc}^\top H_{kk}^{-1} H_{kc})}{\det(W_c)}.$$

            ▷ We may reuse $A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1}$, computed at the latest assembly of F.

        if $\rho_{c^\star} < \tau$ **then**
            $\mathcal{P} \leftarrow \mathcal{P} \cup \{(k, c^\star)\}$.
        else
            $\mathcal{R} \leftarrow \mathcal{R} \setminus \{k\}$.
        end if

    end for
end for

Compute the FSAI matrix F based on the current state of $\mathcal{P}$, according to (2).
        ▷ As above, it is unnecessary to recompute finished rows $k \notin \mathcal{R}$.

return F

preconditioner can be written as

$$M = \hat{F}^\top \hat{S}^{-1} \hat{F}, \quad \hat{F} \coloneqq F_{n_\ell} F_{n_\ell - 1} \cdots F_0, \quad \hat{S} \coloneqq \operatorname{diag}_{\mathfrak{B}}(A_{n_\ell}) = \operatorname{diag}_{\mathfrak{B}}(\hat{F} A \hat{F}^\top).$$

Nesting thus allows us to increase the density of the preconditioning matrix M while avoiding the computational cost of doing it with a single FSAI matrix F, which would require a sufficiently dense sparsity pattern. A higher density of M is achieved by combining multiple F matrices of limited density, each of which can be computed adaptively. The cost is thus transferred to computing each triple matrix product $A_{k+1} = F_k A_k F_k^\top$, which is required to construct the subsequent F_{k+1}.

3 The Chebyshev iteration

Let us now look at a completely different preconditioner, based on a polynomial iteration. Polynomial iterations are iterative methods for the approximation of a solution to a linear system $Ax = b$, such that the error $e_k \coloneqq x - x_k$ evolves as

$$e_k = p_k(A)e_0, \quad k \geq 1,$$

where e_0 is the error of the initial solution x_0, and p_k is some polynomial of degree k with $p_k(0) = 1$. This is equivalent to having

$$x_k = x_0 + q_{k-1}(A)(b - Ax_0), \quad k \geq 1, \tag{5}$$

where $p_k(t) = 1 - tq_{k-1}(t)$. Additionally, $p_k(A)$ controls the residual evolution, i.e. $r_k = p_k(A)r_0$, where $r_k = b - Ax_k$. Under the assumption that A is a s.p.d. matrix, we have that,

$$\frac{\|e_k\|_2}{\|e_0\|_2} \leq \max_{\lambda \in \sigma(A)} |p_k(\lambda)| \leq \max_{\lambda \in [\alpha,\, \beta]} |p_k(\lambda)|,$$

for an interval $[\alpha,\, \beta] \supset \sigma(A)$. The unique solution to the minimax problem

$$\min_{\substack{p_k \in \mathbb{P}_k \\ p_k(0)=1}} \max_{\lambda \in [\alpha,\, \beta]} |p_k(\lambda)|, \quad \beta > \alpha > 0, \tag{6}$$

where $\mathbb{P}_k$ is the set of polynomials of degree k, is the reparametrized and rescaled Chebyshev polynomial of degree k

$$\hat{T}_k(\zeta) \coloneqq \left[T_k\left(\frac{\alpha+\beta}{\alpha-\beta}\right)\right]^{-1} T_k\left(\frac{2\zeta - \alpha - \beta}{\beta - \alpha}\right), \tag{7}$$

where T_k is the standard Chebyshev polynomial (of the first kind), based on the $[-1, 1]$ interval, and thus $\hat{T}_k$ is reparametrized to the $[\alpha,\, \beta]$ interval and rescaled so that $\hat{T}_k(0) = 1$. Let us introduce, in as simple a manner as possible, a minimal set of concepts to understand the proof.

Definition 3.1. Let $[a, b] \subset \mathbb{R}$. We say that a set of functions $\{g_i\}_{i=1}^n$, $g_i \in C([a, b])$, is a *Chebyshev system* (or a *Haar system*) if every non-zero element in $\operatorname{span}\langle g_i \rangle_{i=1}^n$ has at most $n - 1$ distinct roots in $[a, b]$.

Theorem 3.1 (Chebyshev's Alternation Theorem)**.** *Let $[a, b] \subset \mathbb{R}$, $f \in C([a, b])$, and $\{g_i\}_{i=1}^n$, $g_i \in C([a, b])$, be a Chebyshev system. Then $\hat{g} \in G \coloneqq \operatorname{span}\langle g_i \rangle_{i=1}^n$ is a solution to the minimax problem*

$$\min_{g \in G} \max_{t \in [a,b]} |f(t) - g(t)|,$$

if and only if there exist $n+1$ points $a \leq t_0 < \cdots < t_n \leq b$ satisfying

$$|r(t_i)| = \max_{t\in[a,b]} |r(t)|, \quad i = 0, \ldots, n,$$
$$r(t_i) = -r(t_{i-1}), \quad i = 1, \ldots, n,$$

for $r(t) = f(t) - \hat{g}(t)$. The points $\{t_i\}_{i=1}^n$ are called "alternation points" of r in $[a, b]$. Furthermore, if a minimizer exists, then it is unique.

Proof. This is a combination of the Alternation Theorem and the Unicity Theorem in [6]. For a more recent version, we refer to [2]. □

Lemma 3.1. *For $n > 0$, the Chebyshev polynomial T_n, defined in the $[-1, 1]$ interval by*

$$T_n(t) = \cos n\theta, \quad t = \cos\theta \in [-1, 1],$$

has the $n+1$ alternation points

$$t_i = \cos\frac{i\pi}{n}, \quad i = 0, \ldots, n.$$

Proof. Trivial given the trigonometric definition. For more details, we refer to [18]. □

Usually, instead of for polynomials with a pointwise constraint, the minimax property of Chebyshev polynomials is stated for monic polynomials [18, 19], which is easily proved using the Alternation Theorem to show that a minimizer of

$$\min_{q_{k-1}\in\mathbb{P}_{k-1}} \max_{t\in[-1,1]} |t^k - q_{k-1}(t)|$$

satisfies $t^k - q_{k-1}(t) \propto T_k(t)$, and hence the properly rescaled Chebyshev polynomial is the one least deviating from 0 in the $L^\infty([-1, 1])$-norm among all monic polynomials. For the pointwise constraint $p_k(\xi) = 1$ with $\xi \in \mathbb{R}$, one simply needs to restate the minimax problem as

$$\min_{q_{k-1}\in\mathbb{P}_{k-1}} \max_{t\in[-1,1]} |1 - (t - \xi)\, q_{k-1}(t)|,$$

which in this case introduces a caveat: in order to apply the Alternation Theorem, $\{(t-\xi)\, t^i\}_{i=0}^{k-1}$ has to be a Chebyshev system in $C([-1, 1])$, which means that for any nonzero $q \in \mathbb{P}_{k-1}$, $(t - \xi)\, q(t)$ may have at most $k - 1$ roots in $[-1, 1]$, which holds if and only if $\xi \notin [-1, 1]$. This is the reason behind the requirement that $\alpha > 0$ in (6).

Now that we have properly justified the use of Chebyshev polynomials for iterative methods, we need to find an appropriate construction of the iteration (5). Following the recursive rule of Chebyshev polynomials

$$T_0(t) = 1,$$
$$T_1(t) = t,$$
$$T_{n+1}(t) = 2t\, T_n(t) - T_{n-1}(t), \quad n \geq 1,$$

one can write the resulting linear iteration for the system $Ax = b$ in a variety of different ways, as thoroughly discussed by Gutknecht and Röllin [9]. Based on their two-term iteration, we can write the Chebyshev iteration as in Algorithm 2.

Algorithm 2 Chebyshev iteration of degree k

Require: A linear system $Ax = b$ with a s.p.d. matrix A. An initial tentative solution x_0. An interval $[\alpha, \beta] \supset \sigma(A)$ with $0 < \alpha < \beta$. A positive integer $k \geq 1$.

Rewrite $[\alpha, \beta] = [c - d, c + d]$, i.e. set

$$c = \frac{\alpha + \beta}{2}, \quad d = \frac{\beta - \alpha}{2}.$$

Initialize $r = b - Ax_0$, $p = r$.
Initialize $\omega = \frac{1}{c}$, $\psi = \frac{1}{2}(d\omega)^2$.

Initialize $x = x_0 + \omega\, p$

for $i = 2, \ldots, k$ **do**
    $r \leftarrow b - Ax$
    $p \leftarrow r + \psi\, p$
    $\omega \leftarrow \left(c - \frac{\psi}{\omega}\right)^{-1}$
    $\psi \leftarrow \left(\frac{d\omega}{2}\right)^2$
    $x \leftarrow x + \omega\, p$
end for

return x

Additionally, the Chebyshev iteration naturally lends itself to preconditioning with a s.p.d. matrix M, in which case the interval $[\alpha, \beta]$ should contain $\sigma(MA)$ instead of $\sigma(A)$. The inclusion of preconditioning in Algorithm 2 is trivial: we simply replace r by Mr in the initialization as well as in the update of p.

