Multigrid methods for stress-based contact problems in linear elasticity

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Gabriele Rovi

under the supervision of
Prof. Rolf Krause

June 2021

Dissertation Committee

Prof. Christian Hesch	Universität Siegen, Germany
Prof. Gerhard Starke	Universität Duisburg-Essen
Prof. Michael Multerer	Università della Svizzera italiana, Switzerland
Prof. Cesare Alippi	Università della Svizzera italiana, Switzerland

Dissertation accepted on 01 June 2021

Research Advisor	PhD Program Director
Prof. Rolf Krause	**Prof. Silvia Santini**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Gabriele Rovi
Lugano, 01 June 2021

To my parents

Whereof one cannot speak,
thereof one must be silent.
Ludwig Wittgenstein

Abstract

Problems of contact mechanics arise in many engineering applications. The contact conditions make the problem constrained and are the main challenge to be tackled. Different weak forms can be used for the modeling of contact problems. The primal weak form solves for the only displacement $\mathbf{u} \in \mathbf{H}_1$. After the finite element (FE) discretization, the problem can be solved by means of the monotone multigrid (MMG) method, a solver for constrained problems with optimal complexity. However, the primal formulation is affected by locking and can compute the stress $\boldsymbol{\sigma}$, a physical quantity of primary interest, only by means of differentiation of the displacement.

In contrast to the primal formulation, the stress-based formulations are not affected by locking, as they are based on the stresses as main unknowns. Thus they are quite attractive for nearly incompressible and incompressible materials. In this thesis, we study the first-order system least squares (FOSLS) and the dual formulations for the Signorini problem, i.e., a unilateral contact problem. In the first approach, an energy functional subject to only box-constraints has to be minimized. In the second case, also global equality constraints must be enforced and an LBB (Ladyzhenskaya-Babuška-Brezzi) condition must be satisfied. This thesis extends the MMG method for the primal formulation to the stress-based formulations applied to the Signorini problem for nearly incompressible and incompressible materials. However, in the stress-based formulations, the stress $\boldsymbol{\sigma}$ belongs to the space $\mathbf{H}_{\text{div}}$ and therefore special care is needed for the finite element discretization and the corresponding solution methods. Linear multigrid methods which work for $\mathbf{H}_{\text{div}}$ spaces have been already investigated. To the author's knowledge, this thesis is the first attempt to generalize the MMG from the primal formulation to the stress-based ones. To this purpose, we generalize the Arnold-Falk-Winther smoother patch smoother for $\mathbf{H}_{\text{div}}$-regular problems, so that the contact constraints are solved locally. We show several numerical experiments that illustrate the performance of our new multigrid method for both the FOSLS and the dual cases.

Contents

Figures

Tables

Chapter 1

Introduction

Contact problems arise in many engineering applications. From the physical point of view, the two main unknowns involved in problems of mechanics are the displacement field $\mathbf{u}$ and the stress field $\boldsymbol{\sigma}$. However, when it comes to the weak formulations, only one of the two can be the unknown to be found, while the other one can be post-processed. For example, the primal formulation is the most known weak form for problems of mechanics. It solves for the only displacement variable, while the stress has to be computed a posteriori, once the displacement is known. Furthermore, in this case, the functional of the formulation becomes unbounded for nearly incompressible or incompressible materials. Nevertheless, in many engineering problems, it is necessary to have a good approximation of the stresses for nearly incompressible or incompressible materials. One of the most common examples can be found in the field of civil engineering. Indeed, for our safety, the buildings have to be almost incompressible. Furthermore, the computation of the forces which are generated is crucial for determining if the project of the structure satisfies the standard or not. Last but not least, all the components of a building come into contact with each other, and thus the particular case of contact mechanics is worth to be examined. Contact mechanics give rise, even in linear elasticity and for rigid obstacles, to constrained problems. Indeed if the body of interest cannot penetrate the obstacle, linear inequality constraints naturally arise. We refer the reader to Kikuchi and Oden [1988] for further reading on this topic.

Specific weak forms can be used to solve unilateral contact problems. The standard primal weak formulation computes only the displacement $\mathbf{u}$, while the stress $\boldsymbol{\sigma}$ must be post-processed. On the other hand, the stress-based formulations always treat the stress $\boldsymbol{\sigma}$ as the main variable. In particular, in this thesis, we examine the first-order system least-squares (FOSLS) and the dual formulations for the linear elasticity problem and the frictionless unilateral contact problem, also known as the Signorini problem. On one hand, in the FOSLS formulation, $\mathbf{u}$ and $\boldsymbol{\sigma}$ are both main variables. On the other hand, in the dual formulation, the stress $\boldsymbol{\sigma}$ is the main variable, while the displacement $\mathbf{u}$ and the rotation $\boldsymbol{\theta}$ take the role of Lagrange multipliers for enforcing respectively the equilibrium condition and the symmetry of the stress condition. There are two main advantages of the stress-based formulations. First, they enable the automatic treatment of nearly incompressible and incompressible materials. Secondly, they permit direct access to the stress, which does not have to be post-processed and which can be exploited in further non-linear generalizations, like friction, plasticity, and so on. However, if in the primal formulation the displacement $\mathbf{u}$ belongs to the well know Sobolev space $\mathbf{H}^1$, in the stress-based formulations the stress $\boldsymbol{\sigma}$ belongs to the larger space $\mathbf{H}_{\text{div}}$. This is not a minor detail and will influence the analysis not only of the continuum problem but also of the corresponding finite elements (FE) discretization and solution method.

The FOSLS formulations have been widely studied. Bochev, Pavel B. and Gunzburger, Max D. [2009] is an exhaustive book that analyzes the FOSLS method from both abstract and practical points of view. In particular, the success of the FOSLS approach is given by the use of the FOSLS functional as an a posteriori error estimator, like in Cai, Zhiqiang and Lazarov, R. and Manteuffel, Thomas A. and McCormick, Stephen F. [1994], Berndt et al. [1997]. Furthermore, boundary conditions in the FOSLS formulation can be enforced essentially or weakly. A multilevel boundary functional is discussed in Starke [1999]. The case of discontinuous coefficients is examined in Berndt et al. [2005] and in Starke, Gerhard [2000], while generalizations to mechanics can be found in Yang, Suh-Yuh and Liu, Jinn-Liang [1997], Cai, Zhiqiang and Starke, Gerhard [2004], Müller, Benjamin [2015]. The relation between the FOSLS and the dual formulations for the Poisson problem is studied in Brandts et al. [2006]. The same kind of analysis is extended to the linear elastic problem in Starke et al. [2011]. Generalizations of the linear elastic problem to plasticity are discussed in Starke [2007] and in Starke [2009]. The Signorini problem for the FOSLS linear elasticity is examined in Krause, Rolf and Müller, Benjamin and Starke, Gerhard [2017]. For the dual formulation in linear elasticity, we can refer the reader to Brezzi and Fortin [2012] and Kober, Bernhard [2017]. The Signorini problem for the dual linear elasticity is discussed in Kober, Bernhard [2020], where also the case of Coulomb friction is taken into consideration. In Krause, Rolf and Müller, Benjamin and Starke, Gerhard [2017] and Kober, Bernhard [2020], a posteriori error estimators are defined as well. For the general FOSLS problems, the FOSLS functional to be minimized can be used as an error estimator. For the dual contact mechanics, the displacement, which is only a Lagrange multiplier, needs to be properly reconstructed before the estimation of the error, see again Kober, Bernhard [2020].

However, to the author's knowledge, no proper discussion on optimal solvers for both the linear elastic and the Signorini problems of the FOSLS and the dual formulations has been carried out. In this thesis, we will study the multigrid (MG) method for all these cases. Indeed multigrid methods are well known iterative solvers for their optimal complexity, i.e., their rate of convergence is independent of the dimension of the problem. To reach this goal, MG methods tackle the frequency components of the error simultaneously, by introducing a proper hierarchy of FE spaces. For each level of the hierarchy, certain frequency components of the error are smoothed down, for example, by means of the Gauss-Seidel method or the conjugate gradient method. For further reading on linear multigrid methods, see Xu, Jinchao [1992], Chen, Zhangxin [1994], Xu, Jinchao [1996], Bank, Randolph E. and Yserentant, Harry [2010]. However, as already mentioned, the Signorini problem is constrained. Solvers that tackle inequality constraints can be found in Hoppe, Ronald HW [1987], Gelman, E. and Mandel, J. [1990], Dostál, Zdenek [1997]. To recover optimality, a multigrid method must be considered. The monotone multigrid (MMG) method is an optimal solver that sequentially minimizes the energy employing fine and coarse corrections which do also satisfy the constraints. It has been investigated in Badea [2002], Badea [2014], Badea and Krause [2012], Kornhuber, Ralf and Krause, Rolf [2001], Kornhuber, Ralf and Krause, Rolf and Sander, O. and Deuflhard, P. and Ertel, S. [2008], Kornhuber, Ralf [1994], Krause [2009], Krause, Rolf and Rigazzi, Alessandro and Steiner, Johannes [2016]. However, all these papers analyze the only primal case. We aim to generalize the MMG method to the dual and the FOSLS formulations for the Signorini problem.

As an example of the multilevel method for the FOSLS case, we can mention Starke, Gerhard [2000], where multigrid is applied to variably saturated subsurface flows. On the other hand, efficient multigrid methods exist for the saddle point system arising from the discretization of the dual formulation for plane linear elasticity for nearly incompressible materials. However, the finite element discretizations differ from the one proposed in Brezzi and Fortin [2012] which holds, in 2D and 3D, for nearly incompressible and incompressible materials. For example, in Klawonn and Starke [2004] and in Pasciak, Joseph E. and Wang, Yanqiu [2006], for the discretization of the stress, the 2D PEERS elements and the 2D Arnold-Winther elements are respectively used, and only nearly incompressible materials are examined.

The main issue of the FOSLS and the dual forms is that $\boldsymbol{\sigma} \in \mathbf{H}_{\text{div}}$ and, in order to smooth the divergence-free components of the error, more sophisticated strategies are required. In Hiptmair, Ralf [1997], the Helmholtz decomposition is exploited and divergence-free corrections belonging to a specific potential space are computed. For generalizations of this idea from geometric to algebraic multigrid methods, see Kolev and Vassilevski [2012], Hiptmair, Ralf and Xu, Jinchao [2007], Xu, Jinchao and Chen, Long and Nochetto, Ricardo H. [2009]. The Arnold-Falk-Winther smoother, introduced in Arnold Douglas [1998], Arnold et al. [2000], Arnold et al. [1997], computes divergence-free corrections by enlarging the subdomains of the Gauss-Seidel smoother to patch subdomains. For sure, in mechanics, this approach is easier to manage. However, the Arnold-Falk-Winther patch smoother has been developed only for linear problems, while the Signorini problem is constrained. In this thesis, a novel non-linear patch smoother is designed in this direction. In particular, here we present a list of the contributions of this thesis:

- We extend the Arnold-Falk-Winther patch smoother to the Signorini problem for both the FOSLS and the dual formulations. In particular, a monolithic approach is exploited. On one hand, the patch subspaces for the FOSLS formulation tackle both $\mathbf{u}$ and $\boldsymbol{\sigma}$. On the other hand, for the dual formulation, the patch subspaces tackle the stress $\boldsymbol{\sigma}$, the displacement $\mathbf{u}$, and the rotation $\boldsymbol{\theta}$ altogether. However, in contrast to the problems considered in the papers cited above, the Signorini problem is constrained. Thus, the MMG we develop must be able not only to tackle divergence-free functions but also the constraints of the given formulation. Therefore the smoother must solve for local constrained problems on patches.

- Concerning the FOSLS formulation, the Arnold-Falk-Winther patch smoother can be easily defined. However, the truncation of the basis functions, defined in Kornhuber, Ralf [1994] and in Kornhuber, Ralf [1996] to accelerate the convergence of the MMG method for the primal case, is not effective here. This bad performance leads back to the presence of mixed essential boundary conditions.

- Concerning the dual formulation, the truncation of the basis functions works as in the primal formulation. Nevertheless, the Arnold-Falk-Winther patch smoother can be defined with different local boundary conditions: Neumann, Dirichlet, and Robin. By imposing Neumann boundary conditions on each patch, rigid body motions must be somehow removed. All strategies proposed in this direction do not seem to make the smoother properly working. On the other hand, the Robin conditions of parameter

$\alpha \geq 0$ performs better. If $\alpha = 0$, the Dirichlet conditions are recovered. However, for the Signorini problem, convergence is attained only if pure Robin conditions are enforced. Furthermore, if α is chosen properly, the MMG convergence rate is optimal.

- Since the optimal value for the parameter α is unknown, we examine different algorithms for its dynamic computation. In particular, three main scenarios have been studied. First, the parameter α is changed from patch to patch. Second, the parameter α is constant for a given smoothing step. Third, the parameter α is constant for an entire V-cycle. The last option performs the best, but the algorithm for the update of α has to be properly defined.
- Finally, we do also provide a proof for the coercivity of the bilinear form of the FOSLS functional for the Signorini problem. We show that coercivity can be ensured only if the constants of the FOSLS functional are properly chosen.

In chapter 2, we introduce the notation and preliminary well known definitions and results for both H^1 and H_{div} spaces. In chapter 3, the linear elasticity problem is defined in both its strong and weak forms. In particular, we treat the primal, the dual, and the FOSLS weak formulations. In chapter 4, we generalize both the strong linear elasticity problem and the three corresponding weak formulations for the frictionless unilateral contact problem, also known as the Signorini problem. In chapter 5, we discretize all these formulations by means of the FE method. In particular, the Lagrangian and the Raviart-Thomas FE spaces will be discussed in detail. In chapter 6 the MMG method is introduced. We discuss the algorithm and its main ingredients, i.e., the interpolation operator, the monotone restriction operator, the truncation of the basis functions, and the smoother. We will discover that, for the primal formulation approaching the incompressible limit, the MMG method is negatively affected. Thus it is necessary to extend the MMG to the stress-based formulations, with functionals that are still bounded in the incompressible limit. In contrast to the primal formulation, the interpolation operator and the smoother must be modified accordingly. In particular, as already mentioned, we generalize the Arnold-Falk-Winther smoother to a full monolithic and locally non-linear approach. The numerical experiments for the FOSLS formulation are presented in chapter 7. It is shown that the convergence of the MMG can be negatively influenced by the simultaneous presence of essential boundary conditions for both the stress $\boldsymbol{\sigma}$ and the displacement $\mathbf{u}$. Standard truncation for both $\mathbf{u}$ and $\boldsymbol{\sigma}$ can produce coarse spaces that are not rich enough to capture all the coarse divergence-free functions. For this reason, the MMG for the dual formulation has been widely studied in section 8. In this case, truncation does not create any problem, since the only variable to be truncated is $\boldsymbol{\sigma}$. However, the patch smoother can be defined with different local boundary conditions, i.e., Neumann, Dirichlet, and Robin conditions. We show that the Robin conditions perform better than the Neumann or the Dirichlet ones. However a proper tuning of the Robin parameter α must be carried out. Several numerical experiments are presented for varying α. Furthermore, to make the MMG α independent, different algorithms for the dynamic computations of α are examined. Finally, in chapter 9, the implementation in MARS of the assembly of different bilinear and linear forms by means of compile metaprogramming techniques is presented.

Chapter 2

Preliminaries

2.1 Notation and spaces

We denote scalar quantities by lowercase letters, e.g., a and b. The finite multi-dimensional quantities, such as vectors and matrices, are respectively denoted with bold lowercase letters, e.g., **a** and **b**, and bold capital letters, e.g., **A** and **B**. We denote by $\mathbf{a}_i$ the i-th component of a vector **a** and by $(\mathbf{A})_{i,j}$ the component of **A** related to the i-th row and the j-th column. We also use the symbols $\lesssim$ and $\gtrsim$ to denote inequalities that involve a generic constant. To be more precise, the constant can depend on some quantities related to the problem; however, these quantities are not relevant to the inequality itself. Of course, the fact of being relevant or not will depend on the context in which the inequality is involved.

We will denote by $\Omega \subset \mathbb{R}^d$, with $d = 2, 3$, a bounded domain with Lipschitz boundary $\partial\Omega$ and outward normal $\mathbf{n} \in \mathbb{R}^d$. The elements of Ω are denoted by $\mathbf{x} = (x_1, ..., x_d)$. For $d = 2$ we can use the notation $\mathbf{x} = (x, y)$.
We denote the space of infinitely-differentiable functions with compact support in Ω as:

$$\mathcal{D}(\Omega) := \{\phi \in C^\infty(\Omega) : \text{supp}(\phi) \text{ is compact}\}\,, \tag{2.1}$$

and define the L^2 dot product as:

$$(u, v)_{L^2(\Omega)} = \int_\Omega u(\mathbf{x})v(\mathbf{x})d\mathbf{x}\,. \tag{2.2}$$

We define the following Hilbert space:

$$H^1(\Omega) := \{v \in L^2(\Omega) :\; \nabla v \in L^2(\Omega)\}\,, \tag{2.3}$$

equipped with a norm induced by the scalar product $(\cdot,\cdot)_{H^1(\Omega)} : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$, defined by:

$$(u,v)_{H^1(\Omega)} := (u,v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)} \,. \tag{2.4}$$

If it is clear from the context, we can omit from the scalar product the subscript $L^2(\Omega)$. The space $H^1(\Omega)$ can be generalized for a non-negative integer m:

$$H^m(\Omega) := \{v \in L^2(\Omega) :\ D^\alpha v \in L^2(\Omega) \quad \forall |\alpha| \leq m\}\,, \tag{2.5}$$

equipped with a norm induced by the scalar product $(\cdot,\cdot)_{H^m(\Omega)} : H^m(\Omega) \times H^m(\Omega) \to \mathbb{R}$, defined by:

$$(u,v)_{H^m(\Omega)} := \sum_{|\alpha| \leq m} (D^\alpha u, D^\alpha v)_{L^2(\Omega)}\,, \tag{2.6}$$

where

$$\alpha = (\alpha_1, \dots, \alpha_d), \qquad |\alpha| = \sum_{1 \leq i \leq d} \alpha_i\,, \tag{2.7}$$

and:

$$D^\alpha [\cdot] = \frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \dots \frac{\partial^{\alpha_d}}{\partial x_d^{\alpha_d}} [\cdot]\,. \tag{2.8}$$

The semi-norm is defined as $|u|^2_{H^m(\Omega)} := \sum_{|\alpha|=m} (D^\alpha u, D^\alpha u)_{L^2(\Omega)}$. The same definition can be extended from Sobolev spaces to Sobolev-Slobodeckij spaces with a real exponent $H^{m+s}(\Omega)$, with $s \in (0,1)$, with the following norm:

$$||v||^2_{H^{m+s}(\Omega)} = ||v||^2_{H^m(\Omega)} + \sum_{|\alpha|=m} \int_\Omega \int_\Omega \frac{|D^\alpha v(\mathbf{x}) - D^\alpha v(\mathbf{y})|^2}{|\mathbf{x}-\mathbf{y}|^{d+2s}} d\mathbf{x} d\mathbf{y}\,. \tag{2.9}$$

A Hilbert space which will play a preminent role in this work is $H_{\text{div}}(\Omega)$, defined in the following way:

$$H_{\text{div}}(\Omega) := \{\boldsymbol{\sigma} \in \left[L^2(\Omega)\right]^d :\ \text{div}\boldsymbol{\sigma} \in L^2(\Omega)\}\,, \tag{2.10}$$

and equipped with the following scalar product:

$$(\boldsymbol{\sigma}, \boldsymbol{\tau})_{\text{div},\Omega} = (\boldsymbol{\sigma}, \boldsymbol{\tau})_{L^2(\Omega)} + (\text{div}\boldsymbol{\sigma}, \text{div}\boldsymbol{\tau})_{L^2(\Omega)}\,, \tag{2.11}$$

where $\boldsymbol{\sigma}, \boldsymbol{\tau} \in H_{\text{div}}(\Omega)$. In addition, for $d = 2, 3$ we also define the Hilbert space $H_{\text{curl}}(\Omega)$:

$$H_{\text{curl}}(\Omega) := \{\boldsymbol{\sigma} \in \left[L^2(\Omega)\right]^d :\ \mathbf{curl}\,\boldsymbol{\sigma} \in \left[L^2(\Omega)\right]^d\}\,, \tag{2.12}$$

with a scalar product given by:

$$(\boldsymbol{\sigma}, \boldsymbol{\tau})_{\mathrm{curl},\Omega} = (\boldsymbol{\sigma}, \boldsymbol{\tau})_{L^2(\Omega)} + (\mathrm{curl}\boldsymbol{\sigma}, \mathrm{curl}\boldsymbol{\tau})_{L^2(\Omega)}\,, \tag{2.13}$$

where $\boldsymbol{\sigma}, \boldsymbol{\tau} \in H_{\mathrm{curl}}(\Omega)$. For $d = 2$, the curl applied to a scalar field $v : \Omega \to \mathbb{R}$ is defined as the 90° linearized rotation, denoted with the matrix $\mathbf{R}$, of the gradient of v:

$$\mathbf{curl}\, v = \mathbf{R}\nabla v\,. \tag{2.14}$$

In case of multi-dimensional spaces, the corresponding symbol will be marked in bold. So for example, $\mathbf{H}^1(\Omega) = \left[H^1(\Omega)\right]^d$ or $\mathbf{H}_{\mathrm{div}}(\Omega) = [H_{\mathrm{div}}(\Omega)]^d$. Finally, spaces of finite dimension n, which are isomorphic to $\mathbb{R}^n$, will be marked in bold.

2.2 Trace theorems

Partial differential equations (PDEs) require the definition of proper boundary conditions for the unknown of the problem. Therefore the evaluation of a function on the boundary is somehow necessary. A function $v \in L^2(\Omega)$ that is defined almost everywhere cannot be evaluated on the boundary. Indeed the d-dimensional Lebesgue measure of $\partial\Omega$ is zero, because $\partial\Omega$ has dimension $d-1$. Therefore the function v can arbitrarily assume any value on $\partial\Omega$. However, if the function v is sufficiently regular, it is possible to define the so-called trace operator that allows us to evaluate the function on the boundary. Depending on the definition of the space, different trace operators can be defined. In the following, we will give an insight of the topic based on Brezzi and Fortin [2012], Kober, Bernhard [2020].

If the boundary $\partial\Omega$ is smooth enough, e.g Lipischtz continuous, it exists a continuous linear operator $\gamma : H^1(\Omega) \to L^2(\partial\Omega)$ such that $\gamma(v) = v|_{\partial\Omega}$ for sufficiently smooth v. We call γ the trace operator and $\gamma(v)$ the trace of v on $\partial\Omega$. Even though the equality $\gamma(v) = v|_{\partial\Omega}$ makes sense only if the right-hand side of the equation is well defined, for the sake of simplicity, we will make use of it with abuse of notation.

The space $\gamma(H^1(\Omega))$ does not coincide with the space $L^2(\partial\Omega)$, but it is a strict subset, i.e:

$$H^1(\partial\Omega) \subset \gamma(H^1(\Omega)) \subset H^0(\partial\Omega) \equiv L^2(\partial\Omega)\,. \tag{2.15}$$

From this inclusion, it is possible to define a relation between $\gamma(H^1(\Omega))$ and some Sobolev space $H^s(\partial\Omega)$ with $s \in (0,1)$. Indeed we can define:

$$H^{1/2}(\partial\Omega) = \gamma(H^1(\Omega))\,, \tag{2.16}$$

equipped with the following norm:

$$\|g\|_{H^{1/2}(\partial\Omega)} = \sup_{\substack{u\in H^1(\Omega)\\ \gamma(u)=g}} \|u\|_{H^1(\Omega)} \,. \tag{2.17}$$

We denote the dual space of $H^{1/2}(\partial\Omega)$ by $H^{-1/2}(\partial\Omega)$ and we equip it with the following norm:

$$\|v\|_{H^{-1/2}(\partial\Omega)} = \inf_{u\in H^{1/2}(\partial\Omega)} \frac{\langle v,u\rangle_{H^{-1/2}(\partial\Omega)\times H^{1/2}(\partial\Omega)}}{\|u\|_{H^{1/2}(\partial\Omega)}} \,, \tag{2.18}$$

where $\langle\cdot,\cdot\rangle_{H^{-1/2}(\partial\Omega)\times H^{1/2}(\partial\Omega)}$ is the dual pairing between $H^{-1/2}(\partial\Omega)$ and $H^{1/2}(\partial\Omega)$. We can also define the linear and continuous normal trace of $H_{\text{div}}(\Omega)$, which is $\pi : H_{\text{div}}(\Omega) \to H^{-1/2}(\partial\Omega)$. It satisfies $\pi(\boldsymbol{\sigma}) = \boldsymbol{\sigma}\cdot\mathbf{n}|_{\partial\Omega}$ whenever $\boldsymbol{\sigma}$ is smooth enough. As for the trace of $H^1(\Omega)$, with an abuse of notation we will write $\boldsymbol{\sigma}\cdot\mathbf{n}|_{\partial\Omega}$ instead of $\pi(\boldsymbol{\sigma})$. Then we have the validity of Green's formula:

$$\int_\Gamma u\boldsymbol{\sigma}\cdot\mathbf{n}d\mathbf{s} = \int_\Omega \boldsymbol{\sigma}\cdot\nabla u d\mathbf{x} + \int_\Omega u\,\text{div}\boldsymbol{\sigma} d\mathbf{x} \qquad \forall u\in H^1(\Omega), \quad \forall\boldsymbol{\sigma}\in H_{\text{div}}(\Omega)\,. \tag{2.19}$$

Usually, in PDE problems, especially the contact problems, the boundary $\partial\Omega$ is subdivided into more disjoint subsets. Therefore the trace operators have to be understood also on portions of $\partial\Omega$. This topic is not trivial, in particular for the case of the trace related to H_{div}, and we again refer the reader to Brezzi and Fortin [2012], Kober, Bernhard [2020].