Although estimating the maximal eigenvalue of a s.p.d. matrix A (or a preconditioned MA, with M also s.p.d.) is not a complicated task, the same cannot be said about estimating its minimal eigenvalue. Therefore, techniques have been devised to perform the Chebyshev iteration without actually estimating the minimal eigenvalue (see e.g. the LS-Chebyshev polynomial preconditioner from [4]). This issue will also play a role in the forthcoming subsection.

3.1 Chebyshev smoothing

The Chebyshev iteration is also a Krylov space method, but contrary to the usual Krylov space methods (e.g. CG, GMRES), the absence of inner products in its iteration makes it particularly well-suited to be used not as a solver itself, but (for a fixed, predefined number of iterations) as a preconditioner within another solver (e.g. CG). Note that we presented Algorithm 2 already in this spirit by fixing the number of iterations.

Furthermore, the Chebyshev iteration can be repurposed as a smoother for a multilevel iteration by targeting only the higher end of the spectrum of MA, as initially performed by Adams et al. [1], and later by Baker et al. [5]. This means that, instead of relying on an interval $[\alpha, \beta] \supset \sigma(MA)$, which would require estimates of both the maximal and minimal eigenvalues of MA, one may cover only a right-

subinterval of $\sigma(A)$, e.g. in the form of $[\rho\,\lambda_{max}(MA),\ \lambda_{max}(MA)]$ for some $\rho \in (0,1)$.

In this scenario, the usual smoothing stage consisting of k repeated applications of a preconditioner, i.e. $S = (I - MA)^k$, also known as Richardson smoothing, can be replaced by a single application of the M-preconditioned Chebyshev iteration of degree k, i.e. $S = p_k(MA)$ (since it also involves k matrix-vector products with MA).

Recently, Lottes [17] has sharply reconsidered whether the minimax property (6) satisfied by Chebyshev polynomials of the first kind is still meaningful when using the Chebyshev iteration to concatenate smoothing steps within a multilevel iteration. Following the splitting of the two-level error propagation leading to the Approximation Property and the Smoothing Property (see e.g. Equation (6.1.5) from Hackbusch's monograph [10]), and defining the norm $\|\cdot\|_{M,A}^2$ as

$$\|B\|_{M,A}^2 \coloneqq \sup_{\|v\|_A \leq 1} \|Bv\|_M^2, \quad \|w\|_Q^2 \coloneqq \langle w,\, w\rangle_Q \coloneqq \langle Qw,\, w\rangle,$$

we can formulate the Smoothing Property for a smoother $S = p_k(MA)$ with respect to this norm and bound it as

$$\|AS\|_{M,A}^2 = \sup_{\|v\|_A \leq 1} \|ASv\|_M^2 = \sup_{\|v\|_A \leq 1} \|Ap_k(MA)v\|_M^2 \leq \max_{\lambda \in \sigma(MA)} \lambda p_k(\lambda)^2, \tag{8}$$

where the last step follows by writing the test vectors v in an $\langle\cdot,\cdot\rangle_A$-orthonormal basis of eigenvectors of MA (and thus also of $p_k(MA)$). This points us towards the minimax problem

$$\min_{\substack{p_k \in \mathbb{P}_k \\ p_k(0)=1}} \max_{\lambda \in [\alpha,\,\beta]} \lambda^{1/2} |p_k(\lambda)|, \quad \beta > \alpha > 0, \tag{9}$$

for an interval $[\alpha,\,\beta] \supset \sigma(MA)$. The following lemma gives us the solution.

Lemma 3.2. *Let $k \geq 1$ be a positive integer. There exists $\delta^\star \in (-1, 1)$ such that, for any $\delta \in (-1, \delta^\star)$, the minimax problem*

$$\min_{\substack{p_k \in \mathbb{P}_k \\ p_k(-1)=1}} \max_{t \in [\delta,\,1]} (1+t)^{1/2} |p_k(t)|, \tag{10}$$

has a unique minimizer, which is given by

$$\hat{p}_k(t) = \frac{1}{2k+1}\, W_k\left(-t\right),$$

where W_k is the kth-degree Chebyshev polynomial of the fourth kind. *Furthermore, this also holds for the limit case $\delta = -1$, i.e. $\hat{p}_k$ is also a solution to*

$$\min_{\substack{p_k \in \mathbb{P}_k \\ p_k(-1)=1}} \max_{t \in [-1,\,1]} (1+t)^{1/2} |p_k(t)|.$$

Proof. Let us first note that, for V_k the kth-degree Chebyshev polynomial of the *third* kind (see [18, 19]), the weighted polynomial $(1+t)^{1/2} V_k(t)$ has $k+1$ alternation points $t_i = \cos\theta_i$, $i = 0, \ldots, k$ with

$$\theta_i = \frac{2i\pi}{2i+1} \in [0, \pi],$$

and in particular $t_i \in [\delta^\star,\, 1]$ with

$$\delta^\star \coloneqq \min_{i \in \{0,\ldots,k\}} t_i = -\cos\frac{\pi}{2k+1} > -1.$$

Additionally, the Chebyshev polynomials of the third kind can be written in terms of those of the fourth kind as $V_k(t) = (-1)^k W_k(-t)$.

Now, for any $\delta \in (-1, \delta^\star)$, the set of functions $\{(1+t)^{3/2} t^i\}_{i=0}^{k-1}$ is a Chebyshev system in $C([\delta, 1])$ (because the fixed root at -1 lies outside the interval), and thus it follows from the Alternation Theorem that any minimizer $\hat{q}_{k-1}$ of

$$\min_{q_{k-1} \in \mathbb{P}_{k-1}} \max_{t \in [\delta,\, 1]} (1+t)^{1/2} |1 - (1+t) q_{k-1}(t)|,$$

equivalent to (10) by rewriting $p_k(t) = 1 + (1+t) q_{k-1}(t)$, is characterized by

$$(1+t)^{1/2} \left[1 - (1+t)\hat{q}_{k-1}(t)\right]$$

having $k+1$ alternation points within $[\delta, 1]$. This is in turn equivalent to $1 - (1+t)\hat{q}_{k-1}(t) \propto V_k(t)$, which means that $\hat{p}_k$ defined by

$$\hat{p}_k(t) = \frac{V_k(t)}{V_k(-1)} = \frac{W_k(-t)}{W_k(1)}$$

is the unique minimizer of (10). The first claim then follows by noting that $W_k(1) = 1/(2k+1)$.

Finally, since $(t+1)^{1/2} |p_k(t)|$ is 0 at $t = -1$ for any $p_k \in \mathbb{P}_k$, $\hat{p}_k$ is also the unique minimizer for the limit case $\delta = -1$: indeed, any other candidate $\tilde{p}_k$ would have to attain a maximum of $(t+1)^{1/2} |\tilde{p}_k(t)|$ precisely at $t = -1$, which implies $\tilde{p}_k \equiv 0$, but then it clearly would not fulfill the $\tilde{p}_k(-1) = 1$ constraint. □

Remark 3.1. Note that this limit argument was not possible for the unweighted minimax problem and Chebyshev polynomials of the first kind, since the endpoints of the baseline interval $[-1, 1]$ are alternation points of every T_k.

After a trivial reparametrization to the $[0,\, \beta]$ interval, Lemma 3.2 allows us to characterize the solution to (9) for α sufficiently small, and even 0, thus removing the need for an estimate of the lower end of the fine spectrum of MA. In other words, for $\beta > 0$, we have that

$$\hat{W}_k \coloneqq \operatorname*{arg\,min}_{\substack{p_k \in \mathbb{P}_k \\ p_k(0) = 1}} \max_{\lambda \in [0,\, \beta]} \lambda^{1/2} |p_k(\lambda)|, \quad \hat{W}_k(\zeta) = \frac{1}{2k+1} W_k\left(1 - \frac{2}{\beta}\,\zeta\right). \tag{11}$$