2.3 Spaces for the contact problem

For the definition of the functionals and the related weak forms used in this thesis, we have to introduce the following spaces subject to equality constraints on the boundary:

$$\mathbf{H}^1_{0,\Gamma_D}(\Omega) := \{\mathbf{u}\in\mathbf{H}^1(\Omega) : \mathbf{u}|_{\Gamma_D} = \mathbf{0}\}\,, \tag{2.20a}$$

$$\mathbf{H}^1_{\mathbf{g}_D,\Gamma_D}(\Omega) := \{\mathbf{u}\in\mathbf{H}^1(\Omega) : \mathbf{u}|_{\Gamma_D} = \mathbf{g}_D\}\,, \tag{2.20b}$$

$$\mathbf{H}_{0,\Gamma_N,\text{div}}(\Omega) := \{\boldsymbol{\sigma}\in\mathbf{H}_{\text{div}}(\Omega) : \boldsymbol{\sigma}\cdot\mathbf{n}|_{\Gamma_N} = \mathbf{0}\}\,, \tag{2.20c}$$

$$\mathbf{H}_{\mathbf{g}_N,\Gamma_N,\text{div}}(\Omega) := \{\boldsymbol{\sigma}\in\mathbf{H}_{\text{div}}(\Omega) : \boldsymbol{\sigma}\cdot\mathbf{n}|_{\Gamma_N} = \mathbf{g}_N\}\,. \tag{2.20d}$$

We see that only the first and the third sets with homogeneous boundary conditions are linear spaces. However, it is always possible to recast problems with non homogeneous boundary conditions into problems that are subject only to homogeneous boundary conditions, by means of a simple transformation. For simplicity of notation we can set:

$$\mathbf{U} := \mathbf{H}^1(\Omega)\,, \tag{2.21a}$$

$$\mathbf{\Sigma} := \mathbf{H}_{\text{div}}(\Omega)\,, \tag{2.21b}$$

$$\mathbf{U}_0 := \mathbf{H}^1_{0,\Gamma_D}(\Omega)\,, \tag{2.21c}$$

$$\mathbf{\Sigma}_0 := \mathbf{H}_{0,\Gamma_N,\text{div}}(\Omega)\,, \tag{2.21d}$$

$$\mathbf{U}_{\mathbf{g}_D} := \mathbf{H}^1_{\mathbf{g}_D,\Gamma_D}(\Omega)\,, \tag{2.21e}$$

$$\mathbf{\Sigma}_{\mathbf{g}_N} := \mathbf{H}_{\mathbf{g}_N,\Gamma_N,\text{div}}(\Omega). \tag{2.21f}$$

We also define the space of rotations:

$$\mathbf{\Theta} := \{\boldsymbol{\gamma} \in \left[L^2(\Omega)\right]^{d,d} : \boldsymbol{\gamma} + \boldsymbol{\gamma}^T = \mathbf{0}\} \tag{2.22}$$

and

$$\mathbf{V} = \left[L^2(\Omega)\right]^d\,. \tag{2.23}$$

2.4 Definitions and theorems for constrained minimization problems

As we will see in the next chapters, the contact problems can be formulated as constrained minimization problems. Therefore we need a theoretical framework and proper tools to show their well-posedness. To this aim, we now give a list of some useful definitions and theorems from Kikuchi and Oden [1988]. For the proofs, we refer the reader to the same book. In this section, we just want to explore the main general theorem for the minimization of a functional on a closed convex set and its particular case which will apply to the formulations of this thesis. To this purpose, let V be a reflexive Banach space equipped with the norm $||\cdot||_V : V \to \mathbb{R}$ and let V' be its dual space. We denote by $\langle \cdot, \cdot \rangle_{V' \times V} : V' \times V \to \mathbb{R}$ the duality pairing between V' and V. We can omit the subscript if it is clear from the context. We also denote by $\mathcal{K}$ a closed convex set of V. We give the following definitions.

Definition 1 (Weak convergence). *A sequence $\{u_n\} \subset V$ converges weakly to $u \in V$ if:*

$$\lim_{n\to\infty} F(u_n) = F(u) \qquad \forall F \in V'\,. \tag{2.24}$$

Definition 2 (Weak lower semicontinuity). *A functional $F : \mathcal{K} \to \mathbb{R}$ is weakly lower semicontinuous on $\mathcal{K}$ if, for any sequence $\{u_n\} \subset \mathcal{K}$ converging weakly to $u \in \mathcal{K}$, it satisfies:*

$$\liminf_{n\to\infty} F(u_n) \geq F(u)\,. \tag{2.25}$$

Definition 3 (Coercivity). *A functional $F : \mathcal{K} \to \mathbb{R}$ is coercive on $\mathcal{K}$ if:*

$$\lim_{\|v\|_V\to\infty} F(v) = +\infty \qquad \forall v \in \mathcal{K}\,. \tag{2.26}$$

Definition 4 (Ellipticity). *A bilinear form $a : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ is elliptic on $\mathcal{K}$ if:*

$$a(v,v) \gtrsim \|v\|_V^2 \qquad \forall v \in \mathcal{K}\,. \tag{2.27}$$

Theorem 2.4.1. *Let $a : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ be an elliptic bilinear form on $\mathcal{K}$. Then a is coercive on $\mathcal{K}$.*

Definition 5 (Convexity). *The functional $F : \mathcal{K} \to \mathbb{R}$ is convex on $\mathcal{K}$ if:*

$$F(\alpha v + (1-\alpha)u) \leq \alpha F(v) + (1-\alpha)F(u) \qquad \forall u, v \in \mathcal{K}, \quad \alpha \in [0,1]\,. \tag{2.28}$$

It is strictly convex if the inequality holds strictly.

Definition 6 (G-differentiability). *The functional $F : \mathcal{K} \to \mathbb{R}$ is G-differentiable at $u \in \mathcal{K}$ if there exists a linear functional $DF(u) \in V'$ such that, given $\epsilon > 0$:*

$$\lim_{\epsilon\to 0} F(u + \epsilon v) = \langle DF(u), v\rangle \qquad \forall v \in \mathcal{K}\,. \tag{2.29}$$

Theorem 2.4.2 (Minimization 1). *Let $F : \mathcal{K} \to \mathbb{R}$ be a weakly lower semicontinuous functional on $\mathcal{K}$, that satisfies* Definition 2*. If $\mathcal{K}$ is bounded or if $\mathcal{K}$ is unbounded and F is coercive, satisfying* Definition 3*, then F attains its minimum value on $\mathcal{K}$, i.e.:*

$$\exists u \in \mathcal{K} : \quad F(u) \leq F(v) \qquad \forall v \in \mathcal{K}\,. \tag{2.30}$$

Since lower semicontinuity can be difficult to prove, the following Theorem 2.4.3 states that convexity and G-differentiability can be proven it its place.

Theorem 2.4.3. *Let $F : \mathcal{K} \to \mathbb{R}$ satisfy the following conditions:*

- *F is convex;*
- *F is G-differentiable on $\mathcal{K}$;*

Then F is weakly lower semicontinuous on $\mathcal{K}$.

In this way, we can specify the Theorem 2.4.2 by using Theorem 2.4.3. We obtain Theorem 2.4.4.

Theorem 2.4.4 (Minimization 2). *Let $F : \mathcal{K} \to \mathbb{R}$ be a convex and coercive functional on $\mathcal{K}$. Then F attains its minimum value on $\mathcal{K}$. If, in addition, f is strictly convex, the minimum is unique.*

We can further specialize Theorem 2.4.4 by considering elliptic quadratic functionals. We obtain the following theorems.

Definition 7 (Elliptic quadratic functional). *We say $F : \mathcal{K} \to \mathbb{R}$ is an elliptic quadratic functional on $\mathcal{K}$ if:*

$$F(v) = \frac{1}{2}a(v,v) - f(v) \qquad \forall v \in \mathcal{K}, \tag{2.31}$$

where $a : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ is a symmetric, continuous and elliptic bilinear form:

$$a(u,v) = a(v,u) \qquad \forall u,v \in \mathcal{K}, \tag{2.32a}$$

$$a(u,v) \lesssim \|u\|_V \|v\|_V \qquad \forall u,v \in \mathcal{K}, \tag{2.32b}$$

$$a(v,v) \gtrsim \|v\|_V^2 \qquad \forall v \in \mathcal{K}, \tag{2.32c}$$

and $f : \mathcal{K} \to \mathbb{R}$ is a continuous linear form, i.e.:

$$f(v) \lesssim \|v\|_V \qquad \forall v \in \mathcal{K}. \tag{2.33}$$

Theorem 2.4.5. *Let $F : \mathcal{K} \to \mathbb{R}$ be an elliptic quadratic functional on $\mathcal{K}$. Then F is convex, G-differentiable and coercive on $\mathcal{K}$.*

Theorem 2.4.6 (Minimization 3). *Let $F : \mathcal{K} \to \mathbb{R}$ be an elliptic quadratic functional on $\mathcal{K}$. Then F attains its unique minimum value on $\mathcal{K}$.*

We end the section with the Poincaré and Korn's inequalities, which are very important results for many proofs in mechanics.

Theorem 2.4.7 (Poincaré's inequality). *There exists C_p*

$$\|v\|_{H^1(\Omega)}^2 \leq C_p \|\nabla v\|_{L^2(\Omega)}^2 \qquad \forall v \in H^1_{0,\Gamma_D}(\Omega). \tag{2.34}$$

Theorem 2.4.8 (Korn's inequality). *There exists C_k*

$$\|\boldsymbol{u}\|_{\boldsymbol{H}^1(\Omega)}^2 \leq C_k \|\boldsymbol{\varepsilon}(\boldsymbol{u})\|_{\boldsymbol{L}^2(\Omega)}^2 \qquad \forall \boldsymbol{u} \in \boldsymbol{H}^1_{0,\Gamma_D}(\Omega), \tag{2.35}$$

where $\boldsymbol{\varepsilon}(\boldsymbol{u}) := \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)$ is the symmetric part of the gradient of $\boldsymbol{u}$.

Chapter 3

The linear elasticity problem

In this chapter, we introduce the strong formulation of linear elasticity. Then we discuss the corresponding weak formulations of the primal, the dual, and the FOSLS formulations.

3.1 The strong formulation

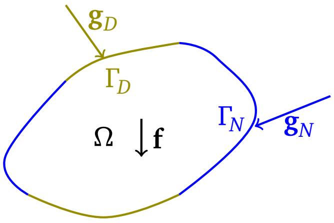


Figure 3.1. Linear elasticity model problem.

Let $\Omega \subset \mathbb{R}^d$, with boundary $\partial\Omega$, represent the solid body of interest. We can consider two disjoint open sets of the boundary, the Neumann boundary Γ_N and the Dirichlet boundary Γ_D, satisfying $\overline{\partial\Omega} = \overline{\Gamma}_N \cup \overline{\Gamma}_D$, with $\Gamma_N \cap \Gamma_D = \emptyset$. The body is subject to the external volumetric force $\mathbf{f} : \Omega \to \mathbb{R}^d$ on Ω and to the boundary force $\mathbf{g}_N : \Gamma_N \to \mathbb{R}^d$ on Γ_N. On Γ_D a prescribed displacement $\mathbf{g}_D$ is enforced. We seek for sufficiently smooth displacements $\mathbf{u} : \Omega \to \mathbb{R}^d$ and internal stresses $\boldsymbol{\sigma} : \Omega \to \mathbb{R}^{d,d}$ which solve the linear elasticity problem described in Figure 3.1. Linear elasticity requires to enforce equilibrium directly on the reference configuration Ω, instead of its deformed configuration, which is unknown. Then the conservation of linear and angular momenta of the body Ω are given by the following relations:

$$\begin{aligned} \operatorname{div}\boldsymbol{\sigma} &= -\mathbf{f} \qquad \text{in } \Omega\,, && (3.1a)\\ \mathbf{as}\,\boldsymbol{\sigma} &= \mathbf{0} \qquad \text{in } \Omega\,, && (3.1b) \end{aligned}$$

where we define

$$\mathbf{as}\;\boldsymbol{\sigma} := \frac{1}{2}(\boldsymbol{\sigma} - \boldsymbol{\sigma}^T)\,, \tag{3.2}$$

the antisymmetric part of $\boldsymbol{\sigma}$. Furthermore, the strains have to be small enough so that the kinematic relation $\boldsymbol{\varepsilon}$ between strains and displacements reduces to a linear one:

$$\boldsymbol{\varepsilon}(\mathbf{u}) := \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\,, \tag{3.3}$$

and the constitutive law has to be a linear relation between the stress and the strain:

$$\boldsymbol{\sigma} = \mathcal{C}\boldsymbol{\varepsilon}. \tag{3.4}$$

In particular, for a homogeneous and isotropic body, the constitutive Hook's law reads as follows:

$$\boldsymbol{\sigma} = \mathcal{C}\boldsymbol{\varepsilon} := 2\mu\boldsymbol{\varepsilon} + \lambda(\operatorname{tr}(\boldsymbol{\varepsilon}))\mathbf{I}\,, \tag{3.5}$$

where $\mathcal{C}$ is the *stiffness tensor*, μ and λ are the Lamé parameters, $\mathbf{I} \in \mathbb{R}^{d,d}$ is the identity matrix in d-dimension and $\operatorname{tr}:\mathbb{R}^{d,d} \to \mathbb{R}$, defined such that $\operatorname{tr}(\cdot) := \sum_{i=1}^{d}[\cdot]_{ii}$, is the trace operator. Since the stiffness tensor has full rank, we can also define its inverse, the so-called *compliance tensor* $\mathcal{A}$ such that:

$$\boldsymbol{\varepsilon} = \mathcal{A}\boldsymbol{\sigma} := \frac{1}{2\mu}\left(\boldsymbol{\sigma} - \frac{\lambda}{d\lambda + 2\mu}(\operatorname{tr}\boldsymbol{\sigma})\mathbf{I}\right)\,. \tag{3.6}$$

We can observe that, in the incompressible limit $\lambda \to \infty$, only the constitutive law expressed in terms of the compliance tensor is bounded and defined for any entry. Given any bounded $\boldsymbol{\sigma}$ and any bounded $\boldsymbol{\varepsilon}$ such that $\operatorname{tr}\boldsymbol{\varepsilon} = 0$:

$$\begin{aligned} \lim_{\lambda\to\infty} \mathcal{C}\boldsymbol{\varepsilon} &= 2\mu\boldsymbol{\varepsilon}\,,\\ \lim_{\lambda\to\infty} \mathcal{A}\boldsymbol{\sigma} &= \frac{1}{2\mu}\left(\boldsymbol{\sigma} - \frac{\operatorname{tr}\boldsymbol{\sigma}}{d}\mathbf{I}\right)\,. \end{aligned} \tag{3.7}$$

Finally we must also close the problem with the boundary conditions:

$$\boldsymbol{\sigma}\mathbf{n} = \mathbf{g}_N \qquad \text{on } \Gamma_N\,, \tag{3.8a}$$

$$\mathbf{u} = \mathbf{g}_D \qquad \text{on } \Gamma_D\,. \tag{3.8b}$$

The strong formulation of linear elasticity for a homogenous and isotropic material is:

$$\operatorname{div}\boldsymbol{\sigma} = -\mathbf{f} \quad \text{in } \Omega\,, \tag{3.9a}$$

$$\mathbf{as}\,\boldsymbol{\sigma} = \mathbf{0} \quad \text{in } \Omega\,, \tag{3.9b}$$

$$\mathcal{A}\boldsymbol{\sigma} = \boldsymbol{\varepsilon} \quad \text{in } \Omega\,, \tag{3.9c}$$

$$\boldsymbol{\sigma}\mathbf{n} = \mathbf{g}_N \quad \text{on } \Gamma_N\,, \tag{3.9d}$$

$$\mathbf{u} = \mathbf{g}_D \quad \text{on } \Gamma_D\,, \tag{3.9e}$$

which is meant to be pointwise. On the other hand, the formuations that we will present in the next sections have to be understood in a weak sense.

3.2 The compressible displacement-based primal formulation

3.2.1 Weak form and minimization problem

We search for $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$ and $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N}$. We multiply the first equilibrium condition in (3.1a) by a test function $\mathbf{v} \in \mathbf{U}_0$, integrate over the volume Ω and then integrate by parts:

$$-\int_\Omega \operatorname{div}\boldsymbol{\sigma}\cdot\mathbf{v} = \int_\Omega \mathbf{f}\cdot\mathbf{v}\,, \tag{3.10a}$$

$$\int_\Omega \boldsymbol{\sigma} : \nabla\mathbf{v} = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_{\partial\Omega} (\boldsymbol{\sigma}\mathbf{n})\cdot\mathbf{v}\,, \tag{3.10b}$$

$$\int_\Omega \boldsymbol{\sigma} : \nabla\mathbf{v} = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_{\Gamma_N} \mathbf{g}_N\cdot\mathbf{v}\,, \tag{3.10c}$$

where we use $\mathbf{v}|_{\Gamma_D} = \mathbf{0}$ and $\boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{g}_N$ on Γ_N. We can now first exploit both (3.2) and (3.3) and then (3.5) to get:

$$\int_\Omega \boldsymbol{\sigma} : \nabla\mathbf{v} = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_{\Gamma_N} \mathbf{g}_N\cdot\mathbf{v}\,, \tag{3.11a}$$

$$\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\varepsilon}(\mathbf{v}) = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_{\Gamma_N} \mathbf{g}_N\cdot\mathbf{v}\,. \tag{3.11b}$$

As it is clear, we get the following weak form, where we search for $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$ such that:

$$\int_\Omega \mathcal{C}\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v}) = \int_\Omega \mathbf{f} \cdot \mathbf{v} + \int_{\Gamma_N} \mathbf{g}_N \cdot \mathbf{v} \qquad \forall \mathbf{v} \in \mathbf{U}_0 \,, \tag{3.12}$$

that can be rewritten by means of the bilinear form $a_p : \mathbf{U} \times \mathbf{U} \to \mathbb{R}$ and linear form $f_p : \mathbf{U} \to \mathbb{R}$:

$$a_p(\mathbf{u}, \mathbf{v}) := (\mathcal{C}\boldsymbol{\varepsilon}(\mathbf{u}), \boldsymbol{\varepsilon}(\mathbf{v}))_{L^2(\Omega)} \,, \tag{3.13a}$$

$$f_p(\mathbf{v}) := (\mathbf{f}, \mathbf{v})_{L^2(\Omega)} + \langle \mathbf{g}_N, \mathbf{v} \rangle_{\Gamma_N} \,, \tag{3.13b}$$

so that we search for $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$ such that:

$$a_p(\mathbf{u}, \mathbf{v}) = f_p(\mathbf{v}) \qquad \forall \mathbf{v} \in \mathbf{U}_0 \,. \tag{3.14}$$

The variable $\boldsymbol{\sigma}$ does not appear in the formulation and can be postprocessed by means of (3.5). Therefore, in the primal formulation, the main variable is the displacement $\mathbf{u}$.

The formulation (3.12) is of the Bubnov-Galerkin type, meaning that $\mathbf{u}$ and $\mathbf{v}$ do not belong to the same space. However, given the known function $\mathbf{u}_{\mathbf{g}_D} \in \mathbf{U}_{\mathbf{g}_D}$, we can write $\mathbf{u}$ in terms of a new unknown $\mathbf{u}_0 \in \mathbf{U}_0$ so that:

$$\mathbf{u} = \mathbf{u}_{\mathbf{g}_D} + \mathbf{u}_0 \,. \tag{3.15}$$

Indeed we can substitute this formula and express (3.12) in terms of $\mathbf{u}_0$ and $\mathbf{v}$, so that the new trial and test functions belong to $\mathbf{U}_0$. Then one can solve the problem and finally reobtain $\mathbf{u}$ as $\mathbf{u} = \mathbf{u}_{\mathbf{g}_D} + \mathbf{u}_0$. This fact will be implicitly used also for the other formulations.

The equation (3.12) can also be read as the first order necessary condition for the minimization of the following quadratic functional $\mathcal{F}_p : \mathbf{U} \to \mathbb{R}$ over $\mathbf{U}_{\mathbf{g}_D}$:

$$\mathcal{F}_p(\mathbf{u}) := \frac{1}{2} a_p(\mathbf{u}, \mathbf{u}) - f_p(\mathbf{u}) \,. \tag{3.16}$$

The solution $\hat{\mathbf{u}}$ is then the minimizer $\hat{\mathbf{u}}$ such that:

$$\hat{\mathbf{u}} = \underset{\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}}{\arg\min} \, \mathcal{F}_p(\mathbf{u}) \,. \tag{3.17}$$

3.2.2 Existence and uniqueness

Existence and uniqueness of (3.19) can be proved by means of (2.4.6). We again refer the reader to Kikuchi and Oden [1988] for a more detailed discussion. Indeed, the set $\mathbf{U}_{\mathbf{g}_D}$ is a non-empty convex set. The functional $\mathcal{F}_p$ is quadratic and it can be easily proved that f_p is linear and continuous. The same goes for the symmetry and the continuity of a_p. For its ellipticity, Theorem 2.35 is required. Then Theorem 2.4.6 follows.

3.3 The incompressible displacement-based primal formulation

In (3.12), the substitution of the constitutive law $\boldsymbol{\sigma} = \mathcal{C}\boldsymbol{\varepsilon}(\mathbf{u})$, with $\mathbf{u}$ as the independent variable, enables to write the whole problem only in terms of the displacement. However, such a choice makes difficult dealing with nearly incompressible ($\lambda \gg 1$) and incompressible ($\lambda = \infty$) materials. Indeed the tensor $\mathcal{C}$, as explicited in (3.7), becomes unbounded for increasing λ. At least for the incompressible case, one can enforce the incompressibility constraint $\mathrm{div}\mathbf{u} = 0$ by means of the pressure p as Lagrange multiplier. The functional (3.16) can be consequently augmented:

$$\mathcal{F}_{p,\mathrm{aug}}(\mathbf{u}, p) := \mathcal{F}_p(\mathbf{u}) + \int_\Omega \mathrm{div}\mathbf{u}p \,. \tag{3.18}$$

Then the solution of the problem is the pair $(\hat{\mathbf{u}}, \hat{p})$ computed as:

$$(\hat{\mathbf{u}}, \hat{p}) = \arg min_{\mathbf{u}\in\mathbf{U}_{\mathbf{g}_D}} \arg max_{p\in L^2(\Omega)} \mathcal{F}_{p,\mathrm{aug}}(\mathbf{u}, p) \,. \tag{3.19}$$

The weak form reads as the following saddle point problem: search for $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$ and $p \in L^2(\Omega)$ such that:

$$2\mu \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v}) + \int_\Omega p\mathrm{div}(\mathbf{v}) = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_{\Gamma_N} \mathbf{g}_N \cdot \mathbf{v} \qquad \forall \mathbf{v} \in \mathbf{U}_0 \,, \tag{3.20a}$$

$$\int_\Omega q\mathrm{div}(\mathbf{u}) = 0 \qquad \forall q \in L^2(\Omega) \,, \tag{3.20b}$$

where the term $\mathrm{tr}(\boldsymbol{\varepsilon})$ in the constitutive law is identically zero. The system (3.20) is a saddle point, in contrast to (3.12) which is symmetric and positive definite. Therefore the spaces of (3.20) have to satisfy the LBB (Ladyzhenskaya-Babuška-Brezzi) condition.