Finally, based on the recurrence relationship

$$\begin{aligned} W_0(t) &= 1, \\ W_1(t) &= 2t + 1, \\ W_{n+1}(t) &= 2t\, W_n(t) - W_{n-1}(t), \quad n \geq 1, \end{aligned}$$

the same as with Chebyshev polynomials of the first kind but with a different linear polynomial, we can replicate the process from [9] to arrive at a two-term iteration. For simplicity, let us ignore preconditioning in our derivation. First of all, we may write the generic form of three-term Krylov iterations for $n \geq 0$ as

$$\begin{aligned} r_{n+1} &= \frac{1}{\gamma_n}(A r_n - \alpha_n r_n - \beta_{n-1} r_{n-1}), \\ x_{n+1} &= \frac{-1}{\gamma_n}(r_n + \alpha_n x_n + \beta_{n-1} x_{n-1}), \end{aligned}$$

with the consistency constraint that $\gamma_n = -(\alpha_n + \beta_{n-1})$, and understanding $r_{-1} \equiv 0$, $x_{-1} \equiv 0$, $\beta_{-1} = 0$. For the polynomial iteration based on Chebyshev polynomials of the fourth kind (11), we can write, for $n \geq 1$ and with $c = \beta/2$,

$$\begin{aligned}\hat{W}_{n+1}(t) = \frac{1}{2n+3} W_{n+1}\left(1 - \frac{t}{c}\right) = \\ = \frac{1}{2n+3}\left[2\left(1 - \frac{t}{c}\right) W_n\left(1 - \frac{t}{c}\right) - W_{n-1}\left(1 - \frac{t}{c}\right)\right] = \\ = 2\,\frac{2n+1}{2n+3}\left(1 - \frac{t}{c}\right)\hat{W}_n(t) - \frac{2n-1}{2n+3}\,\hat{W}_{n-1}(t),\end{aligned}$$

which leads us to

$$r_{n+1} = 2\,\frac{2n+1}{2n+3}\,r_n - \frac{2}{c}\,\frac{2n+1}{2n+3}\,Ar_n - \frac{2n-1}{2n+3}\,r_{n-1}, \quad n \geq 1,$$

and thus

$$\gamma_n = -\frac{2n+3}{2n+1}\,\frac{c}{2}, \quad \alpha_n = c, \quad \beta_{n-1} = -\frac{2n-1}{2n+1}\,\frac{c}{2}, \quad n \geq 1.$$

The remaining parameter values can be extracted from

$$r_1 = \frac{1}{3}\,W_1\left(I - \frac{1}{c}\,A\right) r_0 = r_0 - \frac{2}{3c}\,A\,r_0,$$

from which we can infer that $\gamma_0 = -\frac{3c}{2} = -\alpha_0$. At this point, following [9, Section 5], we can derive a two-term recurrence based on the parameters

$$\psi_n = \frac{\beta_n}{\gamma_n} = -\left(\frac{2n+1}{2n+3}\right)^2, \quad \omega_n = -\frac{1}{\gamma_n} = \frac{2n+1}{2n+3}\,\frac{2}{c}, \quad n \geq 0,$$

as

$$\begin{cases} v_n = r_n - \psi_{n-1} v_{n-1}, \\ x_{n+1} = x_n + \omega_n v_n, \\ r_{n+1} = b - A x_{n+1}. \end{cases}$$

for $n \geq 0$, understanding $\psi_{-1} = 0$ and $v_{n-1} \equiv 0$. Denoting $\mu_n = \frac{2n+1}{2n+3}$ for $n \geq 0$ and realizing that $\mu_{n+1} = (2 - \mu_n)^{-1}$, we can finally write the (preconditioned) polynomial iteration based on Chebyshev polynomials of the fourth kind as in Algorithm 3.

4 The Partition of Unity Method

For the experiments of this section, we will rely on the Partition of Unity Method of Schweitzer and Griebel [7, 23]. Let us succinctly introduce it. The method relies on sets of functions $\{\varphi_i\}_{i=1}^n$, $\varphi_i \colon \bar{\Omega} \to \mathbb{R}$, forming a *partition of unity* (PU) over some Lipschitz domain $\Omega \subset \mathbb{R}^d$, meaning that

$$0 \leq \varphi_i(x) \leq 1 \quad \forall i \in \{1, \ldots, n\}, \quad \text{and} \quad \sum_{i=1}^{n} \varphi_i(x) = 1, \quad \forall x \in \bar{\Omega},$$

which are used to generate discrete function spaces as

$$V^{\mathrm{PU}}(\Omega) = \sum_{i=1}^{n} \varphi_i V_i(\omega_i), \quad \omega_i \coloneqq \operatorname{supp}(\varphi_i),$$

Algorithm 3 Chebyshev iteration of the fourth kind

Require: A linear system $Ax = b$ with a s.p.d. matrix A, and an equally s.p.d. preconditioner M. An initial tentative solution x_0. An upper bound on the spectrum of MA, $\beta \geq \lambda_{max}(MA)$. A positive integer $k \geq 1$.

Initialize $r = b - Ax_0$, $p = Mr$.
Initialize $\mu = 1/3$.
Set $d = 4\beta^{-1}$.

Initialize $x = x_0 + d\mu\, p$

for $i = 2, \ldots, k$ **do**
    $r \leftarrow b - Ax$
    $p \leftarrow \mu^2\, p + Mr$
    $\mu \leftarrow (2 - \mu)^{-1}$
    $x \leftarrow x + d\mu\, p$
end for

return x

where each $V_i(\omega_i)$, called *local space*, is defined on the corresponding ω_i, all of them being "glued together" by the PU functions. Usually Ω and ω_i are dropped from the notation. The simplest choice for the local spaces V_i is to make them polynomial spaces of degree p_i (although more complicated spaces are naturally allowed), while the PU functions φ_i can be constructed to satisfy $\varphi_i \in C^k(\Omega)$ for any predefined $k \in \mathbb{N}$ (or be simply piecewise constant) [15]. If we denote $\{\vartheta_i^k\}_{k=1}^{m_i}$ a basis for each local space V_i,

$$\left\{\varphi_i \vartheta_i^k : \; i = 1, \ldots, n;\; k = 1, \ldots, m_i\right\}$$

is a global basis of V^{PU}, and thus any classical Galerkin problem of the type

$$u \in V^{\mathrm{PU}} \quad : \quad a(v, u) = \ell(v), \quad \forall v \in V^{\mathrm{PU}},$$

can be represented by a linear system $A\bar{u} = b$, where $u = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \bar{u}_{(i,k)} \varphi_i \vartheta_i^k$ and the stiffness matrix A and load vector b are given by

$$A_{(i,k),(j,l)} = a(\varphi_i \vartheta_i^k,\, \varphi_j \vartheta_j^l), \quad b_{(i,k)} = \ell(\varphi_i \vartheta_i^k).$$

In particular, A fits Definition 2.1 with the blocks $A_{ij} = (A_{(i,k),(j,l)}) \in \mathbb{R}^{m_i \times m_j}$. The sparsity of A is achieved through a particular construction of the PU $\{\varphi_i\}$, which in practice is constructed based on an *open cover* of the domain Ω,

$$\mathcal{C}(\Omega) = \{\omega_i\}_{i=1}^{n} \quad : \quad \Omega \subset \bigcup_{i=1}^{n} \omega_i, \quad \mathrm{supp}(\varphi_i) = \omega_i,$$

whose elements we call *patches*, which is based on a regular grid for a bounding box of Ω. In Fig. 1 we show such a grid and the resulting cover for $\Omega = (0, 1)^2$ and uniform refinement level 3 (which corresponds to $2^3 \cdot 2^3 = 64$ patches). A progressive refinement of such regular grids allows us to construct a *nonnested* sequence of function spaces

$$V_0^{\mathrm{PU}} \not\subset V_1^{\mathrm{PU}} \not\subset \cdots \not\subset V_{J-1}^{\mathrm{PU}} \not\subset V_J^{\mathrm{PU}},$$

where we can write each space as

$$V_l^{\mathrm{PU}} = \sum_{i=1}^{n_l} \varphi_i^{(l)} V_i^{(l)} \left(\omega_i^{(l)}\right), \quad \omega_i^{(l)} \coloneqq \operatorname{supp}\left(\varphi_i^{(l)}\right), \quad l = 0, \ldots, J.$$

This nonnestedness makes the prolongation operators $I_{l-1}^l : V_{l-1}^{\mathrm{PU}} \to V_l^{\mathrm{PU}}$ and the restriction operators $I_l^{l-1} : V_l^{\mathrm{PU}} \to V_{l-1}^{\mathrm{PU}}$ nontrivial to define. Fortunately, they are also not too difficult: we can define the prolongation operators via a global-to-local L^2-projection (see [23]), i.e.

$$\begin{aligned} I_{l-1}^l : V_{l-1}^{\mathrm{PU}} &\to V_l^{\mathrm{PU}} \\ u^{(l-1)} &\mapsto I_{l-1}^l u^{(l-1)} = \sum_{i=1}^{n_l} \varphi_i^{(l)} u_i^{(l)} \end{aligned}$$

where each $u_i^{(l)} \in V_i^{(l)}$, $i = 1, \ldots, n_l$, solves

$$\left\langle u_i^{(l)}, v \right\rangle_{L^2\left(\omega_i^{(l)}\right)} = \left\langle u^{(l-1)}, v \right\rangle_{L^2\left(\omega_i^{(l)}\right)}, \quad \forall v \in V_i^{(l)},$$

and the restriction operators can then be defined by transposition. For more details, we refer once again to [7, 15, 23].