3.4 The stress-based dual formulation

3.4.1 Weak form and minimization problem

We can express the constitutive law, not as in (3.5), but as in (3.6). In this way, the tensor $\mathcal{A}\boldsymbol{\sigma}$ we get is bounded for any λ and we are able to describe both nearly-incompressible and incompressible materials. We can then multiply such equation by $\boldsymbol{\tau} \in \boldsymbol{\Sigma}_0$, integrate over the volume, exploit $\nabla \mathbf{u} = \boldsymbol{\varepsilon}(\mathbf{u}) + \mathbf{as}\,(\nabla \mathbf{u})$ and then integrate by parts:

$$\int_\Omega \mathcal{A}\boldsymbol{\sigma} : \boldsymbol{\tau} - \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\tau} = 0\,, \tag{3.21a}$$

$$\int_\Omega \mathcal{A}\boldsymbol{\sigma} : \boldsymbol{\tau} - \int_\Omega \nabla \mathbf{u} : \boldsymbol{\tau} + \int_\Omega \mathbf{as}\,(\nabla \mathbf{u}) : \boldsymbol{\tau} = 0\,, \tag{3.21b}$$

$$\int_\Omega \mathcal{A}\boldsymbol{\sigma} : \boldsymbol{\tau} + \int_\Omega \mathbf{u} \cdot \mathrm{div}\boldsymbol{\tau} + \int_\Omega \mathbf{as}\,(\nabla \mathbf{u}) : \mathbf{as}\;\boldsymbol{\tau} = \int_{\partial\Omega} (\boldsymbol{\tau}\mathbf{n}) \cdot \mathbf{u}\,, \tag{3.21c}$$

$$\int_\Omega \mathcal{A}\boldsymbol{\sigma} : \boldsymbol{\tau} + \int_\Omega \mathbf{u} \cdot \mathrm{div}\boldsymbol{\tau} + \int_\Omega \boldsymbol{\theta} : \mathbf{as}\;\boldsymbol{\tau} = \int_{\Gamma_D} (\boldsymbol{\tau}\mathbf{n}) \cdot \mathbf{g}_D\,, \tag{3.21d}$$

where we have introduced the definition of the rotation variable $\boldsymbol{\theta} := \mathbf{as}\,(\nabla \mathbf{u})$ and have used $\boldsymbol{\tau}\mathbf{n}|_{\Gamma_N} = \mathbf{0}$ and $\mathbf{u}|_{\Gamma_D} = \mathbf{g}_D$. We need also to enforce the equilibrium condition and the symmetry of the stress tensor. To this purpose, we introduce two additional test functions, $\mathbf{v} \in \mathbf{V}$ and $\boldsymbol{\gamma} \in \boldsymbol{\Theta}$, and we need to seek for $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N}$, $\mathbf{u} \in \mathbf{V}$, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ such that:

$$\int_\Omega \mathcal{A}\boldsymbol{\sigma} : \boldsymbol{\tau} + \int_\Omega \mathbf{u} \cdot \mathrm{div}\boldsymbol{\tau} + \int_\Omega \boldsymbol{\theta} : \mathbf{as}\;\boldsymbol{\tau} = \int_{\Gamma_D} (\boldsymbol{\tau}\mathbf{n}) \cdot \mathbf{g}_D \qquad \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}_0, \tag{3.22a}$$

$$\int_\Omega \mathbf{v} \cdot \mathrm{div}\boldsymbol{\sigma} = -\int_\Omega \mathbf{v} \cdot \mathbf{f} \qquad \forall \mathbf{v} \in \mathbf{V}, \tag{3.22b}$$

$$\int_\Omega \boldsymbol{\gamma} : \mathbf{as}\;\boldsymbol{\sigma} = 0 \qquad \forall \boldsymbol{\gamma} \in \boldsymbol{\Theta}, \tag{3.22c}$$

where the boundary condition on the displacement has already been enforced into the boundary integral, no derivative is applied to $\mathbf{u}$ and therefore we just seek for $\mathbf{u} \in \mathbf{V}$ and not in $\mathbf{U}_{\mathbf{g}_D}$. Indeed, in the dual formulation, the dual variable $\boldsymbol{\sigma}$ is the main variable, while the displacement $\mathbf{u}$ and the rotation $\boldsymbol{\theta}$ can be interpreted as Lagrange multipliers for the equilibrium condition (3.1a) and the symmetry of the stress tensor (3.1b). See Brezzi and Fortin [2012], Starke et al. [2011]. The same system (3.22) can be written by using proper bilinear and linear forms:

$$\begin{aligned}
&a_d(\boldsymbol{\sigma},\boldsymbol{\tau}) + b_d(\boldsymbol{\tau},\mathbf{u}) + c_d(\boldsymbol{\tau},\boldsymbol{\theta}) = f_{d,1}(\boldsymbol{\tau}) && \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}_0\,, && (3.23a)\\
&b_d(\boldsymbol{\sigma},\mathbf{v}) \qquad\qquad\qquad\qquad\quad = f_{d,2}(\mathbf{v}) && \forall \mathbf{v} \in \mathbf{V}\,, && (3.23b)\\
&c_d(\boldsymbol{\sigma},\boldsymbol{\gamma}) \qquad\qquad\qquad\qquad\quad = 0 && \forall \boldsymbol{\gamma} \in \boldsymbol{\Theta} && (3.23c)\\
& && && (3.23d)
\end{aligned}$$

where:

$$a_d(\boldsymbol{\sigma},\boldsymbol{\tau}) := \int_\Omega \mathcal{A}\boldsymbol{\sigma} : \boldsymbol{\tau}\,, \tag{3.24a}$$

$$b_d(\boldsymbol{\tau},\mathbf{u}) := \int_\Omega \mathbf{u}\cdot \mathrm{div}\boldsymbol{\tau}\,, \tag{3.24b}$$

$$c_d(\boldsymbol{\tau},\boldsymbol{\theta}) := \int_\Omega \boldsymbol{\theta} : \mathbf{as}\ \boldsymbol{\tau}\,, \tag{3.24c}$$

$$f_{d,1}(\boldsymbol{\tau}) := \int_{\Gamma_D} (\boldsymbol{\tau}\mathbf{n})\cdot \mathbf{g}_D\,, \tag{3.24d}$$

$$f_{d,2}(\mathbf{v}) := -\int_\Omega \mathbf{f}\cdot\mathbf{v}\,. \tag{3.24e}$$

The linear elasticity problem (3.22) can be formulated as the first-order necessary condition for the minimisation of the following quadratic functional $\mathcal{F}_d : \boldsymbol{\Sigma} \to \mathbb{R}$ over the closed convex set $\mathcal{S}_d \subset \boldsymbol{\Sigma}$:

$$\begin{aligned}
\mathcal{F}_d(\boldsymbol{\sigma}) :&= \frac{1}{2} a_d(\boldsymbol{\sigma},\boldsymbol{\sigma}) - f_{d,1}(\boldsymbol{\sigma})\,,\\
\mathcal{S}_d :&= \{\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N} :\ b_d(\boldsymbol{\sigma},\mathbf{v}) = f_{d,2}(\mathbf{v}) \quad \forall \mathbf{v}\in\mathbf{V},\\
&\qquad\qquad\qquad\ \ c_d(\boldsymbol{\sigma},\boldsymbol{\gamma}) = 0 \qquad\quad \forall \boldsymbol{\gamma}\in\boldsymbol{\Theta}\}\,.
\end{aligned} \tag{3.25}$$

The solution $\hat{\boldsymbol{\sigma}}$ is then the minimizer $\hat{\boldsymbol{\sigma}}$ such that:

$$\hat{\boldsymbol{\sigma}} = \arg\min_{\boldsymbol{\sigma}\in\mathcal{S}_d} \mathcal{F}_d(\boldsymbol{\sigma})\,. \tag{3.26}$$

3.4.2 Existence and uniqueness

As for the primal formulation, we want to discuss existence and uniqueness using Theorem 2.4.6. As explained in Brezzi and Fortin [2012], the set $\mathcal{S}_d$ is convex and closed. Furthermore it is not empty, because the following LBB condition is fulfilled: there exists $\beta > 0$ such that:

$$\inf_{\substack{\mathbf{u}\in\mathbf{V}\\ \boldsymbol{\theta}\in\boldsymbol{\Theta}}} \sup_{\boldsymbol{\sigma}\in\Sigma} \frac{b_d(\boldsymbol{\sigma},\mathbf{u}) + c_d(\boldsymbol{\sigma},\boldsymbol{\theta})}{\|\boldsymbol{\sigma}\|_{\Sigma}(\|\mathbf{u}\|_{\mathrm{U}} + \|\boldsymbol{\theta}\|_{\Theta})} \geq \beta > 0\,. \tag{3.27}$$

The functional $\mathcal{F}_d$ is quadratic. It is easy to show that $f_{d,1}$ and $f_{d,2}$ are linear and continuous. The forms b_d and c_d are bilinear and continuous as well. Also a_d is bilinear, symmetric and continuous. As explained in Brezzi and Fortin [2012], it is also possible to show that:

$$a_p(\boldsymbol{\sigma},\boldsymbol{\sigma}) \geq \frac{1}{d(d\lambda+2\mu)}\|\boldsymbol{\sigma}\|^2_{\mathrm{L}^2(\Omega)} \qquad \forall\boldsymbol{\sigma}\in\boldsymbol{\Sigma}\,, \tag{3.28}$$

where the ellipticity holds for any $\lambda \geq 0$, but not for $\lambda \to \infty$. In this case, ellipticity is lost. However, ellipticity must hold on the convex set $\mathcal{S}_d$ and not on the whole space $\boldsymbol{\Sigma}$. Assuming homogeneous Dirichlet boundary conditions, i.e., $\mathbf{g}_D = \mathbf{0}$, it is possible to prove that:

$$\|\boldsymbol{\sigma}\|^2_{\Sigma} \lesssim (\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})_{L^2(\Omega)} \quad \forall\boldsymbol{\sigma}\in\{\boldsymbol{\sigma}\in\boldsymbol{\Sigma}: \quad \int_\Omega \mathrm{tr}\boldsymbol{\sigma}=0, \quad b_d(\boldsymbol{\sigma},\mathbf{v})=0 \;\; \forall\mathbf{v}\in\mathbf{V}\}\,. \tag{3.29}$$

For more details on this topic, see Brezzi and Fortin [2012] and Kober, Bernhard [2020].

Remark 3.4.1. *The minimization problem (3.26) can be also written as a min-max problem, by using the Lagrangian functional that takes into account also the Lagrange multipliers,* $\boldsymbol{u}$ *and* $\boldsymbol{\theta}$*. To this purpose, the global equality constraints, that are now included in* S_d*, would be transferred into the Lagrangian functional. However, the monotone multigrid, described in Chapter 6 for the primal, the dual and the FOSLS formulation, is based on the minimization of a functional. For this reason, we prefer to write the dual formulation as a minimization problem instead of a min-max problem.*

3.5 FOSLS formulation

3.5.1 Weak form and minimization problem

For all the previous formulations, we have multiplied some of the equations in their strong form by proper test functions, integrated at first over the volume and then by parts. Then we have noticed that the same weak forms can be seen as the first-order necessary condition for the minimization of a given functional. On the other hand, in the first-order system least squares (FOSLS) case, we directly start by defining a functional. From the first-order necessary condition for its minimization, the FOSLS weak form is recovered.

Indeed the main idea is to build a fictitious functional as the weighted sum of the squared L^2-norms of the residuals of the equations defined on the domain. As literature about this topic we can mention Cai, Zhiqiang and Lazarov, R. and Manteuffel, Thomas A. and McCormick, Stephen F. [1994], Yang, Suh-Yuh and Liu, Jinn-Liang [1997], Berndt et al. [1997], Starke [1999], Cai, Zhiqiang and Starke, Gerhard [2004], Berndt et al. [2005], Brandts et al. [2006], Starke [2007], Starke [2009], Bochev, Pavel B. and Gunzburger, Max D. [2009], Müller, Benjamin [2015], Krause, Rolf and Müller, Benjamin and Starke, Gerhard [2017]. The FOSLS functional is a mathematical tool that does not have a physical meaning. It is also clear that now the primal and the dual variables are both main variables of the problem.

For the linear elasticity problem, the FOSLS principle is translated in finding $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$, $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N}$, that minimize the least-square functional $\mathcal{F}_f : \mathbf{U} \times \boldsymbol{\Sigma} \to \mathbb{R}$ over the set $\mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_{\mathbf{g}_N}$:

$$\mathcal{F}_f(\mathbf{u}, \boldsymbol{\sigma}) = \gamma ||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||^2_{L^2(\Omega)} + \delta ||\text{div}\boldsymbol{\sigma} + \mathbf{f}||^2_{L^2(\Omega)}, \tag{3.30}$$

where δ, γ are positive constants to be chosen. The solution $(\hat{\mathbf{u}}, \hat{\boldsymbol{\sigma}})$ is then the minimizer $(\hat{\mathbf{u}}, \hat{\boldsymbol{\sigma}}) = \arg\min_{(\mathbf{u},\boldsymbol{\sigma}) \in \mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_{\mathbf{g}_N}} \mathcal{F}_f(\mathbf{u}, \boldsymbol{\sigma})$. The minimizer can actually be computed as the solution of the following weak form: search for $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$, $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N}$ so that:

$$\gamma \int_\Omega \mathcal{A}\boldsymbol{\sigma} : \mathcal{A}\boldsymbol{\tau} + \delta \int_\Omega \text{div}\boldsymbol{\sigma} \cdot \text{div}\boldsymbol{\tau} - \gamma \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \mathcal{A}\boldsymbol{\tau} = -\delta \int_\Omega \mathbf{f} \cdot \text{div}\boldsymbol{\tau} \qquad \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}_0, \tag{3.31a}$$

$$-\gamma \int_\Omega \boldsymbol{\varepsilon}(\mathbf{v}) : \mathcal{A}\boldsymbol{\sigma} + \gamma \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v}) = 0 \qquad \forall \mathbf{v} \in \mathbf{U}_0. \tag{3.31b}$$

It is possible to define the following bilinear forms $a_f : \boldsymbol{\Sigma} \times \boldsymbol{\Sigma} \to \mathbb{R}$,

$b_f : \mathbf{U} \times \boldsymbol{\Sigma} \to \mathbb{R}$, $c_f : \mathbf{U} \times \mathbf{U} \to \mathbb{R}$ and the linear form $f_f : \boldsymbol{\Sigma} \to \mathbb{R}$:

$$a_f(\boldsymbol{\sigma}, \boldsymbol{\tau}) := \gamma \int_\Omega \mathcal{A}\boldsymbol{\sigma} : \mathcal{A}\boldsymbol{\tau} + \delta \int_\Omega \operatorname{div}\boldsymbol{\sigma} \cdot \operatorname{div}\boldsymbol{\tau} \,, \tag{3.32a}$$

$$b_f(\mathbf{u}, \boldsymbol{\tau}) := -\gamma \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \mathcal{A}\boldsymbol{\tau} \,, \tag{3.32b}$$

$$c_f(\mathbf{u}, \mathbf{v}) := \gamma \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v}) \,, \tag{3.32c}$$

$$f_f(\boldsymbol{\tau}) := -\delta \int_\Omega \mathbf{f} \cdot \operatorname{div}\boldsymbol{\tau} \,, \tag{3.32d}$$

so that the problem (3.31) becomes:

$$\begin{aligned} a_f(\boldsymbol{\sigma}, \boldsymbol{\tau}) + b_f(\mathbf{u}, \boldsymbol{\tau}) &= f_f(\boldsymbol{\tau}) \quad && \forall \boldsymbol{\tau} \in \boldsymbol{\Sigma}_0 \,, \\ b_f(\mathbf{v}, \boldsymbol{\sigma}) + c_f(\mathbf{u}, \mathbf{v}) &= 0 && \forall \mathbf{v} \in \mathbf{U}_0 \,, \end{aligned} \tag{3.33}$$

and the functional (3.30) can be rewritten as:

$$\mathcal{F}_f(\mathbf{u}, \boldsymbol{\sigma}) = a_f(\boldsymbol{\sigma}, \boldsymbol{\sigma}) + 2b_f(\mathbf{u}, \boldsymbol{\sigma}) + c_f(\mathbf{u}, \mathbf{u}) - f_f(\boldsymbol{\sigma}) \,. \tag{3.34}$$

Using the substitution (3.15) for both $\mathbf{u}$ and $\boldsymbol{\sigma}$, the system is transformed from a Babunov-Galerkin to a Ritz-Galerkin problem and becomes symmetric and positive definite. The same property could be maintained by adding to the functional another penalty term, $||\mathbf{as}\ \boldsymbol{\sigma}||^2$, which would enforce the condition for the symmetry of the stress. Nevertheless it has been showed in Cai, Zhiqiang and Starke, Gerhard [2004] that we can bound this term by means of the constitutive law, i.e.:

$$||\mathbf{as}\ \boldsymbol{\sigma}|| \leq 4\mu ||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})|| \,. \tag{3.35}$$

Therefore, if the pair $(\mathbf{u}, \boldsymbol{\sigma})$ makes $||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||$ small and μ is not too large, then also $||\mathbf{as}\ \boldsymbol{\sigma}||$ is small. Since the term $||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||$ is already added into the functional to be minimized, we may not need to add the condition on the symmetry of the stress. Therefore we can say that the first addendum in (3.30) is related to the angular momentum, while the second term to the linear momentum. It is then clear that, with this formulation, we do have a trade-off between the momenta that have to be conserved. In applications where the body is subject to an important bending, sometimes (3.30) does not well reflect the phenomenon. In particular, if μ is too large, then (3.35) is far from being sharp and the symmetry of $\boldsymbol{\sigma}$ is not properly prescribed. Therefore it can be usefull to extend the functional so that it penalize (3.1b):

$$\mathcal{F}_f(\mathbf{u}, \boldsymbol{\sigma}) = \gamma ||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||^2_{L^2(\Omega)} + \delta ||\operatorname{div}\boldsymbol{\sigma} + \mathbf{f}||^2_{L^2(\Omega)} + ||\mathbf{as}\ \boldsymbol{\sigma}||^2_{L^2(\Omega)} \,, \tag{3.36}$$

so that the conservation of angular momentum can be well captured. The weak form of this functional can be modified in order to improve the momentum balance law, as explained in Starke et al. [2011].

But the FOSLS functional can be changed also in other ways. For example, we can notice that, in the minimization of the functional (3.30), the unknown $(\mathbf{u}, \boldsymbol{\sigma})$ belongs to $\mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_{\mathbf{g}_N}$. Thus the boundary conditions are essential, because they are directly enforced into the space of the unknown. However, in the FOSLS formulations, the boundary conditions can be enforced also weakly. To this aim, it is sufficient to add them to the functional, as in (3.37):

$$\begin{aligned}\mathcal{F}_{f,bc}(\mathbf{u}, \boldsymbol{\sigma}) = &+ \gamma ||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||^2_{L^2(\Omega)} + \delta ||\mathrm{div}\boldsymbol{\sigma} + \mathbf{f}||^2_{L^2(\Omega)} \\ &+ \xi ||\mathbf{u} - \mathbf{u}_D||^2_{H^{1/2}(\Gamma_D)} + ||\boldsymbol{\sigma}\mathbf{n} - \mathbf{g}_N||^2_{H^{-1/2}(\Gamma_N)},\end{aligned} \tag{3.37}$$

where $\xi > 0$. Unfortunately the formulation (3.37) is cumbersome to deal with in practice. The $||\cdot||_{H^{1/2}(\Gamma_D)}$ and $||\cdot||_{H^{-1/2}(\Gamma_N)}$ norms would make the assembly of the related weak form not trivial. Then it is necessary to replace negative or fractional Sobolev norms with L^2 norms, transforming (3.37) into:

$$\begin{aligned}\mathcal{F}_{f,bc}(\mathbf{u}, \boldsymbol{\sigma}) = &+ \gamma ||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||^2_{L^2(\Omega)} + \delta ||\mathrm{div}\boldsymbol{\sigma} + \mathbf{f}||^2_{L^2(\Omega)} \\ &+ \xi ||\mathbf{u} - \mathbf{u}_D||^2_{L^2(\Gamma_D)} + ||\boldsymbol{\sigma}\mathbf{n} - \mathbf{g}_N||^2_{L^2(\Gamma_N)}.\end{aligned} \tag{3.38}$$

The trade-off between the right norm, negative or fractional, and the practicality of implementation is investigated in Bochev, Pavel B. and Gunzburger, Max D. [2009]. In Starke [1999] a multilevel technique for enforcing weakly the boundary conditions is proposed.

Remark 3.5.1. *In (3.36) no weight is used for the term $||\mathbf{as}\,\boldsymbol{\sigma}||^2_{L^2(\Omega)}$ because all the weights can be normalized. This means that one addendum can have a weight equal to one, while the other ones have to be chosen accordingly.*

3.5.2 Existence and uniqueness

As for the previous formulations, we want to exploit Theorem 2.4.6. The set $\mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_{\mathbf{g}_N}$ is a non empty closed convex set. The functional (3.34) is quadratic. Linearity and continuity of the forms in (3.32a), (3.32b), (3.32c), (3.32d) is trivial. For the discussion on the ellipticity of the (3.32a), (3.32b), (3.32c), we refer the reader to Cai, Zhiqiang and Starke, Gerhard [2004].

Chapter 4

The Signorini problem

In this chapter, we generalize the formulation of linear elasticity introduced in chapter 3 to unilateral contact problems. First we examine the strong formulation. Then we introduce the weak formulations of the primal, the dual and the FOSLS formulations.

4.1 Contact conditions

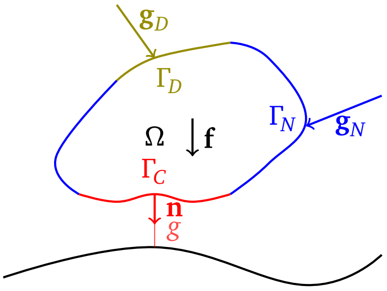


Figure 4.1. 2D Signorini's problem.

The linear elasticity problem (3.9) for a body Ω of Figure 3.1 is now generalized to the case in which it can interact with a rigid obstacle. See Figure 4.1. We say this problem is unilateral because the body obstacle is rigid. In case it was deformable as Ω, we would consider bilateral contact body problems, with two deformable bodies Ω_1 and Ω_2.

We now introduce a contact boundary Γ_C, an open subset of $\partial\Omega$, so that $\overline{\partial\Omega} = \overline{\Gamma_D} \cap \overline{\Gamma_N} \cap \overline{\Gamma_C}$ and $\Gamma_i \cap \Gamma_j = \emptyset$, for $i, j = C, D, N$ and $i \neq j$. Furthermore, we assume Γ_C to be surrounded by Γ_N. This is the portion of the boundary which could come into contact with the obstacle. We say *could* because we do not know, in advance, what subregion of Γ_C will be actually in contact. If we knew, we could simply prescribe a displacement and go back to the standard linear elasticity formulation. However, since this is not the case, a non-linearity is introduced into the problem.

Let us denote with g the distance in the normal direction between Γ_C and the obstacle. We denote with the subscripts n and t the normal and tangential components. So we define $u_n := \mathbf{u} \cdot \mathbf{n}$, $\sigma_n = \mathbf{n}^T \cdot (\boldsymbol{\sigma}\mathbf{n})$ and $(\boldsymbol{\sigma}\mathbf{n})_t := \boldsymbol{\sigma}\mathbf{n} - \sigma_n \mathbf{n}$. Then the frictionless contact conditions read as follows:

$$\begin{aligned} g - u_n &\leq 0 \quad \text{on } \Gamma_C\,, && (4.1a)\\ \sigma_n &\leq 0 \quad \text{on } \Gamma_C\,, && (4.1b)\\ \sigma_n(g - u_n) &= 0 \quad \text{on } \Gamma_C\,, && (4.1c)\\ (\boldsymbol{\sigma}\mathbf{n})_t &= \mathbf{0} \quad \text{on } \Gamma_C\,. && (4.1d) \end{aligned}$$

The first inequality (4.1a) implies that the body Ω can never penetrate the obstacle and for this reason it is called *non-penetration condition*. The second inequality (4.1b) states that, in case of contact, only negative pressure is permitted and no adhesion force can occur. The third equation (4.1c) is referred as *complementarity condition*. It says that the body can be subject to an external pressure only if it is in contact with the obstacle. The last equation (4.1d) is known as frictionless condition: we assume that no friction force can arise during the contact process. Considering (3.9) and the previous conditions (4.1a), (4.1b), (4.1c), (4.1d), we get the strong formulation of the Signorini problem, that in this context have to be interpreted in a pointwise manner.

In order to generalize the weak forms of linear elasticity by taking into account (4.1), it is convenient to modify the functional and the set on which the minimization has to be carried out. We will examine only the compressible primal, the dual, and the FOSLS formulations. Indeed, we can say that the incompressible primal formulation has more disadvantages than the dual and the FOSLS ones. First, it gives rise to a saddle point system -even though this is also true for the dual formulation-. But most importantly, it is not suited for nearly incompressible materials and does not give direct access to the stress variable, which still needs to be post-processed.
In conclusion, we introduce also the following set:

$$\boldsymbol{\Sigma}_t := \{\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N} : (\boldsymbol{\sigma}\mathbf{n})_t = \mathbf{0} \quad \text{on } \Gamma_C\}\,. \tag{4.2a}$$

4.2 The compressible displacement-based primal formulation

4.2.1 Minimization problem

In the primal formulation, the main variable is the displacement $\mathbf{u}$, therefore we just need to satisfy the contact conditions related to $\mathbf{u}$. The ones related to $\boldsymbol{\sigma}$ will automatically follow. Then the Signorini problem can be formulated as the minimization of the following quadratic functional $\mathcal{J}_p : \mathbf{U} \to \mathbb{R}$ over the closed convex set $\mathbf{K}_p \subset \mathbf{U}_{\Gamma_D}$:

$$\mathcal{J}_p(\mathbf{u}) = \frac{1}{2} a_p(\mathbf{u},\mathbf{u}) - f_p(\mathbf{u})\,, \tag{4.3a}$$

$$\mathbf{K}_p = \{\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D} : u_n \leq g \text{ on } \Gamma_C\}\,. \tag{4.3b}$$

The solution $\hat{\mathbf{u}}$ is then the minimizer $\hat{\mathbf{u}} = \arg\min_{\mathbf{u}\in\mathbf{K}_p} \mathcal{J}_p(\mathbf{u})$.

4.2.2 Existence and uniqueness

The functional (4.3a) is the same of the linear case (3.16). The difference between the two formulations resides in the set $\mathbf{K}_p$ that can be easily shown to be a non-empty closed convex set. Thus Theorem 2.4.6 holds. See Kikuchi and Oden [1988].

4.3 The stress-based dual formulation

4.3.1 Minimization problem

The Signorini problem can be formulated as the minimisation of the following quadratic functional $\mathcal{J}_d : \boldsymbol{\Sigma} \to \mathbb{R}$ over the closed convex set $\mathbf{K}_d \subset \boldsymbol{\Sigma}_{\mathbf{g}_N}$:

$$\mathcal{J}_d(\boldsymbol{\sigma}) := \frac{1}{2} a_d(\boldsymbol{\sigma},\boldsymbol{\sigma}) - f_{d,1}(\boldsymbol{\sigma}) - \langle g, \sigma_n \rangle_{\Gamma_C}\,, \tag{4.4a}$$

$$\begin{aligned} \mathcal{J}_d := \{\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_t :\ & b_d(\boldsymbol{\sigma},\mathbf{v}) && = f_{d,2}(\mathbf{v}) && \forall \mathbf{v} \in \mathbf{V}, \\ & c_d(\boldsymbol{\sigma},\boldsymbol{\gamma}) && = 0 && \forall \boldsymbol{\gamma} \in \boldsymbol{\Theta}, \\ & \sigma_n && \leq 0 && \text{on } \Gamma_C\}\,. \end{aligned} \tag{4.4b}$$

The solution $\hat{\boldsymbol{\sigma}}$ is then the minimizer $\hat{\boldsymbol{\sigma}} = \arg\min_{\boldsymbol{\sigma}\in\mathbf{K}_d} \mathcal{J}_d(\boldsymbol{\sigma})$. The equality constraints div $\boldsymbol{\sigma} = -\mathbf{f}$ and **as** $\boldsymbol{\sigma} = \underline{0}$ are respectively enforced by Lagrange multipliers, $\mathbf{v} \in \mathbf{V}$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. With respect to (3.22), we have enriched the minimization set by adding the frictionless contact condition (4.1d) and the negative pressure condition (4.1b) on Γ_C; we have also modified the functional by defining $\mathcal{J}_d(\boldsymbol{\sigma}) := \mathcal{F}_d(\boldsymbol{\sigma}) - \langle \sigma_n, g \rangle_{\Gamma_C}$.

4.3.2 Existence and uniqueness

The functional $\mathcal{J}_d$, with respect to $\mathcal{F}_d$ in (4.4), has an additional term that is linear and continuous. Furthermore the set of the dual Signorini problem is a subset of the linear elasticity case: $\mathbf{K}_d \subset \mathcal{S}_d$. As explained in Kober, Bernhard [2020], $\mathbf{K}_d$ is still a non-empty closed convex set, thus Theorem 2.4.6 holds.

4.4 FOSLS formulation

4.4.1 Minimization problem

For the Signorini problem applied to the FOSLS case, since both $\mathbf{u}$ and $\boldsymbol{\sigma}$ are main variables of the problem, we must satisfy explicitly (4.1a), (4.1b), (4.1c), (4.1d). The frictionless condition (4.1d) is a pointwise equality constraint and thus it is easy to handle. The two relations (4.1a), (4.1b) are inequalities, but linear and pointwise. On the other hand, the term (4.1c) is pointwise but non-linear. Therefore including all the contact conditions in the closed convex set on which we minimize the functional would make it too complicated for numerical simulations. An easier solution is to define the convex set by means of the only linear conditions (4.1a), (4.1b), (4.1d), while the quadratic term (4.1c) can be added to the functional $\mathcal{F}_f$, so that we obtain an augmented FOSLS functional $\mathcal{J}_f(\mathbf{u}, \boldsymbol{\sigma}) := \mathcal{F}_f(\mathbf{u}, \boldsymbol{\sigma}) + \langle \sigma_n, u_n - g \rangle_{\Gamma_C}$.

The FOSLS Signorini problem then is: find $\mathbf{u} \in \mathbf{U}_{\mathbf{g}_D}$, $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_{\mathbf{g}_N}$, that minimize the FOSLS functional $\mathcal{J}_f$ over the closed convex set $\mathbf{K}_f \subset \mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_{\mathbf{g}_N}$:

$$\mathcal{J}_f(\mathbf{u}, \boldsymbol{\sigma}) = \gamma ||\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})||^2_{L^2(\Omega)} + \delta ||\mathrm{div}\boldsymbol{\sigma} + \mathbf{f}||^2_{L^2(\Omega)} + \langle \sigma_n, u_n - g \rangle_{\Gamma_C}, \tag{4.5a}$$

$$\mathbf{K}_f = \{(\mathbf{u}, \boldsymbol{\sigma}) \in \mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_t : u_n \leq g, \ \sigma_n \leq 0 \ \text{ on } \Gamma_C\}, \tag{4.5b}$$

where also in this case δ, γ are positive constants to be chosen. The solution $(\hat{\mathbf{u}}, \hat{\boldsymbol{\sigma}})$ is then the minimizer $(\hat{\mathbf{u}}, \hat{\boldsymbol{\sigma}}) = \arg\min_{(\mathbf{u},\boldsymbol{\sigma}) \in \mathbf{K}_f} \mathcal{J}_f(\mathbf{u}, \boldsymbol{\sigma})$.

4.4.2 Existence and uniqueness

The set $\mathbf{K}_f$ is a subset of $\mathbf{U}_{\mathbf{g}_D} \times \boldsymbol{\Sigma}_{\mathbf{g}_N}$ and it is still non-empty closed and convex. What differs in the functional $\mathcal{J}_f$, with respect to $\mathcal{F}_f$ in (3.30), is the presence of the complementarity term. The corresponding bilinear and linear forms are continuous. However, even if it is clear that, if evaluated on $\mathbf{K}_f$, the functional is non-negative, the corresponding bilinear forms are not necessarily elliptic. Indeed, as we show in section 7.1, the weights γ and δ are not arbitrary anymore but need to satisfy given relations.

Chapter 5

The Finite Element method for the Signorini problem

In this chapter, we discuss the Finite Element (FE) method. We define the local and global FE spaces for the linear Lagrangian family and the zero-order and first-order Raviart-Thomas families. In this way, the continuous weak form for the primal, the dual, and the FOSLS formulations can be discretized.

5.1 The Finite Element method

In order to solve the aforementioned continuous problems (4.3), (4.4), (4.5), their discretization is required. Indeed, except for some specific cases, an explicit formula for the exact solution has not been discovered yet. Thus, let us consider a given Hilbert space **H**. We introduce a sequence of nested subspaces $\{\mathbf{H}_j\}_{j=1}^{\infty}$ such that:

$$\mathbf{H}_j \subset \mathbf{H} \quad \forall j, \qquad \overline{\underset{\mathrm{j}\in\mathbb{N}}{\cup}\mathbf{H}_{\mathrm{j}}}^{\|\cdot\|_{\mathbf{H}}} = \mathbf{H}\,, \tag{5.1}$$

where the symbol $^{\|\cdot\|_{\mathbf{H}}}$ is the closure with respect to the norm of the Hilbert space **H**. A nested sequence of subspaces of this kind is said to be **H**-conforming. Indeed each space $\mathbf{H}_j$ is actually a subspace of **H** and, by taking the limit, its closure coincide with **H** itself. This means that, by approximating **H** with $\mathbf{H}_j$, the approximate solution belongs to a subspace of the the space of the real unknown. Furthermore, by taking increasingly richer spaces $\mathbf{H}_j$, the approximation becomes more and more accurate.

In order to introduce conforming discrete spaces, we opt for the Finite Element (FE) method. See Ciarlet, Philippe G. [2002], Braess, Dietrich [2007], Quarteroni, Alfio and Quarteroni, Silvia [2009], Rognes, Marie E. and Kirby, Robert C. and Logg, Anders [2009]. Let $\mathcal{T}_h = \{K_1, ..., K_{N_e}\}$ be a shape-regular simplicial mesh of Ω with N_e elements. The general simplex K is a triangle, in 2D, or a tetrahedron, in 3D. The subscript h represents the maximal diameter of $\mathcal{T}_h$. We also denote the faces of each simplex, a segment in 2D or a triangle in 3D, by means of the capital letter f. The set of faces of the mesh is $\mathcal{F}_h = \{f_1, ..., f_{n_f}\}$, where n_f is the number of the faces. We also denote the set of vertices of the mesh with $\mathcal{P}_h = \{\mathbf{p}_1, ..., \mathbf{p}_N\}$.

We need to discretize all the linear spaces of (2.21), (2.22), (2.23) on the mesh $\mathcal{T}_h$. Now the discretization parameter is identified by the maximal diameter of the mesh h. Therefore the discretized versions of $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{\Theta}$, $\mathbf{V}$ are $\mathbf{U}_h$, $\mathbf{\Sigma}_h$, $\mathbf{\Theta}_h$, $\mathbf{V}_h$. If a function belongs to one of the now approximated continuous spaces, it can be discretized as well. The functions of the problem, such as $\mathbf{n}$, $\mathbf{f}$, $\mathbf{g}_D$, $\mathbf{g}_N$, g, are discretized as $\mathbf{n}_h$, $\mathbf{f}_h$, $\mathbf{g}_{h,D}$, $\mathbf{g}_{h,N}$, g_h. In this way it is also possible to obtain the discrete sets $\mathbf{U}_{h,\mathbf{g}_{h,D}}$, $\mathbf{\Sigma}_{h,\mathbf{g}_{h,N}}$. And the same goes with the discrete closed convex set $\mathbf{K}_{h,i}$ and the discrete functionals $\mathcal{J}_{h,i}$ for $i = p,\ d,\ f$. The bilinear and linear forms are consequently approximated and their discretizations are denoted with the subscript h.

We assume the discrete spaces to be conforming, so we have $\mathbf{U}_h \subset \mathbf{U}$, $\mathbf{\Sigma}_h \subset \mathbf{\Sigma}$, $\mathbf{\Theta}_h \subset \mathbf{\Theta}$, $\mathbf{V}_h \subset \mathbf{V}$. However, if the space is also subject to equality or inequality constraints, the corresponding discrete set is effectively a subset of its continuous version only if the discretized constraints coincide with their continuous versions. This scenario is true, for example, for constant functions. Their discrete versions are identical to the continuous ones. However for more complicated data, the inclusion between the discrete and the continuous sets does not hold anymore. In particular we can say this is true for the closed convex sets:

$$\mathbf{K}_{h,i} \not\subset \mathbf{K}_i \qquad i = p,\ d,\ f\ . \tag{5.2}$$

If $\mathbf{K}_{h,i}$ is characterized by a box-constraint, this one is typically approximated only pointwise and interpolated linearly. With this in mind, we can now translate the formulations (4.3), (4.4), (4.5) into their discrete counterparts. Before proceeding in this direction, it is convenient to introduce conforming FE spaces for $H^1(\Omega)$ and $H_{\text{div}}(\Omega)$.

5.2 Local FE spaces

Many are the books treating the finite element method, like Brezzi and Fortin [2012], Brenner, Susanne and Scott, Ridgway [2007], Quarteroni, Alfio and

Quarteroni, Silvia [2009]. Here we want to exploit the definition from Ciarlet, Philippe G. [2002].

Definition 8. *The finite element is a triplet $\mathcal{S} = (K, \mathcal{V}, \mathcal{L})$ such that:*

- *K is a closed convex subspace of $\mathbb{R}^d$ with a non-empty interior and Lipschitz-continous boundary.*
- *$\mathcal{V}(K)$ is a space of continuous functions defined on K of dimension $n_{\mathcal{V}(K)} = dim(\mathcal{V}(K))$.*
- *$\mathcal{L}(K) = \{l_{K,1}, ..., l_{K,n_{\mathcal{V}(K)}}\}$ is a set of linear functionals, named degrees of freedom (dofs). This set $\mathcal{L}(K)$ is $\mathcal{V}(K)$-unisolvent, meaning that for any set of scalars $\{\alpha_1, \dots, \alpha_{n_{\mathcal{V}(K)}}\}$ there exist a unique $\phi \in \mathcal{V}(K)$ such that $l_{K,i}(\phi) = \alpha_i$ for $i = 1, \dots, n_{\mathcal{V}(K)}$. We then call basis functions the set of functions $\{\phi_1, \dots, \phi_{n_{\mathcal{V}(K)}}\}$ such that:*

$$l_{K,i}(\phi_j) = \delta_{ij} \qquad i, j = 1, \dots, n_{\mathcal{V}(K)}. \tag{5.3}$$

Usually, the space $\mathcal{V}(K)$ is defined by means of polynomials, so it is also useful to introduce the following space:

$$P_q(K) := \{\text{the space of polynomials of degree} \leq q \text{ on K}\} \tag{5.4}$$

and

$$Q_q(K) := \{\text{the space of homogeneous polynomials of degree } q \text{ on K}\}\,. \tag{5.5}$$

Furthermore given a function space V and $v \in V$ sufficiently regular, we can define an interpolation operator $I_K : V \to \mathcal{V}(K)$ such that:

$$I_K(v) = \sum_{i=1}^{n_{\mathcal{V}(K)}} v_i \phi_i \qquad \text{where } v_i = l_{K,i}(v), \text{ for } i = 1, \dots, n_{\mathcal{V}(K)}\,. \tag{5.6}$$

This implies that, for $v = v_h \in \mathcal{V}(K)$, we can state:

$$v_h = \sum_{i=1}^{n_{\mathcal{V}(K)}} v_i \phi_i \qquad \text{where } v_i = l_{K,i}(v_h), \text{ for } i = 1, \dots, n_{\mathcal{V}(K)}\,. \tag{5.7}$$

Typically, for convenience of implementation, it is also introduced the so-called reference element $\hat{K} \subset \mathbb{R}^d$. We will use the hat superscript "^" to denote all the quantities defined on the reference element. Thus we can also define a finite element space $\hat{\mathcal{S}} = (\hat{K}, \hat{\mathcal{V}}, \hat{\mathcal{L}})$. In order to relate $\hat{\mathcal{S}}$ to $\mathcal{S}$, we need a geometric map for the points in $\hat{K}$ and a function map for the basis functions. The reference element $\hat{K}$ is related to a generic element K by means of an affine transformation $F : \hat{K} \to K$ which associates to each point $\hat{\mathbf{x}} \in \hat{K}$ a unique point $\mathbf{x} \in K$. Indeed it is possible to define $\mathbf{J} \in \mathbb{R}^{d,d}$ and $\mathbf{j} \in \mathbb{R}^d$ so that:

$$\mathbf{x} = F(\hat{\mathbf{x}}) := \mathbf{J}\hat{\mathbf{x}} + \mathbf{j}\,. \tag{5.8}$$

It is also desirable to map the reference basis functions from $\hat{K}$ to K. However it can also be important to map their gradient, divergence, curl and so on. Briefly, we would like to map the reference basis functions $\{\hat{O}(\hat{\phi}_i)\}_{i=1}^{n_{\mathcal{V}(K)}}$ of the FE space $\hat{\mathcal{S}}$ subject to an operator $\hat{O}$ (it can be the identity, the gradient, the divergence and so on). We introduce the map $M_{\mathcal{S},O}$:

$$\begin{array}{llll} M_{\mathcal{S},O}: & \hat{O}(\mathcal{V}(\hat{K})) & \to & O(\mathcal{V}(K))\,, \\ M_{\mathcal{S},O}: & \hat{O}(\hat{\phi}) & \mapsto & O(\phi)\,. \end{array} \tag{5.9}$$

In this way, it is possible to transform the set $\{\hat{O}(\hat{\phi}_1), \ldots, \hat{O}(\hat{\phi}_{n_{\mathcal{V}(K)}})\}$ on $\hat{K}$ into $\{O(\phi_1), \ldots, O(\phi_{n_{\mathcal{V}(K)}})\}$ on K.

The advantage of defining a reference element $\hat{K}$ is the possibility of precomputing important quantities that can be later mapped onto a generic K, whenever it is needed. For example, the numerical computation of an integral on K, involving shape functions $\phi_i \in \mathcal{V}(K)$, for $i = 1, \ldots, n_{\mathcal{V}(K)}$, can be reduced to a quadrature integral, i.e., a weighted sum involving the evaluation of ϕ_i in certain quadrature points $\{\mathbf{q}_j\}$, with $\mathbf{q}_j \in K$. Then, if the position of the quadrature points is fixed, it is not necessary to build ϕ_i and then to evaluate it. It is sufficient to precompute $\hat{O}(\hat{\phi}_i(F^{-1}(\mathbf{q}_j)))$ once for all and then to map it back onto K by means of the map $M_{\mathcal{S},O}$.

In case of a simplicial mesh, the reference element $\hat{K}$ is the simplex defined as:

$$\hat{K} := \{(x_1, \ldots, x_d) \in \mathbb{R}^d : \quad \sum_{i=1}^{d} x_i = 1, \quad x_i \geq 0, \quad i = 1, \ldots, d\}, \tag{5.10}$$

that is the simplex with vertices $(\mathbf{p}_i)_j = \delta_{ij}$.

5.3 Global FE spaces

The definition of a local finite element is not enough. We need to decide how to glue together the information among different elements. A global finite element space over the mesh $\mathcal{T}_h$ is a space consisting of functions v_h that, if restricted to each $K \in \mathcal{T}_h$, belong to $\mathcal{V}(K)$. It can happen that, given two elements $K_1, K_2 \in \mathcal{T}_h$ with $K_1 \neq K_2$, two different degrees of freedom $l_{K_1,i}$ and $l_{K_2,j}$ are equal: $l_{K_1,i} = l_{K_2,j}$ for $i \in \{1,\dots,n_{\mathcal{V}(K_1)}\}$, $j \in \{1,\dots,n_{\mathcal{V}(K_2)}\}$. For example, in the Lagrangian case, if the dofs are defined as the evaluation of the function v_h in given points like a vertex shared between two adjacent cells, then the two dofs coincide. We can then decide to enforce or not a dof-continuity of v_h, meaning that if the two dofs are identical, then we enforce their evaluation on v_h to be also identical: $l_{K_1,i}(v_h) = l_{K_2,j}(v_h)$. By doing so, we obtain a continuous Lagrangian space, otherwise, we recover a discontinuous Lagrangian space. The same argument holds for the general definition of degree of freedom.

Definition 9. *We define the dof-continuous space $V(\mathcal{T}_h)$ and the dof-discontinuous space $DV(\mathcal{T}_h)$ as follows:*

$$\begin{aligned} V(\mathcal{T}_h) := \{v_h \in L^2(\Omega_h) :\ & v_h|_K \in \mathcal{V}(K)\ \forall K \in \mathcal{T}_k, \\ & \exists K_1, K_2 \in \mathcal{T}_h,\ i \in \{1,\dots,n_{\mathcal{V}(K_1)}\},\ j \in \{1,\dots,n_{\mathcal{V}(K_2)}\} : \\ & l_{K_1,i} = l_{K_2,j}\ \Rightarrow\ l_{K_1,i}(v_h) = l_{K_2,j}(v_h)\}\,, \end{aligned} \tag{5.11}$$

$$DV(\mathcal{T}_h) := \{v_h \in L^2(\Omega_h) :\quad v_h|_K \in \mathcal{V}(K)\ \forall K \in \mathcal{T}_k\}\,. \tag{5.12}$$

We can also set $n = \dim(V(K))$ or $n = \dim(DV(K))$, not to be confused with the local dimension of the FE space $n_{\mathcal{V}(K)}$. The local decomposition (5.7) can be extended also for functions $v_h \in V(\mathcal{T}_h)$ or $v_h \in DV(\mathcal{T}_h)$. To this aim, we need to define a global set of dofs $\mathcal{L}(\mathcal{T}_h)$. For $v_h \in DV(\mathcal{T}_h)$, $\mathcal{L}(\mathcal{T}_h)$ is simply the union of all local dofs. For $v_h \in V(\mathcal{T}_h)$, on the other hand, $\mathcal{L}(\mathcal{T}_h)$ is the union of all local dofs without ripetition. This means that, if two local dofs satisfy $l_{K_1,i} = l_{K_2,j}$, then there is only a unique global dof representing them. Therefore, for a function $v \in V$ sufficiently regular, we can define the global interpolation operator $I_h : V \to V(\mathcal{T}_h)/DV(\mathcal{T}_h)$:

$$I_h(v) = \sum_{i=1}^{n} v_i \phi_i \qquad \text{where} \quad v_i = l_i(v), \quad l_i \in \mathcal{L}(\mathcal{T}_h)\,. \tag{5.13}$$

Similarly a function $v_h \in V(\mathcal{T}_h)/DV(\mathcal{T}_h)$ can be decomposed as:

$$v_h = \sum_{i=1}^{n} v_i \phi_i \qquad \text{where} \quad v_i = l_i(v_h), \quad l_i \in \mathcal{L}(\mathcal{T}_h)\,. \tag{5.14}$$

If it holds that $V(\mathcal{T}_h) \subset V$ and $\lim_{h\to 0} V(\mathcal{T}_h) = V$, like in (5.1), then we say that $V(\mathcal{T}_h)$ is V-conforming. The same we would say for $DV(\mathcal{T}_h)$. In the next sections, we introduce the continuous Lagrange FE space, which is $H^1(\Omega)$-conforming, and the continuous Raviart-Thomas FE space, which is $H_{\text{div}}(\Omega)$-conforming. We will also consider the discontinuous case of the Lagrange space, which will be useful for the dual formulation.

5.4 The Lagrangian finite element space

Definition 10. *Given a set of points $\{\boldsymbol{p}_i\}_{1\le i\le n(q)}$ with $\boldsymbol{p}_i \in K$, the local Lagrangian space of order q, denoted by $CG^q(K) := (K, \mathcal{V}, \mathcal{L})$, is defined as:*

$$CG^q(K) := \begin{cases} K \text{ is a simplex} \\ \mathcal{V} = P_q(K) \\ \mathcal{L} = \{l_{K,i}(\lambda) = \lambda(\boldsymbol{p}_i)\,, \quad 1 \le i \le n_{\mathcal{V}}(q)\} \end{cases}, \tag{5.15}$$

where:

$$n_{\mathcal{V}}(q) = \prod_{i=1}^{d} \frac{q+i}{i} \tag{5.16}$$

For the reference simplex $\hat{K}$ in (5.10), we can define the points $\{\hat{\mathbf{p}}_i\}_{i=1}^{n_{\mathcal{V}}(q)}$ as follows:

$$\hat{\mathbf{p}} := \frac{1}{q}(p_1, \dots, p_d) \qquad 0 \le \sum_{i=1}^{d} p_i \le q, \qquad p_i \in \mathbb{N}, \quad i = 1, \dots, d \tag{5.17}$$

We define the mapping $M_{CG^q,O}$ for the identity and gradient operators $O = I, \nabla$ as:

$$M_{CG^q,I} := 1 \qquad M_{CG^q,\nabla} := \mathbf{J}^{-T} \tag{5.18}$$

where $\mathbf{J}$ is the matrix in (5.8).
We denote the continuous global FE space of Lagrangian functions of order q on $\mathcal{T}_h$ with $P_q(\mathcal{T}_h)$, while its discontinuous version with $\mathrm{D}P_q(\mathcal{T}_h)$. If the dependency on $\mathcal{T}_h$ is obvious from the contex, we will omit it.

We conclude this section with the following theorem.

Theorem 5.4.1. *Let $\mathcal{T}_h$ be a shape-regular triangulation. Then there exists an interpolation operator $I_h^q : H^m(\Omega) \to P_q(\mathcal{T}_h)$ and a constant C indipendet of h such that:*

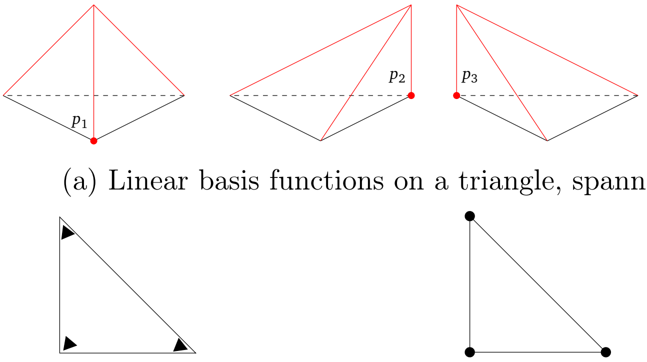
$$||v - I_h^q v||_{L^2(\Omega)} \le C h^{m-s} |v|_{H^m(\Omega)} \quad 0 \le s \le m, \quad 2 \le m \le q+1 \tag{5.19}$$

5.4.1 The local linear Lagrangian space

For the linear Lagrangian element, the points $\{\mathbf{p}_i\}$ correspond to the vertices of the simplex. Thus there are $d+1$ linear polynomial shape functions that are independent and which build a basis for $\mathcal{V}(K)$. We have a 2D representation in Figure 5.1a. Their dofs representation in the global finite element space, for discontinuous and continuous piecewise linear functions, is given respectively in Figure 5.1b and in Figure 5.1c. The black triangles, inside each element, suggest the independence of the degree of freedom from the other elements, while the black circles, which can overlap more elements, show the continuity of the global function.