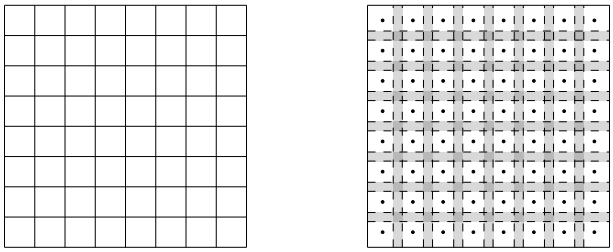


Figure 1: On the left, a regular grid for $\Omega = (0,1)^2$ at discretization level 3, and on the right, the associated open cover generated by stretching the square elements of the grid. Each black dot corresponds to a patch center, patch support boundaries are dashed, and the regions where multiple patches overlap are colored in gray.

4.1 Block-FSAI in the PUM framework

Let us finally mention the similarity between the block-FSAI preconditioner for the PUM and the *multilevel overlapping Schwarz* (MOS) smoother introduced for the PUM in [8]. Indeed, for a system $A\bar{u} = b$, we can understand the action of the FSAI preconditioner $F^\top S^{-1} F$ in two stages, the first one being $\bar{z} = S^{-1} F b$, and the second one $\bar{u} = F^\top \bar{z}$. The first step is equivalent to solving

$$A[\mathcal{P}_k, \mathcal{P}_k]\bar{w} = b[\mathcal{P}_k]$$

for every block-row index k and keeping only the entry from $\bar{w}$ corresponding to index k in each case, which becomes $\bar{z}_k$. In terms of the weak form and a PUM space

$V^{\mathrm{PU}} = \sum_{i=1}^n \varphi_i V_i$, this corresponds to solving the PDE in each *enlarged local space* $\hat{V}_k \coloneqq \sum_{i\in\mathcal{P}_k} \varphi_i V_i$, i.e.

$$w = \varphi_k w_k + \sum_{i\in\tilde{\mathcal{P}}_k} \varphi_i \tilde{w}_i \in \hat{V}_k \quad : \quad a(w,v) = \ell(v), \quad \forall v \in \hat{V}_k,$$

from which we keep only w_k (whose coefficients in the basis of V_k constitute $\bar{z}_k$). At the second step, every $\bar{z}_k$ contributes directly to the final corresponding entry $\bar{u}_k$, but also to the entries $\bar{u}[\tilde{\mathcal{P}}_k]$ via the correction $-A[\tilde{\mathcal{P}}_k, \tilde{\mathcal{P}}_k]^{-1} A[\tilde{\mathcal{P}}_k, \{k\}]\bar{z}_k$. This is equivalent to setting $u = \sum_{k=1}^n q_k \in V^{\mathrm{PU}}$ where each $q_k \in \hat{V}_k$ is of the form

$$q_k = \varphi_k w_k + \tilde{q}_k, \quad \tilde{q}_k \in \tilde{V}_k \coloneqq \sum_{i\in\tilde{\mathcal{P}}_k} \varphi_i V_i$$

(note that $\hat{V}_k = \varphi_k V_k + \tilde{V}_k$) and solves

$$a(q_k, v) = 0, \quad \forall v \in \tilde{V}_k.$$

Each $\tilde{q}_k$ can thus be understood as a correction to $\sum_{i=1}^n \varphi_i w_i$, which accounts for the fact that, for every $k' \in \tilde{\mathcal{P}}_k$, the solution step for $w_{k'}$ had not considered the local space V_k (because $k > k'$).

While the FSAI preconditioner allows for an adaptive construction of the enlarged local spaces, the enlarged local spaces of the MOS smoother

$$W_i \coloneqq \sum_{j\in\mathcal{N}_i} \varphi_j V_j, \quad \mathcal{N}_i \coloneqq \{j : \omega_i \cap \omega_j \neq \emptyset\},$$

are based on a geometric criterion and their size is fixed a priori. This makes it prohibitively expensive in certain scenarios, particularly in 3-dimensional problems (note that $\mathrm{card}(\mathcal{N}_i) = 3^d$ for homogeneously refined covers in $\mathbb{R}^d$). FSAI adaptivity, on the contrary, allows us to control the size of the enlarged local spaces, while also constructing them in an algebraic manner, based solely on information from the matrix itself.

5 Numerical experiments

Let us first describe the PUM spaces that we will use in our tests. We will use PU functions $\{\varphi_i\}$ satisfying the regularity requirements of each problem, with the local spaces V_i being formed by local polynomials of degree up to p, i.e. for $\Omega \subset \mathbb{R}^2$

$$V_i = \mathrm{span}\langle x^a y^b : \ a, b \geq 0, \ a + b \leq p\rangle,$$

and for $\Omega \subset \mathbb{R}^3$

$$V_i = \mathrm{span}\langle x^a y^b z^c : \ a, b, c \geq 0, \ a + b + c \leq p\rangle,$$

so that in particular $\dim(V_i) = \binom{p+d}{d}$ with $\Omega \subset \mathbb{R}^d$. In some cases, the polynomial degree of local spaces V_i for which $\mathrm{supp}(\varphi_i) \cap \partial\Omega \neq \emptyset$ will be increased to accomodate for higher derivatives appearing in the boundary integrals. For the multilevel iteration, we will construct the coarse operators via Galerkin products, i.e. $A_{l-1} = R_l A_l P_l$ with P_l being the prolongation matrix from level $l-1$ to level l, and $R_l = P_l^\top$.

With respect to the block-FSAI preconditioners, we will restrict ourselves to adaptively-constructed sparsity patterns, either nonnested or with 1 level of nesting. In the nested case, the adaptivity parameters will be the same for the two levels. For the adaptive algorithm, we choose the trivial tolerance parameter $\tau = 1$, allowing ourselves to modify the number of adaptive steps t_{max}. Additionally, as initial pattern $\mathcal{P}_0$ for Algorithm 1 we will always choose the diagonal pattern.

For the Chebyshev iteration, we will estimate the maximum eigenvalue of the FSAI-preconditioned matrices with the Lanczos algorithm [26], for an iteration number (and size of the resulting tridiagonal matrix) of 100, and we will multiply the obtained estimate by a factor of 1.01. Furthermore, in addition to the multilevel $V(k,k)$-cycle, following the work of Phillips and Fischer [22] we will also evaluate the performance of the non-symmetric $V(2k,0)$-cycle, justified by the optimality of the polynomial smoother.

Example 5.1. To begin with, we consider the biharmonic equation on the unit square $\Omega = (0,1)^2$, with essential boundary conditions on the whole boundary

$$\begin{cases} \Delta^2 u = f & \text{in } \Omega, \\ u = g_0 & \text{on } \partial\Omega, \\ \nabla u \cdot \mathbf{n} = g_1 & \text{on } \partial\Omega. \end{cases}$$

For a discrete space $V_n \subset H^2(\Omega)$, and following Nitsche's method for the weak imposition of boundary conditions, the weak formulation consists in finding $u \in V_n$ such that

$$a_n(v,u) = \ell_n(v), \quad \forall v \in V_n,$$

where

$$\begin{aligned} a_n(v,u) &= \int_\Omega \Delta u \Delta v \,\mathrm{d}\Omega + \int_{\partial\Omega} \left(\gamma_n^{(0)} uv + v\nabla(\Delta u)\cdot\mathbf{n} + u\nabla(\Delta v)\cdot\mathbf{n}\right)\mathrm{d}\Gamma + \\ &\quad + \int_{\partial\Omega} \left(\gamma_n^{(1)}(\nabla u\cdot\mathbf{n})(\nabla v\cdot\mathbf{n}) - \Delta u(\nabla v\cdot\mathbf{n}) - \Delta v(\nabla u\cdot\mathbf{n})\right)\mathrm{d}\Gamma, \\ \ell_n(v) &= \int_\Omega fv \,\mathrm{d}\Omega + \int_{\partial\Omega} g_0\left(\gamma_n^{(0)} v + \nabla(\Delta v)\cdot\mathbf{n}\right)\mathrm{d}\Gamma + \\ &\quad + \int_{\partial\Omega} g_1\left(\gamma_n^{(1)}(\nabla v\cdot\mathbf{n}) - \Delta v\right)\mathrm{d}\Gamma. \end{aligned} \tag{12}$$

The stabilization functions $\gamma_n^{(0)}$, $\gamma_n^{(1)}$ depend on V_n (hence our n subscript) and have to be "large enough" to ensure that the bilinear form is elliptic. We refer to [15] for the actual criteria and how we enforce them in the PUM.

For this simple domain and with homogeneous refinement, each PUM space V_l^{PU} consists of $2^l \cdot 2^l = 2^{2l}$ PU functions and their corresponding local spaces (recall Fig. 1 illustrating the case of level $l = 3$). Since $\dim(V_i) = \binom{p_i+2}{2}$, this means that $\dim(V_l^{\mathrm{PU}}) = \binom{p+2}{2}2^{2l}$ when using the same polynomial degree p for all local spaces.