We remind the reader that, if $\mathbf{b} \in K$ is the barycenter of K and λ_i for $i = 1, \ldots, d+1$ are the linear Lagrangian shape functions related to the vertex $\mathbf{p}_i$, then:

$$\lambda_1|_{\mathbf{b}} = \ldots = \lambda_{d+1}|_{\mathbf{b}} = \frac{1}{d+1}\,. \tag{5.20}$$



(a) Linear basis functions on a triangle, spanning $P_1(K)$.

(b) P_1 discontinuous dofs. (c) P_1 continuous dofs.

Figure 5.1. 2D linear Lagrange FE space.

5.5 The local Raviart-Thomas finite element space

A space Q_h, in order to be a conforming finite element approximation space of $H_{\mathrm{div}}(\Omega)$, must contain functions $\mathbf{q}_h$ in $L^2(\Omega)$ such that div $\mathbf{q}_h \in \mathbf{L}^2(\Omega)$. A sufficient condition for this to happen is that:

$$\begin{aligned} \mathbf{q}_h|_K &\in H_{\mathrm{div}}(K) \qquad \forall K \in \mathcal{T}_h\,, && (5.21a)\\ \left[\mathbf{q}_h \cdot \mathbf{n}\right]_f &= 0 \qquad \forall f \in \mathcal{F}_h\,, && (5.21b)\end{aligned}$$

where we have denoted the jump across the face f of two adjacent elements $K_i, K_j \in \mathcal{T}_h$ with $\left[\mathbf{q}_h \cdot \mathbf{n}\right]_f = \mathbf{q}_h \cdot \mathbf{n}|_{f\cap K_i} - \mathbf{q}_h \cdot \mathbf{n}|_{f\cap K_j}$. If $\phi \in \mathcal{D}(\Omega)$ —see section 2.1 for the definition —, then $\int_\Omega \mathbf{q}_h \cdot \nabla\phi$ is bounded due to Cauchy-Schwarz and the fact that $\mathbf{q}_h \in \mathbf{L}^2(\Omega)$. Thus, exploiting the Green's formula (2.19) and the hypotheses (5.21a), (5.21b) we can write:

$$\begin{aligned} \int_\Omega \mathbf{q}_h \cdot \nabla\phi &= \sum_{K\in\mathcal{T}_h} \int_K \mathbf{q}_h \cdot \nabla\phi && (5.22a)\\ &\overset{(2.19)}{=} \sum_{K\in\mathcal{T}_h} \left(\sum_{f\in\partial K} (\mathbf{q}_h \cdot \mathbf{n}_f)\phi - \int_K \mathrm{div}\mathbf{q}_h \phi \right) && (5.22b)\\ &= \sum_{f\in\mathcal{F}_h} \left[(\mathbf{q}_h \cdot \mathbf{n}_f)\right]\phi - \sum_{K\in\mathcal{T}_h} \int_K \mathrm{div}\mathbf{q}_h\phi && (5.22c)\\ &\overset{(5.21b)}{=} - \sum_{K\in\mathcal{T}_h} \int_K \mathrm{div}\mathbf{q}_h\phi && (5.22d)\\ &= -\int_\Omega \mathrm{div}\mathbf{q}_h\phi\,. && (5.22e)\end{aligned}$$

The expression at the second line (5.22b) is bounded because of (5.21a) and, in the next steps, because of (5.21b). Since the first term in (5.22a) is bounded and equal to the last one (5.22e), also this one is bounded. One can take $\phi = \mathrm{div}\mathbf{q}_h$ and show that $\mathrm{div}\mathbf{q}_h \in L^2(\Omega)$.

The condition (5.21a) can be satisfied by considering a local space $\mathcal{V}(K)$ of vector-valued polynomials. The second condition (5.21b) requires $\mathbf{q}_h$ to have continuous normal flux between two adjacent elements and can be ensured only by a proper construction of the local FE space. The Raviart-Thomas finite element space does fulfill such a property.

Definition 11. *The Raviart-Thomas finite element space of order q, for $q = 0, 1, \ldots$,* *on a simplex K, is the space* $RT_q(K) := (K, \mathcal{V}, \mathcal{L})$ *defined as:*

$$RT_q(K) := \begin{cases} K \textit{ is a simplex} \\ \mathcal{V} = \left[P_q(K)\right]^d + \boldsymbol{x}P_q(K) \\ \mathcal{L} = \begin{cases} \int_{f_i} \boldsymbol{\phi} \cdot \boldsymbol{n}\, p\, d\boldsymbol{s} & p \in P_q(f_i), \textit{for each facet } f_i \textit{ of } K \\ \int_K \boldsymbol{\phi} \cdot p\, d\boldsymbol{x} & p \in \left[P_{q-1}(K)\right]^d, q \geq 1 \end{cases} \end{cases}, \tag{5.23}$$

with dimension:

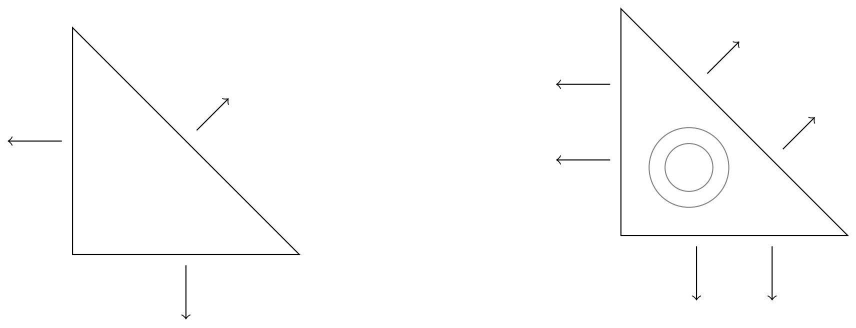
$$n_{\mathcal{V}}(q) = \begin{cases} (q+1)(q+3) & T = \textit{triangle} \\ \frac{1}{2}(q+1)(q+2)(q+4) & T = \textit{tetrahedron}\,. \end{cases} \tag{5.24}$$

We define the mapping $M_{RT_q,O}$ for the identity and the divergence operators $O = I, \text{div}$ as:

$$M_{RT_q,I} := \frac{\mathbf{J}}{\det(\mathbf{J})} \qquad M_{RT_q,\text{div}} := \frac{1}{\det(\mathbf{J})}\,, \tag{5.25}$$

where $M_{RT_q,I}$ is the so-called Piola transform. The symbolic representations in 2D for the cases $q = 0, 1$ are given in Figure 5.2a and Figure 5.2b. The arrows represent the face degrees of freedom, while the internal circles the momentum ones.

In the next paragraphs, we explain how to explicitly build the Raviart-Thomas shape functions for the cases $q = 0, 1$ and for a d-dimensional simplex.



(a) RT_0: one face dof per face.

(b) RT_1: two face dofs per face and two momentum dofs.

Figure 5.2. 2D RT dofs.

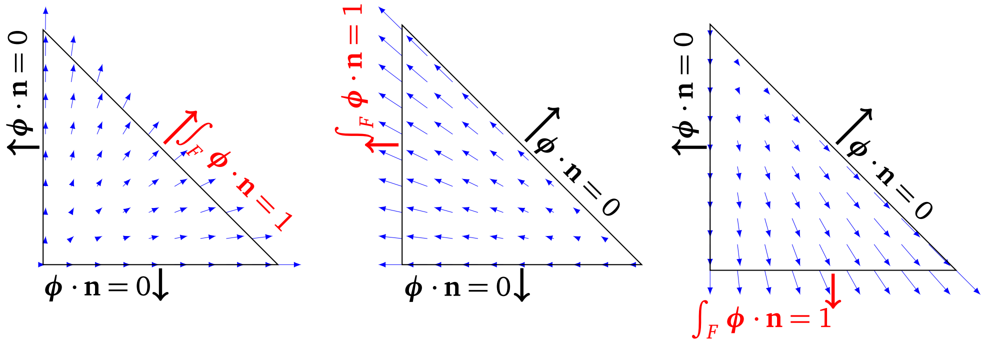

$\boldsymbol{\phi}\cdot\mathbf{n}=0$
$\int_F \boldsymbol{\phi}\cdot\mathbf{n}=1$
$\boldsymbol{\phi}\cdot\mathbf{n}=0$
$\int_F \boldsymbol{\phi}\cdot\mathbf{n}=1$
$\boldsymbol{\phi}\cdot\mathbf{n}=0$
$\boldsymbol{\phi}\cdot\mathbf{n}=0$
$\boldsymbol{\phi}\cdot\mathbf{n}=0$
$\boldsymbol{\phi}\cdot\mathbf{n}=0$
$\int_F \boldsymbol{\phi}\cdot\mathbf{n}=1$

Figure 5.3. Representation on the reference triangle of RT_0 shape functions. The arrows have been scaled by a factor of 0.1.

5.5.1 The local space $RT_0(K)$ on a simplex

In d-dimensions, $RT_0(K)$ has exactly $d+1$ degrees of freedom, one for each facet of the simplex. For defining RT_0 shape basis functions, we can just use the Definition 11, considering $q=0$:

$$RT_0(K) := \begin{cases} K \text{ is a simplex} \\ \mathcal{V} = [P_0(K)]^d + \mathbf{x}P_0(K) \\ \mathcal{L} = \int_{f_i} \boldsymbol{\phi}\cdot\mathbf{n}\, d\mathbf{s} \quad \text{for each facet } f_i \text{ of K}. \end{cases} \tag{5.26}$$

The shape function related to the face f_i of K with opposite vertex $\mathbf{p}_i$, for $i=1,...,d+1$, can be expressed as:

$$\boldsymbol{\phi}_i(\mathbf{x}) = \frac{1}{d|K|}(\mathbf{x}-\mathbf{p}_i) \qquad \forall \mathbf{x}\in K, \quad i=1,\dots,d+1, \tag{5.27}$$

where $|K|$ is the measure of K. In 2D, they look like in Figure 5.3.

5.5.2 The local space RT_1 on a simplex

In d-dimensions, the simplicial RT_1 space has d degrees of freedom per facet and d momentum degrees of freedom. The definition (5.23) can be used for higher-order Raviart-Thomas finite elements only for a theoretical purpose. Thus, for practical implementation of RT_1, we need an alternative way to define the dofs. For example, in Kober, Bernhard [2017] the dofs are defined in a “Lagrangian” way since they are pointwise evaluations. This makes the computation of the corresponding shape functions easier.

Given a facet f_i, let $\mathbf{p}_{i,k}$ for $k = 1,\ldots,d$ denote its vertices, while $\mathbf{b}$ is the barycenter of K; the vectors $\mathbf{e}_1 = (1,0,...)$, $\mathbf{e}_2 = (0,1,0,...)$,..., $\mathbf{e}_d = (...,0,1)$ represent the canonical basis of $\mathbb{R}^d$. We then define:

$$\begin{cases} K \text{ is a simplex} \\ \mathcal{V} = [P_1(K)]^d + \mathbf{x}P_1(K) \\ \mathcal{L} = \begin{cases} \boldsymbol{\psi}\cdot\mathbf{n}|_{\mathbf{p}_{i,k}} & \text{for each vertex } k = 1,\ldots,d \text{ of the face } f_i \text{ of K} \\ \boldsymbol{\psi}\cdot\mathbf{e}_k|_{\mathbf{b}_K} & \text{for } k = 1,\ldots,d \end{cases} \end{cases} . \tag{5.28}$$

For a 2D triangle, the shape functions are represented in Figure 5.4. We want to stress out the fact that, by mapping the reference RT_1 shape functions from $\hat{K}$ to K with the Piola transformation in (5.25), we obtain $\boldsymbol{\phi}(\mathbf{x}) = \dfrac{\mathbf{J}}{\det\mathbf{J}}\hat{\boldsymbol{\phi}}(\hat{\mathbf{x}})$, $\forall\hat{\mathbf{x}}\in\hat{K}$. However, if it is ensured that $\hat{l}_i(\hat{\boldsymbol{\phi}}_j) = \delta_{ij}$ for $i,j = 1,\ldots,n_{\mathcal{V}}$, it can happen that $l_i(\boldsymbol{\phi}_j) \neq \delta_{ij}$. This is because the mapping is the one for Raviart-Thomas space, but the way we have defined the degrees of freedom is intrinsically Lagrangian, since it involves the evaluation of the function in given points. So the relation (5.3) is not necessarily ensured on K. This can be a problem in the definition of the interpolation operators, that requires the relation (5.3) to be satisfied on K. So, in the case we have no interest in an interpolation-like operator, the mapping can still be used. Otherwise, the definitions given for RT_1 basis functions in (5.28) must be directly enforced on K, without the mapping between $\hat{K}$ and K.

Thus we want to show how the RT_1 basis functions on a generic simplex can be computed. We start from the 2D case and we then generalize for a simplex in d-dimensions. Let us consider a 2D triangle with the vertices and the barycenter identified respectively by the letters A, B, C and by D. We consider linear Lagrangian shape functions λ_i such that $\lambda_i(j) = \delta_{ij}$ for $i,j = A,B,C$. Then we consider the lowest order Raviart-Thomas functions that, for simplicity, we now identify with the face and not with the opposite node. Therefore we write $\boldsymbol{\phi}_{ij}$ for $i,j = A,B,C$ and $i \neq j$, where ij refers to a given face. So for example $\boldsymbol{\phi}_{AB}$ is the RT_0 function related to the face AB with opposite vertex C. We can write the RT_1 functions related to the face AB and to the node A as:

$$\boldsymbol{\psi}_{AB,A} = \boldsymbol{\phi}_{AB}(\alpha_{AB}\lambda_A + \gamma_{AB}\lambda_C)\,, \tag{5.29}$$

where $\alpha_{AB}, \gamma_{AB} \in \mathbb{R}$. The letters α and γ refer to the used Lagrangian shape functions, λ_A and λ_C, that correspond to the nodes A and C. We want to define

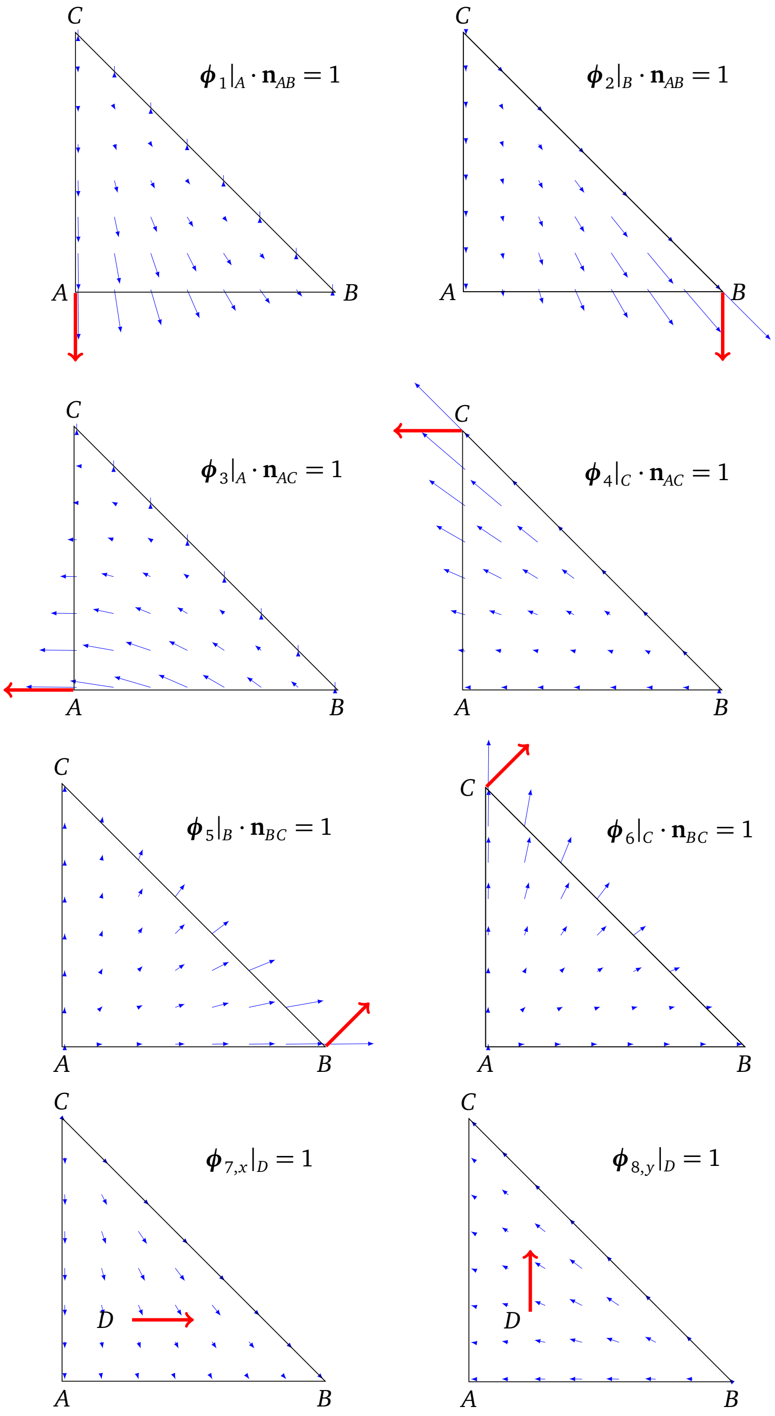


Figure 5.4. Representation on the reference triangle of RT_1 shape functions. The arrows are scaled by a factor of 0.2.

α_{AB}, γ_{AB} so that (5.28) is satisfied. This means that we must satisfy:

$$\begin{aligned}
&\boldsymbol{\psi}_{AB,A}|_A \cdot \mathbf{n}_{AB} = 1 && \boldsymbol{\psi}_{AB,A}|_B \cdot \mathbf{n}_{AB} = 0 && \text{(5.30a)}\\
&\boldsymbol{\psi}_{AB,A}|_A \cdot \mathbf{n}_{AC} = 0 && \boldsymbol{\psi}_{AB,A}|_C \cdot \mathbf{n}_{AC} = 0 && \text{(5.30b)}\\
&\boldsymbol{\psi}_{AB,A}|_B \cdot \mathbf{n}_{BC} = 0 && \boldsymbol{\psi}_{AB,A}|_C \cdot \mathbf{n}_{BC} = 0 && \text{(5.30c)}\\
&\boldsymbol{\psi}_{AB,A}|_D \cdot \mathbf{e}_1 \ \ = 0 && \boldsymbol{\psi}_{AB,A}|_D \cdot \mathbf{e}_2 \ \ = 0 && \text{(5.30d)}
\end{aligned}$$

By construction, we already know that:

$$\boldsymbol{\psi}_{AB,A} \cdot \mathbf{n}_{AC} = \boldsymbol{\psi}_{AB,A} \cdot \mathbf{n}_{BC} = 0\,, \tag{5.31}$$

because $\boldsymbol{\phi}_{AB} \cdot \mathbf{n}_{AC} = \boldsymbol{\phi}_{AB} \cdot \mathbf{n}_{BC} = 0$. And this is true for any α_{AB}, γ_{AB}. Thus (5.30b), (5.30c) are fulfilled. Furthermore $\lambda_A|_B = \lambda_C|_B = 0$, so:

$$\boldsymbol{\psi}_{AB,A}|_B \cdot \mathbf{n}_{AB} = 0\,, \tag{5.32}$$

and the second equation in (5.30a) is automatically satisfied. We now want to find α_{AB}, γ_{AB} such that (5.30d) is true. For this to happen, we just need to enforce:

$$(\alpha_{AB}\lambda_A + \gamma_{AB}\lambda_C)|_D = 0\,, \tag{5.33}$$

which, by using (5.20), boils down to:

$$\alpha_{AB} + \gamma_{AB} = 0\,. \tag{5.34}$$

Since this last equation is scalar in two unknowns, we still require one equation more to fully determine α_{AB} and γ_{AB}. We just need to consider the first equation in (5.30a):

$$\begin{aligned}
1 &= \boldsymbol{\psi}_{AB,A}|_A \cdot \mathbf{n}_{AB}\\
&= (\alpha_{AB}\lambda_A|_A + \gamma_{AB}\lambda_C|_A)\boldsymbol{\phi}_{AB}|_A \cdot \mathbf{n}_{AB}\\
&= \alpha_{AB}\boldsymbol{\phi}_{AB}|_A \cdot \mathbf{n}_{AB}\,,
\end{aligned} \tag{5.35}$$

where we have used $\lambda_C|_A = 0$ and $\lambda_A|_A = 1$. So the condition computed in (5.35) reduces to:

$$\alpha_{AB} = \frac{1}{\boldsymbol{\phi}_{AB}|_A \cdot \mathbf{n}_{AB}}\,, \tag{5.36}$$

while γ_{AB} can be directly computed from (5.34):

$$\gamma_{AB} = -\frac{1}{\boldsymbol{\phi}_{AB}|_A \cdot \mathbf{n}_{AB}}\,. \tag{5.37}$$

The same way of reasoning can be applied to $\boldsymbol{\psi}_{AB,B}$ and to all other shape functions related to the faces of the simplex, $\boldsymbol{\psi}_{AC,A}$, $\boldsymbol{\psi}_{AC,C}$, $\boldsymbol{\psi}_{BC,B}$, $\boldsymbol{\psi}_{BC,C}$. On the other hand, the shape functions related to internal momenta can be defined as:

$$\boldsymbol{\psi}_{D,k} = \gamma_k \boldsymbol{\phi}_{AB} \lambda_C + \beta_k \boldsymbol{\phi}_{AC} \lambda_B \qquad k = 1, 2\,, \tag{5.38}$$

that must fulfill:

$$\begin{aligned}
&\boldsymbol{\psi}_{D,k}|_A \cdot \mathbf{n}_{AB} = 0 && \boldsymbol{\psi}_{D,k}|_B \cdot \mathbf{n}_{AB} = 0 && \text{(5.39a)}\\
&\boldsymbol{\psi}_{D,k}|_A \cdot \mathbf{n}_{AC} = 0 && \boldsymbol{\psi}_{D,k}|_C \cdot \mathbf{n}_{AC} = 0 && \text{(5.39b)}\\
&\boldsymbol{\psi}_{D,k}|_B \cdot \mathbf{n}_{BC} = 0 && \boldsymbol{\psi}_{D,k}|_C \cdot \mathbf{n}_{BC} = 0 && \text{(5.39c)}\\
&\boldsymbol{\psi}_{D,k}|_D \cdot \mathbf{e}_1 \ \ = \delta_{1k} && \boldsymbol{\psi}_{D,k}|_D \cdot \mathbf{e}_2 \ \ = \delta_{2k}\,. && \text{(5.39d)}
\end{aligned}$$

By construction, $\boldsymbol{\psi}_{D,k} \cdot \mathbf{n} = 0$ on the boundary, for $k = 1, 2$. This is true because both addenda in (5.38) satisfy the same condition. Indeed, for example, $\boldsymbol{\phi}_{AB} \cdot \mathbf{n}_{AC} = \boldsymbol{\phi}_{AB} \cdot \mathbf{n}_{BC} = 0$; on the face AB, $\boldsymbol{\phi}_{AB} \cdot \mathbf{n}_{AB} \neq 0$, but $\lambda_C|_{AB} = 0$. At the same way, $\boldsymbol{\phi}_{AC} \cdot \mathbf{n}_{AB} = \boldsymbol{\phi}_{AC} \cdot \mathbf{n}_{BC} = 0$; on the face AC, $\boldsymbol{\phi}_{AC} \cdot \mathbf{n}_{AC} \neq 0$, but $\lambda_B|_{AC} = 0$. Therefore the relations (5.39a), (5.39b), (5.39c) are satisfied. We now need to fulfill (5.39d). This implies solving, for each shape function $\boldsymbol{\psi}_{D,k}$, a 2 × 2 linear system that has, as unknown, the vector of coefficients (γ_k, β_k):

$$\begin{bmatrix} \big([\boldsymbol{\phi}_{AB}]_1 \lambda_C\big)|_D & \big([\boldsymbol{\phi}_{AC}]_1 \lambda_B\big)|_D \\ \big([\boldsymbol{\phi}_{AB}]_2 \lambda_C\big)|_D & \big([\boldsymbol{\phi}_{AC}]_2 \lambda_B\big)|_D \end{bmatrix} \begin{bmatrix} \gamma_k \\ \beta_k \end{bmatrix} = \begin{bmatrix} b_{1,k} \\ b_{2,k} \end{bmatrix} \quad b_{j,k} = \delta_{kj} \quad j = 1, 2\,.$$

We can notice that the matrix to invert is the same for $k = 1, 2$, so it can be computed only once, while the right-hand side can change.