For this initial test, we construct a multilevel sequence from coarsest level 2 up to finest level 7, and local polynomial spaces of degree 2. For patches intersecting the boundary, we increase the degree to 3 to accomodate for third-order derivatives in the boundary integrals. At the finest level, this yields 100,336 degrees of freedom. To measure the multilevel convergence rates, we consider the reference manufactured solution

$$\hat{u}(x,y) = \cos(2\pi x)\sin(2\pi y),$$

and obtain the corresponding discrete solution $u_{\mathrm{PU}} \in V^{\mathrm{PU}}$ with a direct solver. In order to measure the convergence rates of the multilevel iteration, we choose an initial iterate $u^{(0)} = u_{\mathrm{PU}} + e^{(0)}$, where the coefficient vector of $e^{(0)} \in V^{\mathrm{PU}}$ is generated with random entries in $(-1, 1)$, and normalized so that $\|e^{(0)}\|_{L^2(\Omega)} = 1$. We then measure convergence rates for the $L^2(\Omega)$ and the energy norms, which we can do with the corresponding error coefficient vectors $\bar{e}^{(n)} = \bar{u}^{(n)} - \bar{u}_{\mathrm{PU}}$, the mass matrix M and the stiffness matrix A as

$$\rho_{L^2} = \left(\frac{\langle M\bar{e}^{(n)}, \bar{e}^{(n)} \rangle}{\langle M\bar{e}^{(0)}, \bar{e}^{(0)} \rangle} \right)^{1/(2n)} = \langle M\bar{e}^{(n)}, \bar{e}^{(n)} \rangle^{1/(2n)}, \quad \rho_A = \left(\frac{\langle A\bar{e}^{(n)}, \bar{e}^{(n)} \rangle}{\langle A\bar{e}^{(0)}, \bar{e}^{(0)} \rangle} \right)^{1/(2n)},$$

for a number of iterations n. We iterate until $\|e^{(n)}\|_{L^2(\Omega)} = \langle M\bar{e}^{(n)}, \bar{e}^{(n)} \rangle^{1/2} < 10^{-8}$ or a maximum number of 50 iterations is reached. In our first case, we also look at convergence rates of the residual $r^{(n)} = b - A\bar{u}^{(n)}$ in the l^2-vector norm, which is equivalent to the A^2 norm of the error $\bar{e}^n$.

In Fig. 2 we provide the measured convergence rates for smoothing steps $k \in \{1, \dots, 10\}$ and FSAI adaptive steps $t_{max} \in \{4, \dots, 9\}$, with and without a further nested FSAI preconditioner. As we can see, all convergence rates depend similarly on t_{max} and k, and thus from now on (unless explicitly stated otherwise) we will only provide convergence rates in the energy norm, ρ_A.

We rely on nonnested FSAI smoothers for comparison of the $V(k, k)$ and the $V(2k, 0)$ cycles, whose results we present in Fig. 3. As expected, each nonsymmetric cycle outperforms the corresponding symmetric counterpart.

Let us further mention that, in the case of nonnested FSAI, the usual Richardson smoothing cycles have only shown systematic proper convergence for $t_{max} = 9$. The results for nested FSAI are presented in Fig. 4, revealing the superiority of Chebyshev smoothing also in this case.

Finally, we measure the convergence rates when choosing local spaces of polynomial degree $p \in \{3, 4, 5\}$. It suffices to compare the results for nonnested FSAI, which we do in Fig. 5 for $k \in [1, 5]$. As it can be clearly seen, the convergence rates do vary with the polynomial degree, but they decrease for increasing p (or remain similar for $p \in \{4, 5\}$ and $t_{max} \geq 5$).

Example 5.2. For our second example, we introduce anisotropy in the biharmonic equation via a fourth order tensor $\mathbb{D} \in \mathbb{R}^{2\times2\times2\times2}$ with the particular shape $\mathbb{D} = S \otimes S$, for $S \in \mathbb{R}^{2\times2}$ the s.p.d. matrix imposing a scaling with factor κ along the direction with angle θ, i.e.

$$S = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \kappa & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}.$$

The anisotropic biharmonic equation then reads

$$\begin{cases} S_{ij} S_{kl} u_{,ijkl} = f & \text{in } \Omega, \\ u = g_0 & \text{on } \partial\Omega, \\ \nabla u \cdot (S\mathbf{n}) = g_1 & \text{on } \partial\Omega, \end{cases}$$

and the bilinear and linear forms of its weak formulation following Nitsche's method consist in replacing Δw by $\mathrm{tr}(S\, H(w))$, $H(\cdot)$ being the hessian matrix of second order derivatives, and $\nabla w \cdot \mathbf{n}$ by $\nabla w \cdot (S\mathbf{n})$, for $w \in \{u, v\}$, in the isotropic equation Eq. (12).

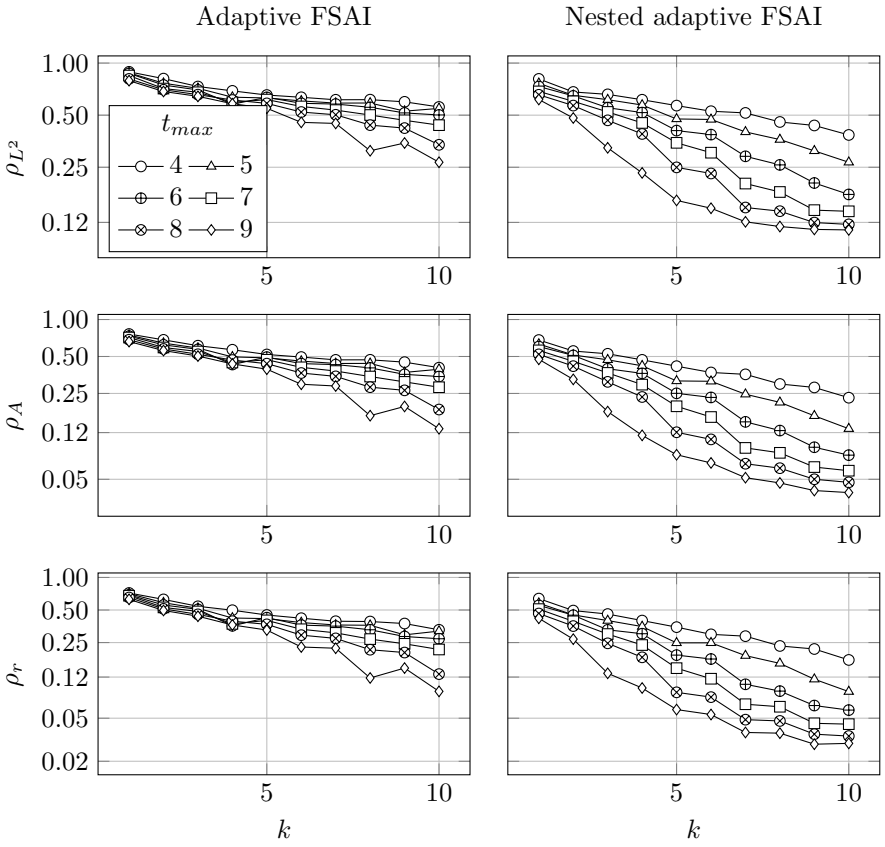


Figure 2: $V(k,k)$-cycle convergence rates in the $L^2(\Omega)$ and energy norms, as well as the l^2 norm of the residual, for Example 5.1, with local polynomial spaces of degree $p = 2$ (refined to $p = 3$ for patches intersecting the boundary).

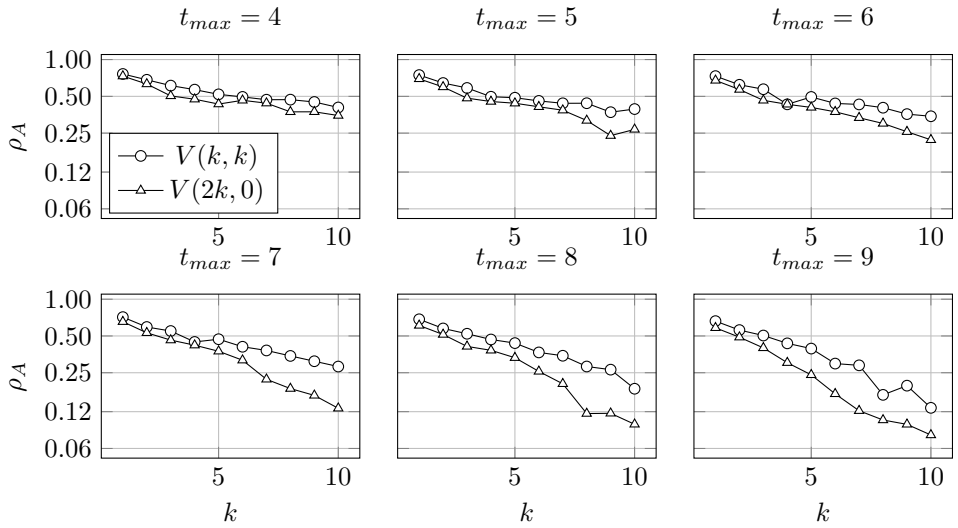


Figure 3: Comparison of the (Chebyshev-smoothing) $V(k,k)$-cycle and $V(2k,0)$-cycle convergence rates for nonnested adaptive FSAI smoothers, for Example 5.1.

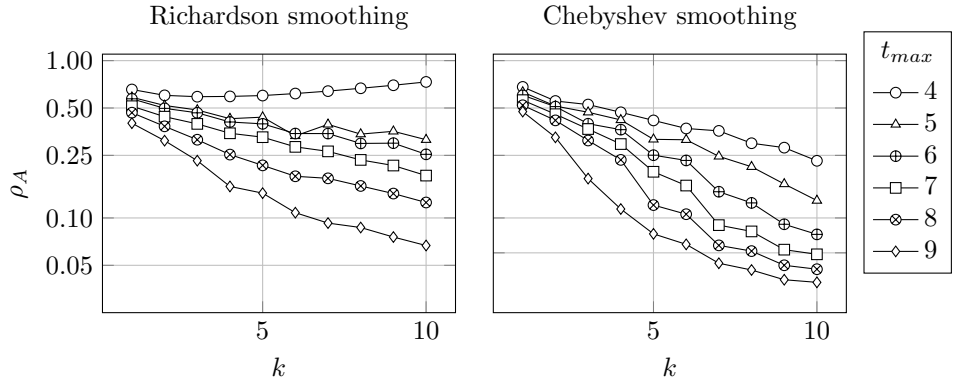


Figure 4: Comparison of the $V(k,k)$-cycle convergence rates for nested adaptive FSAI smoothers performed either with Richardson smoothing or with Chebyshev smoothing, for Example 5.1.

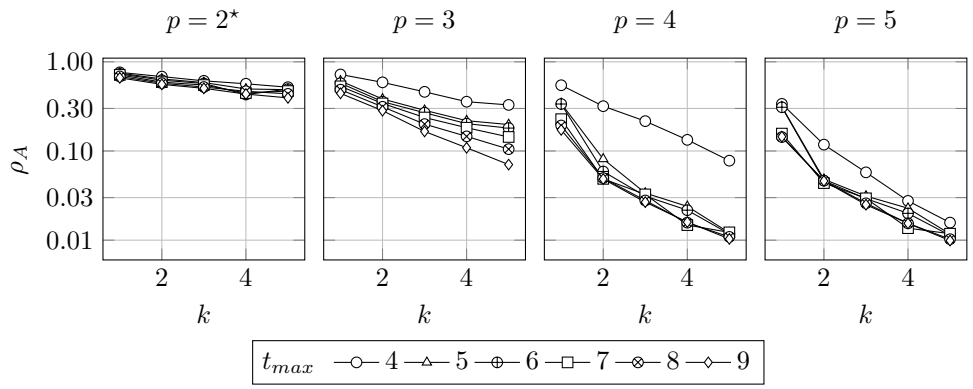


Figure 5: Comparison of the $V(k,k)$-cycle convergence rates for nonnested FSAI smoothers, with polynomial degrees $p \in \{2,3,4,5\}$ for the local spaces in Example 5.1. Note that, in the case $p = 2$, we use degree 3 for local spaces with support at the boundary.

In this anisotropic setting, we are particularly interested in visualizing the FSAI preconditioner adaptively generated by Algorithm 1. For that matter, we focus on the sparsity pattern $\mathcal{S}$ of the preconditioning matrix $F^\top S^{-1} F$ in the nonnested case (at discretization level 7 and for $p = 3$). For each row index k, the column indices in the corresponding row $\mathcal{S}_k$ can be understood as a "FSAI neighborhood" of associated patches in the homogeneously refined cover, $\mathcal{N}_k = \{\omega_j : j \in \mathcal{S}_k\}$. If we center every $\mathcal{N}_k$ around the corresponding ω_k and then superimpose all of them, we can visualize the "average neighborhood" as we do in Fig. 6 for $t_{max} = 6$. As we can see, the FSAI neighborhoods do extend, on average, along the anisotropy axis associated to κ-scaling.

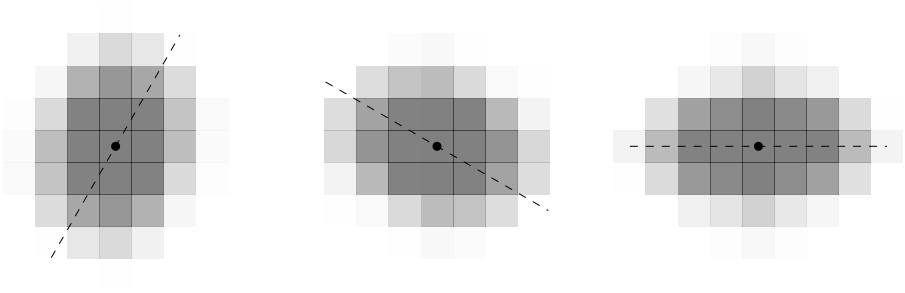


Figure 6: Average FSAI neighborhoods extracted from the rows of preconditioning matrix $F^\top S^{-1} F$ with $t_{max} = 6$, at discretization level 7, for the anisotropic problem from Example 5.2. The left image corresponds to $\theta = \frac{\pi}{3}$, the middle one to $\theta = -\frac{\pi}{6}$, and the right one to $\theta = 0$ (with the respective axes illustrated by dashed lines going through the neighborhood's center). In all three cases we use an anisotropic scaling $\kappa = 10$.

Example 5.3. For this new example, we consider again the biharmonic equation, but this time in the unit cube domain $\Omega = (0,1)^3$, where the weak formulation remains

unmodified. Given the larger size of local spaces with respect to the two-dimensional case, we restrict $t_{max} \leq 4$ for FSAI adaptivity and use 4 as the finest discretization level (keeping 2 as the coarsest level). In this case, each space consists of 2^{3l} local spaces. As reference manufactured solution we pick in this case

$$\hat{u}(x, y, z) = \cos(2\pi x)\sin(2\pi y)\sin(2\pi z).$$

This time we iterate until the $L^2(\Omega)$ norm of the error is below 10^{-7} or a maximum number of 100 iterations is reached. Given the cost of using large polynomial degrees in a 3-dimensional setting (local spaces are of size $\binom{p+3}{3}$, i.e. 20 for $p = 3$ and 35 for $p = 4$, and this size is multiplied by up to t_{max} for the blocks to be inverted at FSAI's setup stage), we provide only convergence rates for $p = 2$ (in Fig. 7, where as before we refine the polynomial degree to 3 for patches intersecting the boundary) and $p = 3$ (in Fig. 8). The combination of discretization level 4 with polynomial degree 3 yields $81,920$ degrees of freedom. As we can observe, the results for $p = 3$ are below those of $p = 2$ both for nested and nonnested FSAI. We remark the similar convergence rate histories for nested FSAI with $t_{max} = c$ and nonnested FSAI with $t_{max} = 2c$, pointing at the similarity of the FSAI F matrices when they are constructed in 2 levels with c non-zero blocks per row, or in a single level with $2c$ non-zero blocks per row.

Finally, in this case we choose $p = 3$ and nested FSAI for the comparison with Richardson smoothing and with nonsymmetric cycles, which we provide simultaneously in Fig. 9. As before, we clearly observe the superiority of Chebyshev smoothing with respect to Richardson smoothing, and of non-symmetric cycles with respect to symmetric ones.

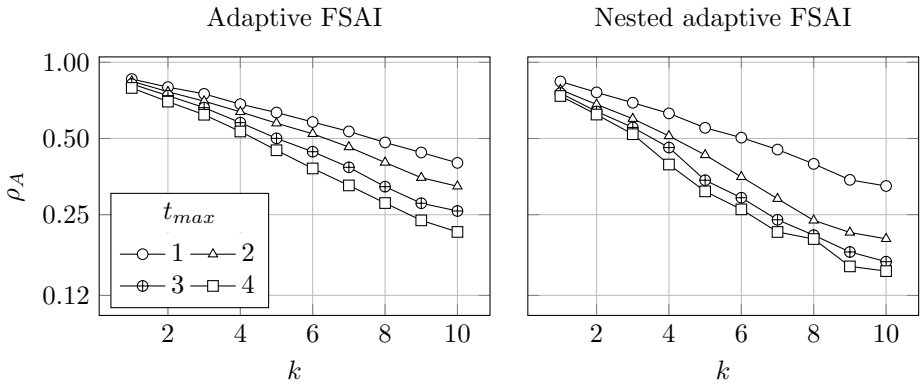


Figure 7: $V(k, k)$-cycle convergence rates for Example 5.3, with local polynomial spaces of degree $p = 2$ (refined to $p = 3$ for patches intersecting the boundary).