A generalization to a d-simplex is straightforward. Let us identify the vertices of the simplex K of coordinates $\{\mathbf{p}_i\}_{i=1}^{d+1}$ with the set $\{p_i\}_{i=1}^{d+1}$ and the barycenter $\mathbf{b}$ with the letter b. The face f_i of normal $\mathbf{n}_i$ for $i = 1, ..., d+1$ is the one which does not contain p_i. Then we can write the shape functions related to the face f_i and to the node p_j, belonging to f_i, as:

$$\boldsymbol{\psi}_{f_i,j} = \boldsymbol{\phi}_{f_i} \left(\alpha_{f_i,j} \lambda_j + \alpha_{f_i,i} \lambda_i \right) \quad i, j = 1, \ldots, d+1, \quad i \neq j \tag{5.40}$$

Here it can be replicated the same reasoning used for the 2D case. So we just need to enforce:

$$\begin{cases} \alpha_{f_i,j} + \alpha_{f_i,i} = 0 \\ \alpha_{f_i,j} = \dfrac{1}{\boldsymbol{\psi}_{f_i,j}|_j \cdot \mathbf{n}_{f_i}} \end{cases} , \tag{5.41}$$

so that all the conditions in (5.28) are zero except for $\boldsymbol{\psi}_{f_i,j}|_j \cdot \mathbf{n}_{f_i} = 1$. The internal momentum shape functions look like:

$$\boldsymbol{\phi}_{D,k} = \sum_{i=1}^{d} \alpha_{f_i,k} \boldsymbol{\phi}_{f_i} \lambda_i \qquad k = 1, \ldots, d \tag{5.42}$$

and automatically satisfy $\boldsymbol{\phi}_{D,k} \cdot \mathbf{n} = 0$ on the boundary for any $\alpha_{f_i,k} \in \mathbb{R}$ and for $k = 1, \ldots, d$. In order to close the formula and determine the values $\alpha_{f_i,k}$, the following $d \times d$ system has to be solved:

$$\begin{bmatrix} \big([\boldsymbol{\phi}_{f_1}]_1 \lambda_1\big)|_b & \cdots & \big([\boldsymbol{\phi}_{f_d}]_1 \lambda_d\big)|_b \\ \vdots & \cdots & \vdots \\ \big([\boldsymbol{\phi}_{f_1}]_d \lambda_1\big)|_b & \cdots & \big([\boldsymbol{\phi}_{f_d}]_d \lambda_d\big)|_b \end{bmatrix} \begin{bmatrix} \alpha_{f_1,k} \\ \vdots \\ \alpha_{f_d,k} \end{bmatrix} = \begin{bmatrix} b_{1,k} \\ \vdots \\ b_{d,k} \end{bmatrix} \qquad b_{i,k} = \delta_{ki},\ i = 1, \ldots, d\,. \tag{5.43}$$

The matrix is again indipendent of k. Therefore it can be inverted only once and its inverse can be applied to the right-hand side $(b_{1,k}, \ldots, b_{d,k})^T$, which depends on $k = 1, \ldots, d$.

In conclusion, it is clear that, for satysfying (5.3), the RT_1 shape functions must be defined directly on K. However, their definition is based on linear Lagrangian and RT_0 shape functions, which can be precomputed on $\hat{K}$ and then mapped to K. Therefore, on K, it is just sufficient to solve for (5.41), (5.43).

5.6 The local Nédélec finite element space of the first kind

We now want to give a brief introduction to $H_{\text{curl}}(\Omega)$-conforming FE space. This can be useful for Hiptmair's smoother discussed in section 6.12. In particular, for $d = 2$, due to (2.14), the space $H_{\text{curl}}(\Omega)$ could be identified with $H^1(\Omega)$. For $d = 3$, we need to introduce the Nédélec finite element space for a tetrahedron K. For more details, see Brezzi and Fortin [2012]. It is defined in the following way:

$$ND_q(K) := \begin{cases} K \text{ is a simplex} \\ \mathcal{V} = \big[P_q(K)\big]^d + \mathbf{x} \wedge \big[Q_q(K)\big]^d \\ \mathcal{L} = \begin{cases} \int_{e_i} \boldsymbol{\chi} \cdot \mathbf{t}\, p\, d\mathbf{l} \quad p \in P_q(e_i), \text{ for each edge } e_i \text{ of K} \\ \int_{f_i} \boldsymbol{\chi} \cdot \mathbf{p}\, d\mathbf{s} \quad \mathbf{p} \in \big[P_q(f_i)\big]^d, \text{ for each face } f_i \text{ of K}, \quad q \geq 1 \\ \int_K \boldsymbol{\chi} \cdot \mathbf{p}\, d\,\mathbf{x} \quad \mathbf{p} \in \big[P_{q-2}(K)\big]^d, q \geq 2 \end{cases} \end{cases} , \tag{5.44}$$

with dimension:

$$n_{\mathcal{V}}(q) = \begin{cases} (q+1)(q+3) & T = \text{triangle} \\ \frac{1}{2}(q+1)(q+2)(q+4) & T = \text{tetrahedron}\,. \end{cases} \tag{5.45}$$

A 2D representation for the lowest-order case, ND_0, is represented in Figure 5.5.

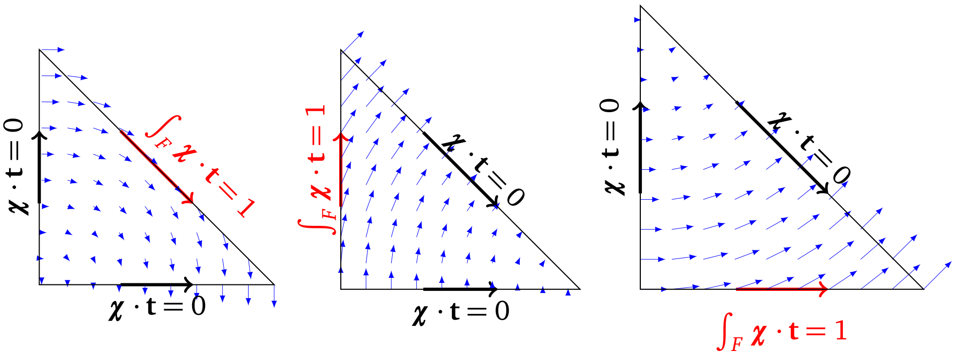

$\boldsymbol{\chi}\cdot\mathbf{t}=0$
$\int_F \boldsymbol{\chi}\cdot\mathbf{t}=1$
$\boldsymbol{\chi}\cdot\mathbf{t}=0$
$\int_F \boldsymbol{\chi}\cdot\mathbf{t}=1$
$\boldsymbol{\chi}\cdot\mathbf{t}=0$
$\boldsymbol{\chi}\cdot\mathbf{t}=0$
$\boldsymbol{\chi}\cdot\mathbf{t}=0$
$\boldsymbol{\chi}\cdot\mathbf{t}=0$
$\int_F \boldsymbol{\chi}\cdot\mathbf{t}=1$

Figure 5.5. Representation on the reference triangle of ND_1 shape functions. The arrows have been scaled by a factor of 0.1.

5.7 Lagrangian and Raviart-Thomas global finite element spaces

In (5.11), (5.12) we have introduced a general definition for spaces that are dof-continuous or discontinuous. Given a tessellation $\mathcal{T}_h$, we have introduced a local finite element space $(K, \mathcal{V}, \mathcal{L})$ for a general element $K \in \mathcal{T}_h$. A global finite element space over $\mathcal{T}_h$ consists of functions that, if restricted onto $K \in \mathcal{T}_h$, belong to $\mathcal{V}(K)$ and satisfy the required continuity conditions on the common interfaces between adjacent cells. Furthermore, to simplify the implementation of the shape functions, a reference element $\hat{K}$ and reference quantities have been examined as well. In this section, we will focus on the Lagrangian and the Raviart-Thomas spaces.

Here the definitions of the continuous and discontinuous Lagrangian FE spaces of order $q \in \mathbb{N}$:

$$P_q(\mathcal{T}_h) = \{v_h \in C^0(\Omega): \quad v_h|K \in P_q(K)\ \forall K \in \mathcal{T}_h\} \tag{5.46}$$

$$DP_q(\mathcal{T}_h) = \{v_h \in L^2(\Omega): \quad v_h|K \in P_q(K)\ \forall K \in \mathcal{T}_h\} \tag{5.47}$$

and the mapping can be implemented as in (5.18). The continuous global Raviart-Thomas space of order $q \in \mathbb{N}$ defined by:

$$RT_q(\mathcal{T}_h) = \{\boldsymbol{\sigma}_h \in H_{\text{div}}(\Omega): \quad \boldsymbol{\sigma}_h|_K \in RT_q(K)\ \forall K \in \mathcal{T}_h\}\,, \tag{5.48}$$

has a mapping that is trickier. We will now focus on the case of RT_0 functions. In fact higher-order RT shape functions can be written as combinations of Lagrangian and RT_0 shape functions and the following argument will apply as well.

The contravariant Piola's transformation in (5.25) preserves normal traces of a vector field mapped from the reference element $\hat{K}$ to K. If on $\hat{K}$, all RT_0 shape functions are defined in order to have a positive external flux, this property will be transferred onto K. In this scenario, using only Piola's transformation does not guarantee the normal flux continuity for the finite element functions defined on $\mathcal{T}_h$. Indeed a degree of freedom related to a face f is shared by two adjacent elements K_1 and K_2. To ensure normal flux continuity, the evaluation of a degree of freedom must be independent of the fact that we consider the face f as belonging to K_1 or K_2. However, as it can be seen from Figure 5.6, the outward normals $\mathbf{n}_1$ and $\mathbf{n}_2$, defined on K_1 and K_2, are opposite. To have a unique definition of the degree of freedom, a unique normal has to be chosen. And it will be outward for one element and inward for the other one. As a rule, we can choose a normal that points from the element with the smaller label to the one with the larger one. On faces that discretize $\partial\Omega$, for simplicity, the normal is chosen to be outward. See also Rognes, Marie E. and Kirby, Robert C. and Logg, Anders [2009].

The mapping via Piola's transformation (5.25) can still be used, but it has to be generalized:

$$\boldsymbol{\phi}_f = \alpha_f \frac{\mathbf{J}}{\det(\mathbf{J})} \hat{\boldsymbol{\phi}}_{\hat{f}} \qquad \alpha_f = \begin{cases} +1 & \mathbf{n}_f \text{ is outward} \\ -1 & \mathbf{n}_f \text{ is inward} \end{cases} \tag{5.49}$$

So we can precompute the shape function $\hat{\boldsymbol{\phi}}_{\hat{f}}$ on the reference element $\hat{K}$ related to the face $\hat{f}$ in some significant points, but then, in addition to the standard map, also a coefficient for the sign detection of the global normal has to be computed. Since RT_q shape functions, for $q > 0$, are built with RT_0 functions, this argument

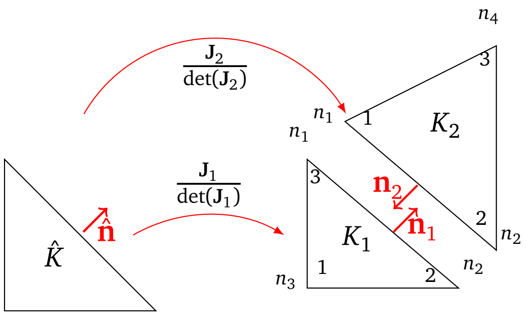


Figure 5.6. To ensure normal flux continuity of Raviart-Thomas functions, we must define global normals: each face f has a normal $\boldsymbol{n}$ which points from the element with smaller label to the element with the larger one. For $RT_1(\mathcal{T}_h)$, we order the degrees of freedom on each face based on the global numbering scheme of the nodes $(n_1 < n_2 < n_3 < n_4)$. Indeed the local ordering can be different for adjacent elements.

is true for all Raviart-Thomas functions.

The introduction of (5.49) is necessary and sufficient for lowest-order Raviart-Thomas spaces. Nevertheless, for higher-order Raviart-Thomas spaces, we must also introduce a proper ordering of the nodes. Indeed, for $RT_0(\mathcal{T}_h)$, there is one single degree of freedom per face and using (5.49) is enough. But, for $RT_1(\mathcal{T}_h)$, each face corresponds to d degrees of freedom, one for each node of the face. These dofs have to be properly glued between adjacent elements. Considering the local numbering of the elements could produce possible mismatches. One way to overcome this difficulty is to order the degrees of freedom of each face based on the global numbering scheme of the nodes. For example, in Figure 5.6, the first dof would be related to n_1, the second one to n_2, assuming $n_1 < n_2$.

5.8 The discrete compressible displacement-based primal formulation

The discrete version of the primal weak form for the Signorini problem (4.3) is the following. We consider the discrete quadratic functional $\mathcal{J}_{h,p} : \mathbf{U}_h \to \mathbb{R}$ to be minimized over the closed convex set $\mathcal{K}_{h,p}$:

$$\mathcal{J}_{h,p}(\mathbf{u}_h) = \frac{1}{2} a_{h,p}(\mathbf{u}_h, \mathbf{u}_h) - f_{h,p}(\mathbf{u}_h)\,, \tag{5.50a}$$

$$\mathcal{K}_{h,p} = \{\mathbf{u} \in \mathbf{U}_{\mathbf{g}_{h,D}} \,:\, u_{h,n} \leq g_h \text{ on } \Gamma_{h,C}\}\,. \tag{5.50b}$$

Then $\hat{\mathbf{u}}_h$ is the minimizer $\hat{\mathbf{u}}_h = \arg\min_{\mathbf{u}_h \in \mathcal{K}_{p,h}} \mathcal{J}_{p,h}(\mathbf{u}_h)$. As it is clear, (5.50) is a quadratic optimization problem subject to box constraints. Therefore we need only to choose $\mathbf{U}_h$ to be $\mathbf{U}$-conforming. For this to happen, it is sufficient to choose:

$$\mathbf{U}_h = \mathbf{P}_q(\mathcal{T}_h)\,. \tag{5.51}$$

In particular, we choose the linear case: $q = 1$.

5.9 The discrete stress-based dual formulation

For the discrete version of the dual weak form of the Signorini problem (4.4), we define the discrete quadratic functional $\mathcal{J}_{h,d} : \mathbf{\Sigma}_h \to \mathbb{R}$ that has to be minimized over the closed convex set $\mathcal{K}_{h,d} \subset \mathbf{\Sigma}_{h,\mathbf{g}_{h,N}}$:

$$\mathcal{J}_{h,d}(\boldsymbol{\sigma}) := \frac{1}{2} a_{h,d}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h) - f_{h,d,1}(\boldsymbol{\sigma}_h) - \langle g_h, \sigma_{h,n} \rangle_{\Gamma_C}\,, \tag{5.52a}$$

$$\begin{aligned} \mathcal{K}_{h,d} := \{\boldsymbol{\sigma} \in \mathbf{\Sigma}_{h,t} : \quad & b_{h,d}(\boldsymbol{\sigma}_h, \mathbf{v}_h) &= f_{h,d,2}(\mathbf{v}_h) \qquad & \forall \mathbf{v}_h \in \mathbf{V}_h, \\ & c_{h,d}(\boldsymbol{\sigma}_h, \boldsymbol{\gamma}_h) &= 0 \qquad & \forall \boldsymbol{\gamma}_h \in \mathbf{\Theta}_h, \\ & \sigma_{h,n} &\leq 0 \qquad & \text{on } \Gamma_C\}\,. \end{aligned} \tag{5.52b}$$

The solution $\hat{\boldsymbol{\sigma}}_h$ is the minimizer $\hat{\boldsymbol{\sigma}}_h = \arg\min_{\boldsymbol{\sigma}_h \in \mathcal{K}_d} \mathcal{J}_{h,d}(\boldsymbol{\sigma}_h)$. The problem (5.52) is a quadratic optimization problem subject to both box constraints and global equality constraints. Thus, conformity of the spaces is not sufficient. The LBB condition (3.27) has to be fulfilled also in the discrete setting: exists $\beta_h > 0$ such that

$$\inf_{\substack{\mathbf{u}_h \in \mathbf{V}_h \\ \boldsymbol{\theta}_h \in \mathbf{\Theta}_h}} \sup_{\boldsymbol{\sigma}_h \in \mathbf{\Sigma}_h} \frac{b_d(\boldsymbol{\sigma}_h, \mathbf{v}_h) + c_d(\boldsymbol{\sigma}_h, \boldsymbol{\theta}_h)}{\|\boldsymbol{\sigma}_h\|_{\mathbf{\Sigma}_h} (\|\mathbf{u}_h\|_{\mathbf{U}_h} + \|\boldsymbol{\theta}_h\|_{\mathbf{\Theta}_h})} \geq \beta_h > 0\,, \tag{5.53}$$

where the continuous bilinear forms, b_d and c_d, and their discrete versions, $b_{h,d}$ and $c_{h,d}$, coincide. The spaces $\mathbf{\Sigma}_h$, $\mathbf{V}_h$ and $\mathbf{\Theta}_h$ cannot be chosen conforming but arbitrarly. To fulfill (5.53) a choice of spaces examined in Brezzi and Fortin [2012] is the following:

$$\begin{aligned} \mathbf{\Sigma}_h &= \mathbf{RT}_1(\mathcal{T}_h)\,, \\ \mathbf{V}_h &= \mathbf{DP}_1(\mathcal{T}_h)\,, \\ \mathbf{\Theta}_h &= [P_1(\mathcal{T}_h)]^{d\times d} \cap \mathbf{\Theta}\,. \end{aligned} \tag{5.54}$$

Remark 5.9.1. *We observe that the space $\mathbf{\Theta}$ consists of L^2 skew-symmetric matrix-value functions. Therefore, in the discrete setting, for $d = 2$ only one scalar function is required, while for $d = 3$ three scalar functions are needed. Indeed we can write:*

$$\mathbf{\Theta}_h = \left\{ \begin{bmatrix} 0 & -a \\ a & 0 \end{bmatrix}, \quad a \in P_1(\mathcal{T}_h) \right\} \qquad d = 2\,, \tag{5.55}$$

$$\mathbf{\Theta}_h = \left\{ \begin{bmatrix} 0 & -a & -b \\ a & 0 & -c \\ b & c & 0 \end{bmatrix}, \quad a, b, c \in P_1(\mathcal{T}_h) \right\} \qquad d = 3\,. \tag{5.56}$$

The formulation can be rewritten in terms of a for $d = 2$ or in terms of a, b, c for $d = 3$. Then, instead of the space $\mathbf{\Theta}_h = [P_1(\mathcal{T}_h)]^{d\times d} \cap \mathbf{\Theta}$, we use $\Theta_h = P_1(\mathcal{T}_h)$ or $\mathbf{\Theta}_h = [P_1(\mathcal{T}_h)]^3$.

5.10 Discrete FOSLS formulation

The discrete version of the FOSLS Signorini problem (4.5), is: find $\mathbf{u}_h \in \mathbf{U}_{h,\mathbf{g}_{h,D}}$, $\boldsymbol{\sigma}_h \in \mathbf{\Sigma}_{h,\mathbf{g}_{h,N}}$, that minimize the discrete FOSLS functional $\mathcal{J}_{h,f}$ over the closed convex set $\mathcal{K}_{h,f} \subset \mathbf{U}_{h,\mathbf{g}_{h,D}} \times \mathbf{\Sigma}_{h,\mathbf{g}_{h,N}}$:

$$\mathcal{J}_{h,f}(\mathbf{u}_h, \boldsymbol{\sigma}_h) = \gamma ||\mathcal{A}\boldsymbol{\sigma}_h - \boldsymbol{\varepsilon}(\mathbf{u}_h)||^2_{L^2(\Omega_h)} + \delta ||\mathrm{div}\boldsymbol{\sigma}_h + \mathbf{f}_h||^2_{L^2(\Omega_h)} + \langle \sigma_{h,n}, u_{h,n} - g_h \rangle_{\Gamma_{h,C}}\,, \tag{5.57a}$$

$$\mathcal{K}_{h,f} = \{(\mathbf{u}_h, \boldsymbol{\sigma}_h) \in \mathbf{U}_{h,\mathbf{g}_{h,D}} \times \mathbf{\Sigma}_{h,\mathbf{g}_{h,N}} : u_{h,n} \le g_h,\ \sigma_{h,n} \le 0, \\ (\boldsymbol{\sigma}_h \mathbf{n}_h)_t = \mathbf{0} \text{ on } \Gamma_{h,C}\}\,, \tag{5.57b}$$

The solution $(\hat{\mathbf{u}}_h, \hat{\boldsymbol{\sigma}}_h)$ is then the minimizer:

$$(\hat{\mathbf{u}}_h, \hat{\boldsymbol{\sigma}}_h) = \underset{(\mathbf{u}_h, \boldsymbol{\sigma}_h) \in \mathcal{K}_{h,f}}{\arg\min}\ \mathcal{J}_{h,f}(\mathbf{u}_h, \boldsymbol{\sigma}_h)\,.$$

The problem (5.57) is a quadratic optimization problem subject to box constraints. Thus no LBB condition has to be satisfied and any pair of conforming spaces can be used. This implies also the lowest order FE spaces:

$$\begin{aligned} \mathbf{U}_h &= \mathbf{P}_1(\mathcal{T}_h)\,, \\ \mathbf{\Sigma}_h &= \mathbf{RT}_0(\mathcal{T}_h)\,. \end{aligned} \tag{5.58}$$

In the sequel, we will consider the functions as appropriately discretized and we will omit the subscript h to simplify the notation. We will highlight the subscript h whenever it is really important, e.g. for the spaces, the sets, and the variables of the problem.

Chapter 6

Monotone multigrid methods

In section 6.2 of this chapter, we take advantage of a compact framework in order to describe in a unique way the previous discrete minimization problems (5.50), (5.52), (5.57). This compact framework reduces to a quadratic programming problem (QPP), with the minimization of a quadratic functional subject to box-constraints and, eventually, global equality constraints. Since the functional to minimize is quadratic, the main difficulty lies in the constraints set which makes the problem to be solved non-linear.

6.1 Introduction

There are various strategies to tackle a non-linearity. The most famous one is the Newton method, which approximates the roots of a functional by solving a sequence of linearized problems. Since this method acts directly on a functional, we need somehow to insert the constraints of the formulations (5.50), (5.52), (5.57) into their functionals. This procedure makes the new functionals non-smooth. Then the semi-smooth Newton method is necessary. See for example Hintermüller, Michael and Ito, Kazufumi and Kunisch, Karl [2002], Hintermüller, Michael [2010], Kunisch, Karl [2008]. In particular, in Hintermüller, Michael and Ito, Kazufumi and Kunisch, Karl [2002] the relation between the semi-smooth Newton method and the primal-dual active set method is discussed. Indeed, another method for tackling the constraints is the active set method. This comes with its variants, i.e., primal, dual, primal-dual.

As explained in Nocedal, Jorge and Wright, Stephen [2006], the primal active set method solves a sequence of linearized problems, as the Newton method. The active set is the set of all dofs which we assume to be active, either satisfying an upper or a lower bound. At each iteration, in addition to the global equality constraints, we enforce as equality constraints all the inequality constraints belonging to the active set. Then, at the given iteration, the constraint set consists only of equality constraints and the problem to be solved is a saddle point system. We terminate the iterative process if the solution of the saddle point coincides with the one of the original constrained problem. Otherwise, we must update the definition of the active set. We will describe this procedure in section 6.3.

Another way to remove the inequality constraints is to use the penalty method. The constraints are added into the functional as a penalty term and the corresponding problem to be solved is just a linear one. Of course, a tradeoff between the accuracy of the solution and the efficiency of the solution method arises. See Carstensen, Carsten and Scherf, Oliver and Wriggers, Peter [1999], Eck, Christof and Steinbach, Olaf and Wendland, Wolfgang L [1999].

The Newton method and the active set methods tackle the constraints by means of a sequence of global linearized problems. However, by increasing the dimension of the problem, the number of constraints and, potentially, of the linear problems to be solved increases as well. On the other hand, penalty methods are not accurate enough for our purposes. We would like to use a method that captures the constraints exactly and has optimal complexity, meaning that the number of iterations that it requires is independent of the dimension of the problem. The monotone multigrid method has both advantages. See Kornhuber, Ralf [1994], Kornhuber, Ralf [1996], Kornhuber, Ralf and Krause, Rolf [2001], Kornhuber, Ralf and Krause, Rolf and Sander, O. and Deuflhard, P. and Ertel, S. [2008], Krause [2009], Krause, Rolf and Rigazzi, Alessandro and Steiner, Johannes [2016], Badea [2002], Badea [2014]. The idea is to generalize the subspace correction method by Xu, Jinchao [1996], introduced in section 6.4, to constrained problems, by sequentially minimizing the energy functional on fine and coarse subspaces. We will discuss monotone multigrid in section 6.6. Nevertheless, as it can be shown in the experiments, the behavior of monotone multigrid for the primal formulation deteriorates for increasing of the Lamé parameter λ. This is why we want to investigate the dual or the FOSLS formulations, which remain bounded even for $\lambda \to \infty$. However, since the stress belongs to $\mathbf{H}_{\text{div}}(\Omega)$ and not to the standard Sobolev space $\mathbf{H}^1(\Omega)$, a special care is required. Geometric and algebraic multigrids have been already examined for H_{div} problems in the linear case. See Hiptmair, Ralf and Xu, Jinchao [2007], Kolev and Vassilevski [2012], Xu, Jinchao and Chen, Long and Nochetto, Ricardo H. [2009]. But, to the author's knowledge, multigrid methods applied to contact problems for a stress-based formulation are still to be discovered. Since no theory is given, we will exploit what has already been used for the multigrid methods applied to the primal Signorini and for the linear H_{div} problems. Thus we will take advantage of the monotone multigrid method and H_{div} smoothers.