Example 5.4. For our last example we further increase the order of the problem and aim to solve the *triharmonic* equation

$$\begin{cases} -\Delta^3 u = f & \text{in } \Omega, \\ u = g_0 & \text{on } \partial\Omega, \\ \nabla u \cdot \mathbf{n} = g_1 & \text{on } \partial\Omega, \\ \Delta u = g_2 & \text{on } \partial\Omega. \end{cases}$$

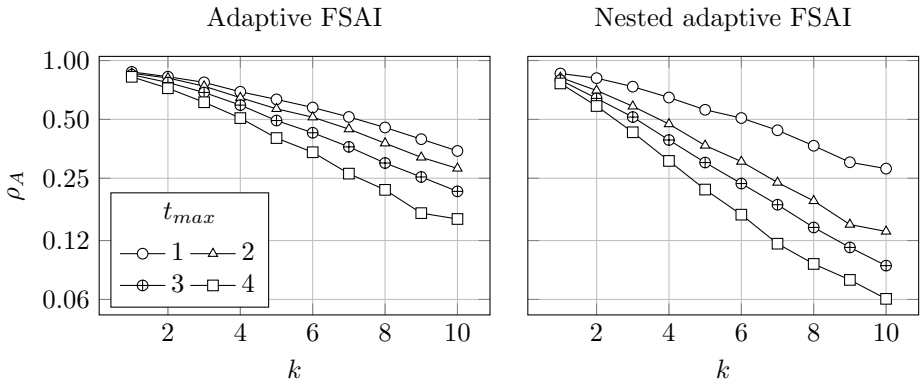


Figure 8: $V(k,k)$-cycle convergence rates for Example 5.3, with local polynomial spaces of degree $p = 3$ (compare with Fig. 7 to assess the impact of p-refinement).

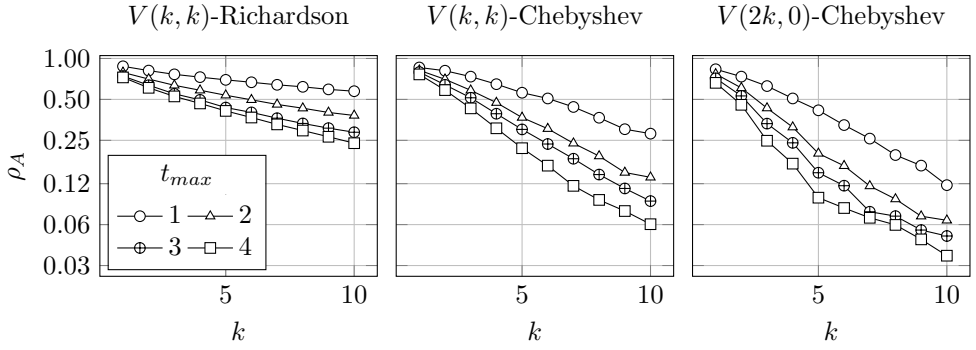


Figure 9: Comparison of Richardson and Chebyshev $V(k,k)$-cycle, and Chebyshev $V(2k,0)$-cycle convergence rates, for Example 5.3, with nested FSAI smoothers and local polynomial spaces of degree $p = 3$.

For a discrete space $V_n \subset H^3(\Omega)$, Nitsche's method in this case yields a weak formulation with bilinear and linear forms

$$\begin{aligned}
a_n(v,u) = &\int_\Omega \nabla(\Delta u)\cdot\nabla(\Delta v)\,\mathrm{d}\Omega + \int_{\partial\Omega}\left(\gamma_n^{(0)}uv - v\nabla(\Delta^2 u)\cdot\mathbf{n} - u\nabla(\Delta^2 v)\cdot\mathbf{n}\right)\mathrm{d}\Gamma + \\
&+ \int_{\partial\Omega}\left(\gamma_n^{(1)}(\nabla u\cdot\mathbf{n})(\nabla v\cdot\mathbf{n}) + \Delta^2 u(\nabla v\cdot\mathbf{n}) + \Delta^2 v(\nabla u\cdot\mathbf{n})\right)\mathrm{d}\Gamma + \\
&+ \int_{\partial\Omega}\left(\gamma_n^{(2)}\Delta u\Delta v - \Delta v\nabla(\Delta u)\cdot\mathbf{n} - \Delta u\nabla(\Delta v)\cdot\mathbf{n}\right)\mathrm{d}\Gamma, \\
\ell_n(v) = &\int_\Omega fv\,\mathrm{d}\Omega + \int_{\partial\Omega} g_0\left(\gamma_n^{(0)}v - \nabla(\Delta^2 v)\cdot\mathbf{n}\right)\mathrm{d}\Gamma + \\
&+ \int_{\partial\Omega} g_1\left(\gamma_n^{(1)}(\nabla v\cdot\mathbf{n}) + \Delta^2 v\right)\mathrm{d}\Gamma + \int_{\partial\Omega} g_2\left(\gamma_n^{(2)}\Delta v - \nabla(\Delta v)\cdot\mathbf{n}\right)\mathrm{d}\Gamma.
\end{aligned}$$

In this case, we choose 5 as the finest discretization level, and $p \in \{4,5\}$ for the local spaces (for $p=4$, we refine the local spaces with support at the boundary to $p=5$). With respect to FSAI adaptivity, we restrict $t_{max} \in [6,9]$. In Fig. 10, we provide the measured convergence rates for the case $p=4$, and in Fig. 11, those for $p=5$. We finally choose the $p=4$ case with nested FSAI smoothers for the comparison with nonsymmetric cycles, whose results we provide in Fig. 12, showing once again the superiority of the $V(2k,0)$-cycles. In all cases, we iterate until the $L^2(\Omega)$ norm of the error is below 10^{-7} or a maximum number of 100 iterations is reached. We note that, for this problem, Richardson smoothing did not yield a convergent cycle for any of our considered FSAI settings in the $p=4$ case.

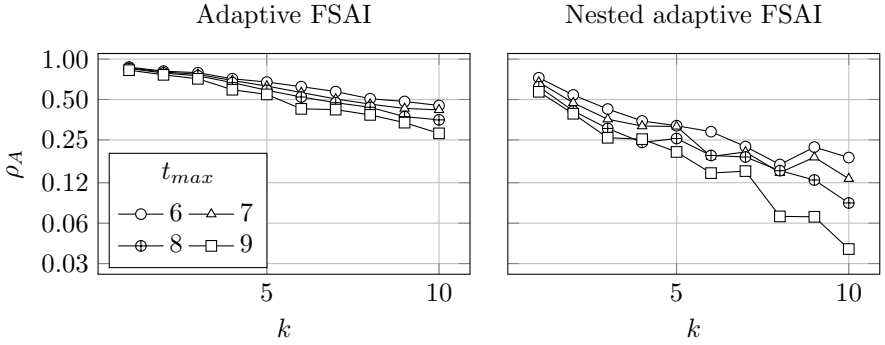


Figure 10: $V(k,k)$-cycle convergence rates for Example 5.4, with local polynomial spaces of degree $p=4$ (refined to $p=5$ for patches intersecting the boundary).

6 Conclusions

We have explored the flexibility of FSAI smoothers with respect to adaptivity and nestedness, relying on PUM discretizations of the biharmonic and triharmonic equations. We have shown how their smoothing capability improves with increasing density of the preconditioning matrix, which we achieved either by allowing more non-zero entries per row in the adaptive algorithm, or by nesting an additional FSAI preconditioner into the first one (or by a combination of both). We have shown that their

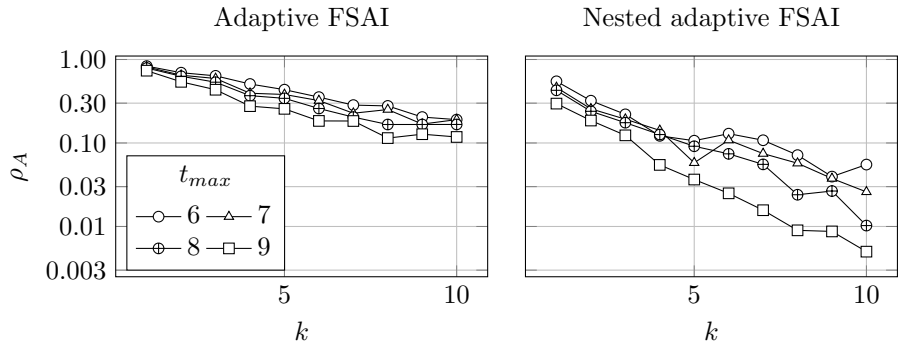


Figure 11: $V(k,k)$-cycle convergence rates for Example 5.4, with local polynomial spaces of degree $p = 5$ (compare with Fig. 10 to assess the impact of p-refinement).