6.2 A compact formulation

The previous problems (4.3), (4.4), (4.5) can be rewritten in a compact way. Let $\mathbf{X}$ be a Hilbert space, $\mathbf{K} \subset \mathbf{X}$ a non empty, closed and convex set and $\mathcal{J} : \mathbf{X} \to \mathbb{R}$ a quadratic functional, non-negative on $\mathbf{K}$, of the form:

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2} a(\mathbf{x},\mathbf{x}) - f(\mathbf{x}) \,, \tag{6.1}$$

where $a : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ is a continuous bilinear form and $f : \mathbf{X} \to \mathbb{R}$ is a continuous linear form. Then the problem is finding the minimum of the functional $\mathcal{J}$ over the closed convex set $\mathbf{K} \subset \mathbf{X}$:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbf{K}} \mathcal{J}(\mathbf{x}) \,. \tag{6.2}$$

For the different formulations, we can set the space $\mathbf{X}$, the convex set $\mathbf{K}$ and the functional $\mathcal{J}$ as in Table 6.1. We have also inserted the space $\mathbf{W}$ which can be used in the dual formulation for enforcing global equality constraints by means of Lagrange multipliers.

	X	W	K	$\mathcal{J}$
Primal	$\mathbf{U}$	–	$\mathbf{K}_p$	$\mathcal{J}_p$
Dual	$\boldsymbol{\Sigma}$	$\mathbf{V} \times \boldsymbol{\Theta}$	$\mathbf{K}_d$	$\mathcal{J}_d$
FOSLS	$\mathbf{U} \times \boldsymbol{\Sigma}$	–	$\mathbf{K}_f$	$\mathcal{J}_f$

Table 6.1. Continuous compact framework.

The same argument can be repeated for a common description of the discrete problems (5.50), (5.52), (5.57). We define a discrete space $\mathbf{X}_h \subset \mathbf{X}$, a non empty closed convex set K_h and a discrete quadratic functional $\mathcal{J}_h : \mathbf{X}_h \to \mathbb{R}$:

$$\mathcal{J}_h(\mathbf{x}_h) = \frac{1}{2} a_h(\mathbf{x}_h,\mathbf{x}_h) - f_h(\mathbf{x}_h) \,, \tag{6.3}$$

where a_h and b_h are discretized versions of a and b. The solution of the problem is:

$$\hat{\mathbf{x}}_h = \arg\min_{\mathbf{x}_h\in\mathbf{K}_h} \mathcal{J}_h(\mathbf{x}_h) \,. \tag{6.4}$$

	$\mathbf{X}_h$	$\mathbf{W}_h$	$\mathbf{K}_h$	$\mathcal{J}_h$
Primal	$\mathbf{P}_1$	$-$	$\mathbf{K}_{h,p}$	$\mathcal{J}_{h,p}$
Dual	$\mathbf{RT}_1$	$\mathbf{DP}_1 \times \mathbf{P}_1^{\mathrm{asym}}$	$\mathbf{K}_{h,d}$	$\mathcal{J}_{h,d}$
FOSLS	$\mathbf{P}_1 \times \mathbf{RT}_0$	$-$	$\mathbf{K}_{h,f}$	$\mathcal{J}_{h,f}$

Table 6.2. Discrete compact framework.

The discrete version of the spaces and sets of Table 6.1 is Table 6.2.

Once the spaces, their bases and dofs are chosen, it is also possibile to reformulate (6.4) in a vector-matrix form. Let n be the dimension of $\mathbf{X}_h$, i.e., $n = \dim(\mathbf{X}_h)$. We collect the values of the coefficients of the unknown into the vector $\mathbf{y}_h \in \mathbf{Y}_h = \mathbb{R}^n$. Then, due to the Riesz theorem, we can associate to $a_h : \mathbf{X}_h \times \mathbf{X}_h \to \mathbb{R}$ and $f_h : \mathbf{X}_h \to \mathbb{R}$ respectively the matrix $\mathbf{A}_h \in \mathbb{R}^{n,n}$ and the vector $\mathbf{f}_h \in \mathbb{R}^n$. Similarly we can introduce the space $\mathbf{Z}_h$ for the coefficients related to the Lagrange multipliers of the dual formulation, $\mathbf{u}_h$ and $\boldsymbol{\theta}_h$, of dimension $m = \dim(\mathbf{Z}_h)$. The corresponding coefficients of the Lagrange multipliers are collected into the vector $\mathbf{z}_h$. Then, with a similar argument, the global equality constraints can be written as $\mathbf{B}_h\mathbf{y}_h = \mathbf{h}_h$, where $\mathbf{B}_h \in \mathbb{R}^{m,n}$ and $\mathbf{h}_h \in \mathbb{R}^m$. Furthermore also the box-constraints can be expressed in terms of a lower bound $\mathbf{l}_h \in \mathbb{R}^n$ and an upper bound $\mathbf{u}_h \in \mathbb{R}^n$. We want to stress out that assuming $(\mathbf{l}_h)_i = (\mathbf{u}_h)_i$, we force $(\mathbf{y}_h)_i = (\mathbf{l}_h)_i = (\mathbf{u}_h)_i$ and thus we can enforce on the i-th dof the corresponding boundary condition. On the other hand if the i-th dof is free, it suffices to put $(\mathbf{l}_h)_i = \{-\infty\}$ and $(\mathbf{u}_h)_i = \{+\infty\}$. The functional and the convex set then become:

$$\mathcal{J}_h(\mathbf{y}_h) = \frac{1}{2}\mathbf{y}_h^T \mathbf{A}_h \mathbf{y}_h - \mathbf{y}_h^T \mathbf{f}_h \ , \tag{6.5a}$$

$$\mathbf{K}_h = \{\mathbf{y}_h \in \mathbf{Y}_h : \quad \mathbf{B}_h\mathbf{y}_h = \mathbf{h}_h, \quad \mathbf{l}_h \le \mathbf{y}_h \le \mathbf{u}_h\}\,, \tag{6.5b}$$

where the inequality $\mathbf{l}_h \le \mathbf{y}_h \le \mathbf{u}_h$ is meant to be component-wise. The residual corresponding to the whole KKT (Karush, Kuhn, Tucker) system related to (6.5) without considering the inequality constraints can be defined as well:

$$\mathbf{r} := \begin{bmatrix} \mathbf{f}_h \\ \mathbf{h}_h \end{bmatrix} - \begin{bmatrix} \mathbf{A}_h & \mathbf{B}_h^T \\ \mathbf{B}_h & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ \mathbf{z}_h \end{bmatrix}. \tag{6.6}$$

In order to take into consideration the inequality constraints, all the active dofs, i.e., all the dofs belonging to the active set are removed from the computation of the residual.

In section 6.3 we introduce the primal active set method for tackling globally the inequality constraints in (6.5b). In the other sections of the chapter, we will see how the constraints can be captured also locally by means of a subspace correction approach.

Remark 6.2.1. *In general the conditions (4.1a) and (4.1b) are not box-constraints. Thus we cannot really recover the set (6.5b). However, by changing the local coordinate system from Cartesian to an orthogonal coordinate system with first component identified by the direction of the normal* **n***, (4.1a) and (4.1b) can be transformed into box-constraints. To this aim it is possible to define on* Γ_C *the Householder transformation. See Krause, Rolf [2001] and section 6.9.5 for further details.*

6.3 Primal active-set

In order to solve the QPP with a primal active set strategy, a sequence of linearized problems, for the iterations $k = 0, 1, \ldots$, has to be solved. Given the *ordered* sets of active dofs W_l^k and W_u^k, related to the lower and the upper bounds, the corresponding saddle point system can be solved. Then, if the solution of the k-th linearized problem, i.e., $\mathbf{y}_h^k$, is also the solution of the original constrained problem, we stop; otherwise, the active sets W_l^k and W_u^k are updated and the process is repeated. Let us examine in more detail this procedure.

We now want to express the problem (6.5) in terms of the correction. In this way, assuming the initial guess is in the feasible set, we can guarantee the lower and the upper bounds to be always respectively negative and positive. Furthermore, this choice will fit the setting of the subspace correction method of section 6.4. The current iterate $\mathbf{y}_h^k$, such that $\mathbf{l}_h \leq \mathbf{y}_h^k \leq \mathbf{u}_h$, is updated by means of a correction $\mathbf{c}_h^k$, in order to get the value $\mathbf{y}_h^{k+1} = \mathbf{y}_h^k + \mathbf{c}_h^k$. The problem (6.5) can be rewritten with respect to the correction $\mathbf{c}_h^k$:

$$\mathcal{J}_h(\mathbf{c}_h^k) = \frac{1}{2}\mathbf{c}_h^{k^T}\mathbf{A}_h\mathbf{c}_h^k - \mathbf{c}_h^{k^T}(\mathbf{f}_h - \mathbf{A}_h\mathbf{y}_h^k) \ , \tag{6.7a}$$

$$\mathbf{K}_h = \{\mathbf{c}_h^k \in \mathbf{Y}_h : \quad \mathbf{B}_h\mathbf{c}_h^k = \mathbf{h}_h - \mathbf{B}_h\mathbf{y}_h^k, \quad \mathbf{l}_h - \mathbf{y}_h^k \leq \mathbf{c}_h^k \leq \mathbf{u}_h - \mathbf{y}_h^k\} \,, \tag{6.7b}$$

and consequently linearized. At the iterate k, the active sets W_l^k and W_u^k are such that::

$$i \in W_l^k \quad \Leftrightarrow \quad (\mathbf{y}_h^k)_i = (\mathbf{l}_h^k)_i \quad \Leftrightarrow \quad (\mathbf{c}_h^k)_i = 0 \,, \tag{6.8a}$$

$$i \in W_u^k \quad \Leftrightarrow \quad (\mathbf{y}_h^k)_i = (\mathbf{u}_h^k)_i \quad \Leftrightarrow \quad (\mathbf{c}_h^k)_i = 0 \,. \tag{6.8b}$$

We also set $n_l^k = \dim(W_l^k)$ and $n_u^k = \dim(W_u^k)$. Since W_l^k and W_u^k are ordered sets, we can also introduce the mappings $w_l^k : \{1, \ldots, n_l^k\} \rightarrow \{1, \ldots, n\}$ and

$w_u^k : \{1,\dots,n_u^k\} \to \{1,\dots,n\}$ which associate to the i-th active constraint the corresponding dof in the ordered sets W_l^k and W_u^k. Thanks to this definition, we define $\mathbf{C}_h^k \in \mathbb{R}^{n_l^k,n}$, $\mathbf{D}_h^k \in \mathbb{R}^{n_u^k,n}$ in the following way:

$$(\mathbf{C}_h^k)_{ij} = \delta_{w_l^k(i),j} \qquad i = 1,\dots,n_l^k,\ j = 1,\dots,n\,, \tag{6.9a}$$

$$(\mathbf{D}_h^k)_{ij} = \delta_{w_u^k(i),j} \qquad i = 1,\dots,n_u^k,\ j = 1,\dots,n\,. \tag{6.9b}$$

Finally, we can define by $\mathbf{l}_h^k$ and $\mathbf{u}_h^k$ the vectors of components $\mathbf{l}_h$ and $\mathbf{u}_h$ corresponding to the ordered dofs in W_l^k and W_u^k:

$$(\mathbf{l}_h^k)_i = (\mathbf{l}_h)_{w_l^k(i)} - (\mathbf{y}_h^k)_{w_l^k(i)} \qquad i = 1,\dots,n_l^k\,, \tag{6.10a}$$

$$(\mathbf{u}_h^k)_i = (\mathbf{u}_h)_{w_u^k(i)} - (\mathbf{y}_h^k)_{w_u^k(i)} \qquad i = 1,\dots,n_u^k\,. \tag{6.10b}$$

Then the saddle problem to be solved at iteration k is:

$$\begin{bmatrix} \mathbf{A}_h & \mathbf{B}_h^T & \mathbf{C}_h^{k^T} & \mathbf{D}_h^{k^T} \\ \mathbf{B}_h & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{C}_h^k & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}_h^k & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_h^k \\ \mathbf{z}_h^k \\ \boldsymbol{\lambda}_{h,l}^k \\ \boldsymbol{\lambda}_{h,u}^k \end{bmatrix} = \begin{bmatrix} \mathbf{f}_h - \mathbf{A}_h\mathbf{y}_h^k \\ \mathbf{h}_h - \mathbf{B}_h\mathbf{y}_h^k \\ \mathbf{l}_h^k \\ \mathbf{u}_h^k \end{bmatrix}, \tag{6.11}$$

a system that can be briefly written as:

$$\mathbf{G}_h^k \mathbf{s}_h^k = \mathbf{t}_h^k\,. \tag{6.12}$$

To determine how to modify the active sets W_l^k and W_u^k, we use Algorithm 1. Every time we change the active sets, a new linear system needs to be solved. If the dimension of the problem increases, the number of the constraints increases as well and more iterations are required to identify the correct active set. Nevertheless, if the active set method is used for relatively small problems, then it will converge within a few iterations. The subspace correction method and, in particular, the monotone multigrid method take advantage of this fact. Indeed, the constraints are tackled not at a global level, but locally on different and sufficiently small fine and coarse subspaces.

6.4 Subspace correction methods

In Xu, Jinchao [1992] a unified framework for different iterative solvers such as Jacobi, Gauss-Seidel, domain decomposition and multigrid methods is presented. The idea is to find a decomposition of $\mathbf{Y}_h$ into N subspaces $\{\mathbf{Y}_i\}_{i=1}^N$ so that:

$$\bigcup_{\mathrm{i=1}}^{\mathrm{N}} \mathbf{Y}_i = \mathbf{Y}_h\,, \tag{6.13}$$

Algorithm 1: Active-set method

```
Result: $\mathbf{y}_h$, $\mathbf{z}_h$
Input: $\mathbf{y}_h^0$, $W_l^0$, $W_u^0$
$k = -1$
stop = true
while stop do
    $k \leftarrow k + 1$
    Define $\mathbf{C}_h^k$ and $\mathbf{D}_h^k$ as in (6.9)
    Define $\mathbf{l}_h^k$ and $\mathbf{u}_h^k$ as in (6.10)
    Find $\mathbf{s}_h^k$ and its components $\mathbf{c}_h^k$, $\mathbf{z}_h^k$, $\boldsymbol{\lambda}_{h,l}^k$, $\boldsymbol{\lambda}_{h,u}^k$, by solving (6.11)
    $\mathbf{r}_h^k = \mathbf{f}_h^k - \mathbf{G}_h^k \mathbf{s}_h^k$
    $W_l^{k+1} \leftarrow W_l^k$
    $W_u^{k+1} \leftarrow W_u^k$
    stop = false
    for $i = 1, \ldots, n$ do
        if $w_l^k(i) \in W_l^k$ and $(\boldsymbol{r}_h^k)_i > 0$ then
            $W_l^{k+1} \leftarrow W_l^{k+1} \setminus \{i\}$
            stop = true
        if $w_u^k(i) \in W_u^k$ and $(\boldsymbol{r}_h^k)_i < 0$ then
            $W_u^{k+1} \leftarrow W_u^{k+1} \setminus \{i\}$
            stop = true
        if $(\boldsymbol{c}_h^k)_i < (\boldsymbol{l}_h^k)_i$ then
            $W_l^{k+1} \leftarrow W_l^{k+1} \cup \{i\}$
            stop = true
        if $(\boldsymbol{c}_h^k)_i > (\boldsymbol{u}_h^k)_i$ then
            $W_u^k \leftarrow W_u^{k+1} \cup \{i\}$
            stop = true
    end
    $\mathbf{y}_h^{k+1} \leftarrow \mathbf{y}_h^k + \mathbf{c}_h^k$
end
$\mathbf{y}_h \leftarrow \mathbf{y}_h^k$
$\mathbf{z}_h \leftarrow \mathbf{z}_h^k$
```

where each $\mathbf{Y}_i$ is a small subspace of $\mathbf{Y}_h$, i.e., $\dim(\mathbf{Y}_i) \ll n$. The subspaces of the decomposition can be overlapping, if $\mathbf{Y}_i \cap \mathbf{Y}_j \neq \emptyset$ for at least two $i, j \in \{1, \ldots, N\}$ with $i \neq j$, or non-overlapping, if $\mathbf{Y}_i \cap \mathbf{Y}_j = \emptyset$ for $i, j = 1, \ldots, N$ with $i \neq j$. Furthermore we define the interpolation operator between the subspace $\mathbf{Y}_i$ and the space $\mathbf{Y}_h$, as $\mathbf{\Pi}_i : \mathbf{Y}_i \rightarrow \mathbf{Y}_h$.

The subspace correction method computes local corrections $\mathbf{c}_i$ on each subspace $\mathbf{Y}_i$ such that the energy functional $\mathcal{J}_h$ is minimized. There are mainly two general ways of computing the corrections. The parallel subspace corrections (PSC) method computes all the corrections $\mathbf{c}_i$, for $i = 1, \ldots, N$, in parallel. In order to make the Algorithm 2 convergent, a proper scalar η must be given as input. Even though this process can be carried out in parallel, it is rather slow. In order to accelerate the convergence, a sequential subspace correction approach (SSC) can be examined. This method computes sequentially each correction and updates the current solution right afterward, as it is shown in Algorithm 3. In both cases, the minimization of a quadratic functional over a closed convex set can be carried out by using Algorithm 1.

Algorithm 2: $\mathbf{y}_h^s$=ParallelSubspaceCorrection($\mathbf{y}_h^0$, s, $\mathcal{J}_h$, $\mathbf{K}_h$, η)

```
for $k = 1, \ldots, s$ do
    for $i = 1, \ldots, N$ do
        $\mathrm{K}_i(\mathbf{y}_h^k) := \{\mathbf{c}_i \in \mathbf{Y}_i : \mathbf{y}_h^k + \mathbf{c}_i \in \mathbf{K}_h\}$
        $\mathbf{c}_i \leftarrow \underset{\mathbf{c}_i \in \mathrm{K}_i(\mathbf{y}_h^k)}{\mathrm{argmin}} \mathcal{J}_h(\mathbf{y}_h^k + \mathbf{\Pi}_i \mathbf{c}_i)$
    end
    $\mathbf{y}_h^k \leftarrow \mathbf{y}_h^{k-1} + \sum_{i=1}^{N} \eta \mathbf{\Pi}_i \mathbf{c}_i$
end
```

If the convex set $\mathbf{K}_h$ coincides with $\mathbf{Y}_h$ and if we identify $\mathbf{Y}_h$ with $\mathbb{R}^n$ and each subspace $\mathbf{Y}_i$ with the i-th component of a vector in $\mathbb{R}^n$, the PSC and SSC schemes presented above coincide with the Jacobi and Gauss-Seidel methods. On the other hand, if $\mathbf{K}_h$ is characterized by box-constraints, the corresponding SSC method is called projected Gauss-Seidel. It is well known that the convergence rate of the Jacobi and Gauss-Seidel methods deteriorates with the increase of the problem size.

Remark 6.4.1. *The sequential minimization of the functional as in Algorithm 3 should guarantee, in the limit for $s \rightarrow \infty$, the convergence of the algorithm. At least, no divergence of the solution is permitted, because the energy is always controlled. On the other hand, for Algorithm 2, a proper parameter η is necessary*

Algorithm 3: $\mathbf{y}_h^s$=SequentialSubspaceCorrection($\mathbf{y}_h^0$, s, $\mathcal{J}_h$, $\mathbf{K}_h$)

```
for $k = 1,\ldots,s$ do
    $\mathbf{y}_0^k \leftarrow \mathbf{y}_h^{k-1}$
    for $i = 1,\ldots,N$ do
        $\mathbf{K}_i(\mathbf{y}_{i-1}^k) := \{\mathbf{c}_i \in \mathbf{Y}_i : \mathbf{y}_{i-1}^k + \mathbf{c}_i \in \mathbf{K}_h\}$
        $\mathbf{c}_i \leftarrow \underset{\mathbf{c}_i \in \mathbf{K}_i(\mathbf{y}_{i-1}^k)}{\operatorname{argmin}} \mathcal{J}_h(\mathbf{y}_{i-1}^k + \mathbf{\Pi}_i \mathbf{c}_i)$
        $\mathbf{y}_i^k \leftarrow \mathbf{y}_{i-1}^k + \mathbf{\Pi}_i \mathbf{c}_i$
    end
    $\mathbf{y}_h^k \leftarrow \mathbf{y}_N^k$
end
```

for the convergence. Even though locally each correction is computed by energy minimization, the overall correction $\mathbf{c} = \sum_{i=1}^N \mathbf{c}_i$ *does not necessarily minimize the energy. This can happen because the corrections are computed independently from each other. Thus, to ensure a monotone minimization of the functional, it is important to employ, for example, a line search strategy onto the global correction* $\mathbf{c}$.

Remark 6.4.2. *In the original work by Xu, Jinchao [1992], the subspace correction is carried out in terms of error reduction instead of energy minimization. Indeed the main problem consists of the solution of a linear system and no energy is mentioned. Nevertheless, a linear symmetric positive definite system can always be interpreted as the first-order necessary condition for the minimization of a quadratic functional on the global space, with no constraints. In this case, the local minimization process coincides with the solution of a local linear subproblem.*

6.5 Smoothers

A smoother is a method which can rapidly damp the high-frequency components of the error. After the high frequencies are removed from the error, the smoother has a small impact on its low-frequency components. Furthermore, as the problem size increases, the more this behavior becomes evident and its convergence property deteriorates.

A well-known smoother for symmetric positive definite systems is the conjugate gradient method. It is examined, as a solver, in Nocedal, Jorge and Wright, Stephen [2006]. It uses global information, like the residual of the problem, to damp the high-frequency components of the error. Assuming exact arithmetic, it converges in no more than n iterations, where n is the dimension of the system. However, for very large n and using inexact arithmetic, its convergence behavior can deteriorate, in agreement with the typical smoother behavior. As already mentioned, a similar performance concerns the Jacobi and the Gauss-Seidel methods. Indeed, some subspace correction methods can be used as smoothers for damping high-frequency components of the error. Indeed, as it is explained in Remark 6.4.2, the error decreasing and the energy minimization can be correlated. We call *number of smoothing steps* the parameter s in Algorithm 2 and in Algorithm 3. Besides, the same algorithms for $s = 1$ are known as single smoothing steps.

It is thus clear that standard methods as Jacobi, Gauss-Seidel, or conjugate gradient, are not very well suited for solving very large problems, because their rate of convergence depends on the dimension of the problem. Historically, at least for elliptic problems, the multigrid method has been invented to get a rate of convergence independent of the dimension of the problem. In the next sections, we will introduce the multigrid method and we will examine it also for constrained problems of the Signorini problem's type.

6.6 Monotone multigrid methods (MMG)

The multilevel idea is to represent the error on coarser subspaces, so that its low-frequency components become high-frequency components on these coarser subspaces and can still be easily damped with proper smoothers. Since the more levels we add, the more frequencies we can capture, a multigrid method has the optimal convergence property: the convergence rate is independent of the dimension of the problem n. For some references concerning the optimal convergence of the multigrid methods, see Bank, Randolph E. and Yserentant, Harry [2010], Chen, Zhangxin [1994], Gelman, E. and Mandel, J. [1990], Mandel, Jan [1984].

Since the convex set in (6.5b) is characterized by local inequality and global equality constraints, the multigrid to be investigated is non-linear. Multigrid methods for the Signorini problem expressed in the primal formulation have already been examined. This is the case, for example, of the monotone multigrid (MMG) method. See Badea [2002], Badea [2014], Bader and Hoppe [1993], Badea and Krause [2012], Kornhuber, Ralf and Krause, Rolf [2001], Kornhuber, Ralf and Krause, Rolf and Sander, O. and Deuflhard, P. and Ertel, S. [2008], Krause [2009], Krause, Rolf and Rigazzi, Alessandro and Steiner, Johannes

[2016]. The MMG method is a generalization of Algorithm 3. Its goal is to sequentially minimize the energy as Algorithm 3 but with optimal complexity. To this purpose, the spaces $\mathbf{Y}_i$ used in Algorithm 3, which can be interpreted as spanned by the high-frequency components of the error, are not sufficient. However, subspaces spanned by low-frequency components can be added as well. In this way, the energy is minimized over different fine and coarse subspaces and optimal complexity can be recovered. The aim of this section is to discuss in more details the MMG method applied to the primal, the dual and the FOSLS formulations, i.e., (5.50), (5.52), (5.57), discretized by means of the FE method.