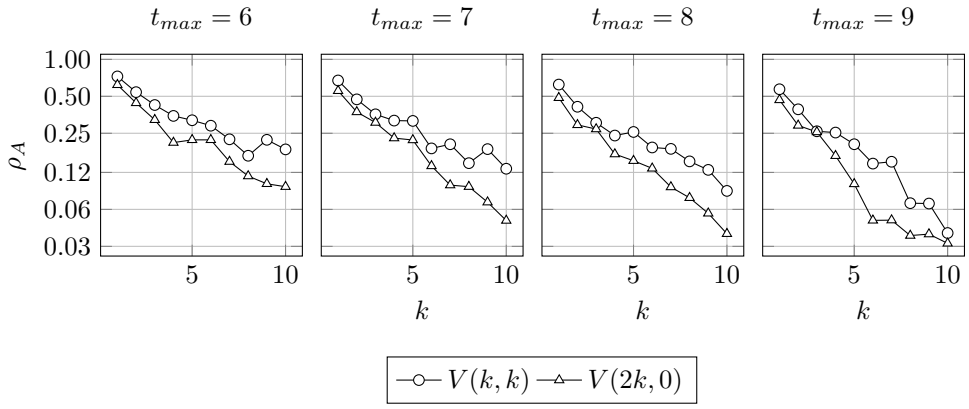


Figure 12: Comparison of the $V(k,k)$ and $V(2k,0)$-cycle convergence rates for Example 5.4, with nested adaptive FSAI smoothers and local polynomial spaces of degree $p = 4$ (refined to $p = 5$ for patches intersecting the boundary).

effectiveness increases with the polynomial degree of the discrete space, and that the adaptive pattern construction allows to capture anisotropies in the PDE.

We have also shown that smoothing based on the Chebyshev iteration of the fourth kind yields better convergence rates than the usual Richardson smoothing, and, for a limited number of smoothing steps, even allows convergence in cases where Richardson smoothing does not. Additionally, we have confirmed that, as pointed out already in the literature, non-symmetric $V(2k, 0)$ cycles yield faster convergence rates than their symmetric $V(k, k)$ counterparts. Nevertheless, we note that when using the multilevel iteration as a preconditioner within some iterative solver, the use of $V(2k, 0)$ cycles prevents the use of a standard conjugate gradient solver, which has to be replaced by some solver accepting a non-symmetric preconditioner (e.g. BiCGStab).

With respect to our algorithms, we have introduced a simple but effective adaptive construction for the FSAI preconditioner, for matrices with a certain block structure. This includes the matrices arising from a PUM discretization, but could be extended to different scenarios. Additionally, we have provided a new formulation of the Chebyshev iteration of the fourth kind, which in our opinion is even simpler than the one originally given by Lottes [17]. We agree with him that this new Chebyshev iteration deserves more wide-spread recognition in contrast to the usual Richardson smoothing, and we hope that our work will make a contribution to that, even if a small one.

References

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *Journal of Computational Physics*, 188(2):593–610, July 2003. ISSN 0021-9991. https://doi.org/10.1016/s0021-9991(03)00194-3.

[2] A. R. Alimov and I. G. Tsar'kov. *Chebyshev Alternation Theorem. Haar's and Mairhuber's Theorems*, pages 19–46. Springer International Publishing, 2021. ISBN 9783030909512. https://doi.org/10.1007/978-3-030-90951-2_2.

[3] D. Arndt, W. Bangerth, M. Bergbauer, M. Feder, M. Fehling, J. Heinz, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, B. Turcksin, D. Wells, and S. Zampini. The deal.II Library, Version 9.5. *Journal of Numerical Mathematics*, 31(3):231–246, Aug. 2023. https://doi.org/10.1515/jnma-2023-0089.

[4] S. F. Ashby, T. A. Manteuffel, and J. S. Otto. A Comparison of Adaptive Chebyshev and Least Squares Polynomial Preconditioning for Hermitian Positive Definite Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 13(1):1–29, Jan. 1992. ISSN 2168-3417. https://doi.org/10.1137/0913001.

[5] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Multigrid Smoothers for Ultraparallel Computing. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, Jan. 2011. ISSN 1095-7197. https://doi.org/10.1137/100798806.

[6] E. W. Cheney. *Introduction to Approximation Theory*. AMS Chelsea Publishing. American Mathematical Society, Providence, RI, 2 edition, Oct. 1998.

[7] M. Griebel and M. A. Schweitzer. A Particle-Partition of Unity Method–Part III: A Multilevel Solver. *SIAM J. Sci. Comput.*, 24(2):377–409, Feb. 2002. ISSN 1064-8275. https://doi.org/10.1137/S1064827501395252.

[8] M. Griebel, P. Oswald, and M. A. Schweitzer. *A Particle-Partition of Unity Method Part VI: A p-robust Multilevel Solver*, pages 71–92. Springer Berlin Heidelberg, 2005. ISBN 9783540230267. https://doi.org/10.1007/3-540-27099-x_5.

[9] M. H. Gutknecht and S. Röllin. The Chebyshev iteration revisited. *Parallel Computing*, 28(2):263–283, Feb. 2002. ISSN 0167-8191. https://doi.org/10.1016/s0167-8191(01)00139-9.

[10] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer Berlin Heidelberg, 1985. ISBN 9783662024270. https://doi.org/10.1007/978-3-662-02427-0.

[11] C. Janna and M. Ferronato. Adaptive Pattern Research for Block FSAI Preconditioning. *SIAM Journal on Scientific Computing*, 33(6):3357–3380, Jan. 2011. ISSN 1095-7197. https://doi.org/10.1137/100810368.

[12] C. Janna and A. Franceschini. Nesting Approximate Inverses for Improved Preconditioning and Algebraic Multigrid Smoothing. *SIAM Journal on Matrix Analysis and Applications*, 46(1):393–415, Feb. 2025. ISSN 1095-7162. https://doi.org/10.1137/24m1679847.

[13] C. Janna, M. Ferronato, and G. Gambolati. A Block FSAI-ILU Parallel Preconditioner for Symmetric Positive Definite Linear Systems. *SIAM Journal on Scientific Computing*, 32(5):2468–2484, Jan. 2010. ISSN 1095-7197. https://doi.org/10.1137/090779760.

[14] C. Janna, M. Ferronato, F. Sartoretto, and G. Gambolati. FSAIPACK: A Software Package for High-Performance Factored Sparse Approximate Inverse Preconditioning. *ACM Transactions on Mathematical Software*, 41(2):1–26, Feb. 2015. ISSN 1557-7295. https://doi.org/10.1145/2629475.

[15] P. Jiménez Recio and M. A. Schweitzer. A Partition of Unity construction of the stabilization function in Nitsche's method for variational problems. *Computer Methods in Applied Mechanics and Engineering*, 426:117002, June 2024. ISSN 0045-7825. https://doi.org/10.1016/j.cma.2024.117002.

[16] I. E. Kaporin. New convergence results and preconditioning strategies for the conjugate gradient method. *Numerical Linear Algebra with Applications*, 1(2):179–210, Mar. 1994. ISSN 1099-1506. https://doi.org/10.1002/nla.1680010208.

[17] J. Lottes. Optimal polynomial smoothers for multigrid V-cycles. *Numerical Linear Algebra with Applications*, 30(6), June 2023. ISSN 1099-1506. https://doi.org/10.1002/nla.2518.

[18] J. Mason and D. C. Handscomb. Chebyshev Polynomials, Sept. 2002.

[19] J. C. Mason. The minimality properties of Chebyshev polynomials and their lacunary series. *Numerical Algorithms*, 38(1-3):61–78, Mar. 2005. ISSN 1572-9265. https://doi.org/10.1007/bf02810616.

[20] V. A. Paludetto Magri, A. Franceschini, and C. Janna. A Novel Algebraic Multigrid Approach Based on Adaptive Smoothing and Prolongation for Ill-Conditioned Systems. *SIAM Journal on Scientific Computing*, 41(1):A190–A219, Jan. 2019. ISSN 1095-7197. https://doi.org/10.1137/17m1161178.

[21] K. B. Petersen and M. S. Pedersen. The Matrix Cookbook, nov 2012. URL http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html. Version 20121115.

[22] M. Phillips and P. Fischer. Optimal Chebyshev Smoothers and One-sided V-cycles, 2023. URL https://arxiv.org/abs/2210.03179.

[23] M. A. Schweitzer. *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*. Springer Berlin Heidelberg, 2003. ISBN 9783642593253. https://doi.org/10.1007/978-3-642-59325-3.

[24] M. Sedlacek. *Sparse approximate inverses for preconditioning, smoothing, and regularization*. PhD thesis, Technische Universität München, 2012.

[25] J. Sogn and S. Takacs. Robust multigrid solvers for the biharmonic problem in isogeometric analysis. *Computers and Mathematics with Applications*, 77(1): 105–124, Jan. 2019. ISSN 0898-1221. https://doi.org/10.1016/j.camwa.2018.09.017.

[26] H. A. van der Vorst. A Generalized Lanczos Scheme. *Mathematics of Computation*, 39(160):559, Oct. 1982. ISSN 0025-5718. https://doi.org/10.2307/2007333.