In the finite element framework, we introduce a sequence of nested tessellations $\{\mathcal{T}_0\}_{j=0}^{J}$ such that $\mathcal{T}_0 \subset \mathcal{T}_1 \subset \cdots \subset \mathcal{T}_{J-1} \subset \mathcal{T}_J := \mathcal{T}_h$. On these meshes, we can define a sequence of nested subspaces $\mathbf{Y}_0 \subset \mathbf{Y}_1 \subset ... \subset \mathbf{Y}_{J-1} \subset \mathbf{Y}_J := \mathbf{Y}_h$, where each $\mathbf{Y}_j$ is the coarse subspace on the level j of the space $\mathbf{Y}_J$, associated with the mesh $\mathcal{T}_j$. We also denote by $\mathbf{\Pi}_j^{j+1} : \mathbf{Y}_j \rightarrow \mathbf{Y}_{j+1}$ the interpolation operator between the levels j and $j+1$, for $j \in \{0,\dots,J-1\}$. Furthermore, we denote by $\mathbf{\Pi}_j^J : \mathbf{Y}_j \rightarrow \mathbf{Y}_J$ the interpolation operator between the levels j and J, for $j \in \{0,\dots,J-1\}$. For each level j, we also define a decomposition $\mathbf{Y}_j = \sum_{i=1}^{n_j} \mathbf{Y}_{j,i}$, where n_j is the number of subspaces on the level j. To relate $\mathbf{Y}_{j,i}$ to $\mathbf{Y}_j$, we need to define an interpolation operator $\mathbf{\Pi}_{j,i} : \mathbf{Y}_{j,i} \rightarrow \mathbf{Y}_j$. We also consider a sequence of nested subspaces $\mathbf{Z}_0 \subset \mathbf{Z}_1 \subset ... \subset \mathbf{Z}_{J-1} \subset \mathbf{Z}_J = \mathbf{Z}_h$ for the Lagrange multipliers. By $\mathbf{Q}_j^{j+1} : \mathbf{Z}_j \rightarrow \mathbf{Z}_{j+1}$ we denote the interpolation operator between the levels j and $j+1$, for $j \in \{0,\dots,J-1\}$. By $\mathbf{Q}_j^J : \mathbf{Z}_j \rightarrow \mathbf{Z}_J$ we denote the interpolation operator between the levels j and J, for $j \in \{0,\dots,J-1\}$. Finally, we also define a decomposition $\mathbf{Z}_j = \sum_{i=1}^{n_j} \mathbf{Z}_{j,i}$ and the interpolations $\mathbf{Q}_{j,i} : \mathbf{Z}_{j,i} \rightarrow \mathbf{Z}_j$.

The fine problem (6.5) can be rewritten on level $j = J$ by substituing the subscript h with the subscript J. For all other levels $j = J-1,\dots,0$ and for a fixed $\mathbf{y}_{j+1}$, the coarse problem at level j reads as follows:

$$\mathcal{J}_j(\mathbf{y}_j;\mathbf{y}_{j+1}) = \frac{1}{2}\mathbf{y}_j^T\mathbf{A}_j(\mathbf{y}_{j+1})\mathbf{y}_j - \mathbf{y}_j^T\mathbf{f}_j(\mathbf{y}_{j+1}), \tag{6.14a}$$

$$\mathbf{K}_j(\mathbf{y}_{j+1}) = \{\mathbf{y}_j \in \mathbf{Y}_j : \quad \mathbf{B}_j(\mathbf{y}_{j+1})\mathbf{y}_j = \mathbf{h}_j(\mathbf{y}_{j+1}), \quad \mathbf{l}_j(\mathbf{y}_{j+1}) \le \mathbf{y}_j \le \mathbf{u}_j(\mathbf{y}_{j+1})\}, \tag{6.14b}$$

where:

$$\mathbf{A}_j(\mathbf{y}_{j+1}) = \left[\mathbf{\Pi}_j^{j+1}\right]^T \mathbf{A}_{j+1}\mathbf{\Pi}_j^{j+1}\,, \tag{6.15a}$$

$$\mathbf{B}_j(\mathbf{y}_{j+1}) = \left[\mathbf{Q}_j^{j+1}\right]^T \mathbf{B}_{j+1}\mathbf{\Pi}_j^{j+1}\,, \tag{6.15b}$$

$$\mathbf{f}_j(\mathbf{y}_{j+1}) = \left[\mathbf{\Pi}_j^{j+1}\right]^T \left(\mathbf{f}_{j+1} - \mathbf{A}_{j+1}\mathbf{y}_{j+1}\right)\,, \tag{6.15c}$$

$$\mathbf{h}_j(\mathbf{y}_{j+1}) = \left[\mathbf{Q}_j^{j+1}\right]^T (\mathbf{h}_{j+1} - \mathbf{B}_{j+1}\mathbf{y}_{j+1})\,, \tag{6.15d}$$

$$\mathbf{l}_j(\mathbf{y}_{j+1}) = \mathbf{R}_{l,j+1}^j(\mathbf{l}_{j+1} - \mathbf{y}_{j+1})\,, \tag{6.15e}$$

$$\mathbf{u}_j(\mathbf{y}_{j+1}) = \mathbf{R}_{u,j+1}^j(\mathbf{u}_{j+1} - \mathbf{y}_{j+1})\,. \tag{6.15f}$$

The vectors $\mathbf{f}_j$, $\mathbf{h}_j$, $\mathbf{l}_j$ and $\mathbf{u}_j$ directly depend on $\mathbf{y}_{j+1}$. The operators $\mathbf{R}_{l,j+1}^j$ and $\mathbf{R}_{u,j+1}^j$ are the so called *monotone restrictions*, non-linear projection operators for box-constraints. The matrices $\mathbf{A}_j$ and $\mathbf{B}_j$ can also depend on $\mathbf{y}_{j+1}$, despite the expressions in (6.15a) and (6.15b), if *truncation of the basis functions* is used. Both monotone restrictions and truncation of the basis will be discussed later in sections 6.8 and 6.9.

The MMG method works as follows. Given an initial guess $\mathbf{y}_J^0$, the number of pre and post-smoothing steps ν_{pre} and ν_{post}, the MMG method updates $\mathbf{y}_J^{k+1}$=VCycle($\mathbf{y}_J^k$, ν_{pre}, ν_{post}) for the iterations $k = 0, 1, \ldots$, until convergence. See Algorithm 4 and Algorithm 5, where the pseudocode, respectively for the smoothing and the V-cycle, is presented. For determining the numerical convergence, a criterion on the norm of the residual is used.

Algorithm 4: $\mathbf{y}_j^k$=Smoothing($\mathbf{y}_{j,0}^k$, ν, $\mathcal{J}_j$, $\mathbf{K}_j$)

```
for $s = 1, \ldots, \nu$ do
    for $i = 1, \ldots, n_j$ do
        $\mathcal{K}_{j,i}(\mathbf{y}_{j,i-1}^k) := \{\mathbf{c}_{j,i} \in \mathbf{Y}_{j,i} : \mathbf{y}_{j,i-1}^k + \mathbf{\Pi}_{j,i}\mathbf{c}_{j,i} \in \mathbf{K}_j\}$
        $\mathbf{c}_{j,i} \leftarrow \arg\min_{\mathbf{c}_{j,i} \in \mathbf{K}_{j,i}(\mathbf{y}_{j,i-1}^k)} \mathcal{J}_j(\mathbf{y}_{j,i-1}^k + \mathbf{\Pi}_{j,i}\mathbf{c}_{j,i})$
        $\mathbf{y}_{j,i}^k \leftarrow \mathbf{y}_{j,i-1}^k + \mathbf{\Pi}_{j,i}\mathbf{c}_{j,i}$
    end
    $\mathbf{y}_j^k \leftarrow \mathbf{y}_{j,n_j}^k$
end
```

Let $\mathbf{y}_J^0 \in \mathbf{K}_J$. Then, for $k = 0, 1, \ldots$ and $j = J-1, \ldots, 0$, two are the possible scenarios:

Algorithm 5: $\mathbf{y}_J^{k+1}$=VCycle($\mathbf{y}_J^k$, ν_{post}, ν_{post})

```
$\mathbf{y}_J^k$=Smoothing($\mathbf{y}_J^k$, $\nu_{\text{pre}}$, $\mathcal{J}_J(\cdot)$, $\mathbf{K}_J$)
for $j = J-1, \dots, 1$ do
    $\mathbf{y}_j^k \leftarrow 0$
    $\mathbf{y}_j^k$=Smoothing($\mathbf{y}_j^k$, $\nu_{\text{pre}}$, $\mathcal{J}_j(\cdot; \mathbf{y}_{j+1}^k)$, $\mathbf{K}_j(\mathbf{y}_{j+1}^k)$)
end
$\mathbf{y}_0^k \leftarrow \arg\min_{\mathbf{y}_0 \in \mathbf{K}_0(\mathbf{y}_1^k)} \mathcal{J}_0(\cdot; \mathbf{y}_1^k)$
for $j = 1, \dots, J-1$ do
    $\mathbf{y}_j^k \leftarrow \mathbf{y}_j^k + \mathbf{\Pi}_{j-1}^j \mathbf{y}_{j-1}^k$
    $\mathbf{y}_j^k$=Smoothing($\mathbf{y}_j^k$, $\nu_{\text{post}}$, $\mathcal{J}_j(\cdot; \mathbf{y}_{j+1}^k)$, $\mathbf{K}_j(\mathbf{y}_{j+1}^k)$)
end
$\mathbf{y}_J^k$=Smoothing($\mathbf{y}_J^k$, $\nu_{\text{post}}$, $\mathcal{J}_J(\cdot)$, $\mathbf{K}_J$)
$\mathbf{y}_J^{k+1} \leftarrow \mathbf{y}_J^k$
```

- It is fulfilled the following inclusion:

$$\left(\mathbf{\Pi}_j^{j+1} \mathbf{y}_j^k\right) + \mathbf{y}_{j+1}^k \in \mathbf{K}_{j+1} \tag{6.16}$$

  where $\mathbf{y}_j^k \in \mathbf{K}_j(\mathbf{y}_{j+1}^k)$ is the resulting correction of the smoothing process. Since all the corrections on the fine and on the coarse levels satisfy the fine constraints, the energy functional is sequentially minimized and the multigrid is monotone.

- Otherwise, there is at least a coarse correction that violates the fine constraints. In this case, the multigrid is not monotone and the value of the energy functional could temporarily increase. However, to get convergence, the behavior of the overall correction of a V-cycle is what really matters. If a single local correction makes the energy functional increase, the convergence process is not necessarily badly affected.

Depending on the definition of the coarse convex sets $\mathbf{K}_j$ for $j = J-1, \dots, 0$, either the first or the second situation is obtained. For example, to get the first scenario, the fine constraints could be enforced on the coarser levels as well. In this way, all the coarse corrections would never violate the fine constraints. Nevertheless, this choice would lead to suboptimal complexity, because quantities on the coarser levels would be compared to the ones on the fine level.

On the fine level, the inequality constraints are just box constraints, i.e., the unknowns can be compared component-wise to the lower and upper bounds.

However, on a coarse level j, the comparison with the bounds on the fine level is not component-wise anymore, but instead involves linear combinations of the unknown $\mathbf{y}_j$. Indeed for $j = J-1, \dots, 0$, it would be necessary to satisfy:

$$\mathbf{l}_{j+1} - \mathbf{y}_{j+1} \leq \mathbf{\Pi}_j^J \mathbf{y}_j \leq \mathbf{u}_{j+1} - \mathbf{y}_{j+1} \,. \tag{6.17}$$

In order to avoid this, we can define $\mathbf{K}_j$ by using bounds which belongs to $\mathbf{Y}_j$ instead of $\mathbf{Y}_J$. In this way, the comparison would still be component-wise. Furthermore, if the corrections do not violate the constraints on the fine level, the multigrid is monotone. To this aim, the monotone restriction operators $\mathbf{R}_{l,j+1}^j$ and $\mathbf{R}_{u,j+1}^j$, as in the equations (6.15e) and (6.15f), can be exploited. Monotone restrictions, introduced in Kornhuber, Ralf [1994], are non-linear projections for constraints from the level $j+1$ to the level j, for $j = J-1, \dots, 0$. They are defined so that on the finer level $j+1$:

$$\mathbf{l}_{j+1} - \mathbf{y}_{j+1} \leq \mathbf{\Pi}_j^{j+1} \mathbf{R}_{l,j+1}^j (\mathbf{l}_{j+1} - \mathbf{y}_{j+1}) \leq \mathbf{\Pi}_j^{j+1} \mathbf{R}_{u,j+1}^j (\mathbf{u}_{j+1} - \mathbf{y}_{j+1}) \leq \mathbf{u}_{j+1} - \mathbf{y}_{j+1} \,, \tag{6.18}$$

while the following component-wise relation holds for the correction $\mathbf{y}_j$ on the coarse level j:

$$\mathbf{R}_{l,j+1}^j (\mathbf{l}_{j+1} - \mathbf{y}_{j+1}) \leq \mathbf{y}_j \leq \mathbf{R}_{u,j+1}^j (\mathbf{u}_{j+1} - \mathbf{y}_{j+1}) \,. \tag{6.19}$$

Even though relations between coarse and fine vectors are presented, the monotone restriction operators can exploit the FE discretization information. In particular, they can be optimized for piecewise linear functions as shown in Kornhuber, Ralf [1994]. This is the case of the traces of linear Lagrangian and first-order Raviart-Thomas elements. Nevertheless, for simplicity of implementation, a more strict definition of restriction operator can be introduced. Given $P_j^{j+1}(i) := \{k \in \mathbb{N} : (\mathbf{\Pi}_j^{j+1})_{k,i} \neq 0\}$:

$$(\mathbf{R}_{l,j+1}^j (\mathbf{l}_{j+1} - \mathbf{y}_{j+1}))_i = \max_{k \in P_j^{j+1}(i)} (\mathbf{l}_{j+1} - \mathbf{y}_{j+1})_k \,, \tag{6.20a}$$

$$(\mathbf{R}_{u,j+1}^j (\mathbf{u}_{j+1} - \mathbf{y}_{j+1}))_i = \min_{k \in P_j^{j+1}(i)} (\mathbf{u}_{j+1} - \mathbf{y}_{j+1})_k \,, \tag{6.20b}$$

where by definition, if $\mathbf{y}_{j+1}$ is feasible, $(\mathbf{l}_{j+1} - \mathbf{y}_{j+1}) \leq 0$ and $(\mathbf{u}_{j+1} - \mathbf{y}_{j+1}) \geq 0$. In this way, the relation (6.16) holds and the multigrid is still monotone. However, if the restriction process is too aggressive, the convergence rate can be negatively affected. Indeed, for example in (6.20b), it is sufficient that exists k such that $(\mathbf{u}_{j+1})_k = (\mathbf{y}_{j+1})_k$, so that $(\mathbf{R}_{u,j+1}^j (\mathbf{u}_{j+1} - \mathbf{y}_{j+1}))_i = 0$. Similarly this also holds for the lower bound. However this situation can be solved by truncating the basis functions. For this reason, after the discussion on global equality constraints, the truncation of the basis functions will be studied as well in section 6.8, while the monotone restrictions operators for piecewise constant and linear functions will be studied in section 6.9.

In contrast to box-constraints, the global equality constraints on the fine level, which are enforced also on the coarser level, would overconstrain the coarse problems. To avoid this difficulty, the global equality constraints need to be projected, as in (6.14b). However, unlike the inequality constraints, they cannot be projected onto the coarser levels so that (6.16) is fulfilled. Therefore monotonicity of the multigrid can be guaranteed for the only primal and the FOSLS formulations. On the other hand, even if monotone restrictions for the inequality constraints are used, no monotonicity can be a priori guaranteed for the dual formulation. However, since the structure of the algorithm is the same as for the primal formulation, we will still use the name MMG method.

In conclusion, typical ingredients of the multigrid method are the interpolation operator and the smoother, which will be examined in section 6.7 and in section 6.11. In particular, for the mixed formulations which involve unknowns belonging to $\mathbf{H}_{\text{div}}(\Omega)$ space, the smoother has to be examined with special care. On the other hand, more specific ingredients of the MMG methods are the monotone restrictions and the trucated basis that will be discussed in sections 6.8 and 6.9.

6.7 Interpolation operators

Let us consider two nested meshes:

$$\mathcal{T}_C := \mathcal{T}_j\,, \qquad \mathcal{T}_F := \mathcal{T}_{j+1}\,, \tag{6.21}$$

and the corresponding FE and coefficients spaces:

$$\mathbf{X}_C := \mathbf{X}_j\,, \quad \mathbf{X}_F := \mathbf{X}_{j+1}\,, \quad \mathbf{Y}_C := \mathbf{Y}_j\,, \quad \mathbf{Y}_F := \mathbf{Y}_{j+1}\,, \tag{6.22}$$

respectively with dimensions:

$$n_C = \dim(\mathbf{Y}_C)\,, \qquad n_F = \dim(\mathbf{Y}_F)\,. \tag{6.23}$$

From now on, by the subpscripts C and F we will denote quantities respectively on coarse and fine levels. Let $\{\boldsymbol{\phi}_{C,1},\dots,\boldsymbol{\phi}_{C,n_C}\}$ and $\{\boldsymbol{\phi}_{F,1},\dots,\boldsymbol{\phi}_{F,n_F}\}$ be the shape functions on the coarse and fine levels, respectively defined so that:

$$l_{C,i}(\boldsymbol{\phi}_{C,j}) = \delta_{ij} \quad i,j = 1,\dots,n_C\,, \tag{6.24}$$

$$l_{F,i}(\boldsymbol{\phi}_{F,j}) = \delta_{ij} \quad i,j = 1,\dots,n_F\,, \tag{6.25}$$

where $\{l_{C,1},\dots,l_{C,n_C}\}$ and $\{l_{F,1},\dots,l_{F,n_F}\}$ are the degrees of freedom on the coarse and on the fine levels. In practice, we want to interpolate the vector of the coefficients of a function belonging to a coarse level, so that it is represented on the fine mesh. Then the interpolation operator $\boldsymbol{\Pi}_C^F : \mathbf{Y}_C \to \mathbf{Y}_F$ is a matrix $\boldsymbol{\Pi}_C^F \in \mathbb{R}^{n_F,n_C}$. Its components are defined as follows:

$$(\boldsymbol{\Pi}_C^F)_{i,j} = l_{F,i}(\boldsymbol{\phi}_{C,j}) \qquad i = 1,\dots,n_F, \quad j = 1,\dots,n_C\,. \tag{6.26}$$

It is clear from its definition that the interpolation requires the evaluation of each fine degree of freedom on each coarse basis function. However, the resulting matrix is typically sparse. Indeed the support of finite element basis functions is usually restricted to few elements, so this evaluation involves only a small number of basis functions. Furthermore, if $\mathcal{T}_C = \mathcal{T}_F$, the interpolation operator must coincide with the identity operator. For this to happen, (5.3) must hold. This is the reason of the discussion in section 5.5.2 regarding RT_1 basis functions defined directly on the actual element K instead of the reference element $\hat{K}$. For the definition of $\boldsymbol{\Pi}_C^F$ involving RT_1 functions, the RT_1 shape functions have to be computed on K.

Typically, a degree of freedom l applied to a given function ϕ, i.e., $l(\phi)$, is a particular integral of the function itself. Thanks to quadrature rules, it boils down just to an evaluation of ϕ in a finite number of points p_k of coordinates $\mathbf{p}_k \in K$, for $k = 1,...,n_{\text{qr}}$, where n_{qr} is the number of points for the quadrature rule. On the other hand, if the dof is defined as a point-wise evaluation of the function, then only one node $\mathbf{p}$ is needed. However the dof l and the points $\mathbf{p}_k$ are defined on K_F, but the shape function ϕ is defined on K_C. This requires particular care for the mappings between the reference and the actual elements.

For example, let us assume that $\phi_{C,j}$ is a basis function of the local FE space $\mathcal{S}_C$. For evaluating $l_{F,i}(\phi_{C,j})$, we need to know $\phi_{C,j}(\mathbf{p}_k)$, where $\mathbf{p}_k$ are points belonging to the fine element K_F for $k = 1,...,4$ (see figure 6.1). If the shape functions are computed on $\hat{K}_C$, we must map the points $\mathbf{p}_k \in K_F$ onto it, obtaining $\hat{\mathbf{p}}_k = F_C^{-1}(\mathbf{p}_k)$, where $F_C : \mathbb{R}^d \to \mathbb{R}^d$, defined in (5.8), is the geometric map between $\hat{K}_C$ and K_C. The reference basis function can then be evaluated in these points. The resulting values $\hat{\phi}(\hat{\mathbf{p}}_k)$ can be mapped back to the actual element, giving rise to $\phi(\mathbf{p}_k) = M_{\mathcal{S}_C,I}(\hat{\phi}(\hat{\mathbf{p}}_k))$. In this way $l_{F,i}(\phi_{C,j}) = l_{F,i}(\phi_{C,j}(p_1),\dots,\phi_{C,j}(p_4))$ can be finally computed.

If the points $\hat{\mathbf{p}}_k$ are known a priori, also $\hat{\phi}(\hat{\mathbf{p}}_k)$ can be computed a priori. Therefore the first mapping $\hat{\mathbf{p}}_k = F_C^{-1}(\mathbf{p}_k)$ would not be necessary. This is the case when we compute standard surface/volume integrals on the whole surface/volume of the element, for example in the assembly of the linear and bilinear forms. Nevertheless, for the calculation of $(\mathbf{\Pi}_C^F)_{i,j}$, the domain of integration is a subset of the coarse element which depends on the run-time refinement process. Therefore the points $\mathbf{p}_k$ can vary from element to element and the computation $\hat{\mathbf{p}}_k = F_C^{-1}(\mathbf{p}_k)$ can only be done at run-time.

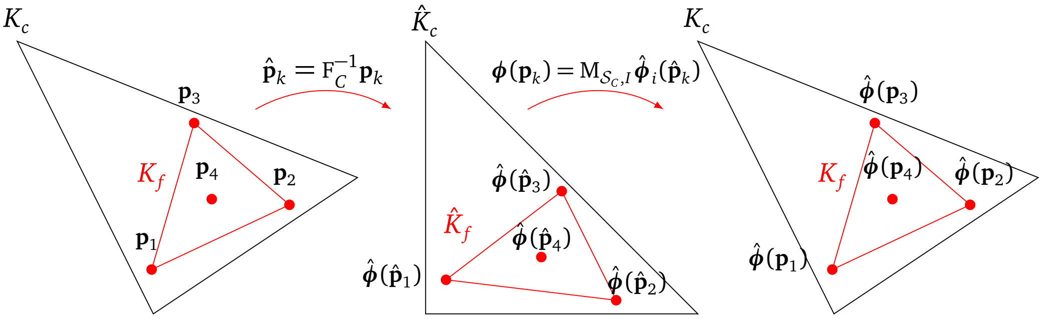

K_c
$\hat{\mathbf{p}}_k = \mathrm{F}_C^{-1}\mathbf{p}_k$
$\hat{K}_c$
$\boldsymbol{\phi}(\mathbf{p}_k) = \mathrm{M}_{\mathcal{S}_C,I}\hat{\boldsymbol{\phi}}_i(\hat{\mathbf{p}}_k)$
K_c
$\mathbf{p}_3$
K_f
$\mathbf{p}_4$
$\mathbf{p}_2$
$\mathbf{p}_1$
$\hat{\boldsymbol{\phi}}(\hat{\mathbf{p}}_3)$
$\hat{K}_f$
$\hat{\boldsymbol{\phi}}(\hat{\mathbf{p}}_4)$
$\hat{\boldsymbol{\phi}}(\hat{\mathbf{p}}_1)$
$\hat{\boldsymbol{\phi}}(\hat{\mathbf{p}}_2)$
$\hat{\boldsymbol{\phi}}(\mathbf{p}_3)$
K_f
$\hat{\boldsymbol{\phi}}(\mathbf{p}_4)$
$\hat{\boldsymbol{\phi}}(\mathbf{p}_2)$
$\hat{\boldsymbol{\phi}}(\mathbf{p}_1)$

Figure 6.1. Mapping from the fine actual element to the corresponding coarse reference element; computing the shape functions in the desired reference nodes; mapping the obtained values to the actual element.

Remark 6.7.1. *The discussion presented here and in the sequel for truncation and monotone restrictions is based on the particular choice of the basis functions. The construction of the interpolation or the restriction operators cannot be expected to be necessarily generalized to other discretizations, like NURBS.*

6.8 Truncation of the basis

As previoulsy explained, the monotone restriction operators could produce too restrictive box-constraints on the coarser levels. The spaces $\{\mathbf{Y}_j\}_{j=0}^J$ refer to coefficients of FE functions belonging to $\{\mathbf{X}_j\}_{j=0}^J$. In particular, it is possible to relate each i-th component of a correction $\mathbf{y}_j \in \mathbf{Y}_j$ to the i-th shape function of the corresponding FE space $\mathbf{X}_j$ on level j. Since the hierarchy is nested, i.e., $\mathbf{X}_0 \subset ... \subset \mathbf{X}_J$, the basis functions of the coarse FE spaces can be expressed as linear combinations of the basis functions on the fine level, for $j = 0, ..., J-1$. During the iterative multigrid process, a box-constraint on the level $j+1$ can become active. This means that, at the iteration k, exists $i \in \{1, \ldots, n_{j+1}\}$ such that $(\mathbf{l}_{j+1}^k)_i = (\mathbf{y}_{j+1}^k)_i$ or

$(\mathbf{u}^k_{j+1})_i = (\mathbf{y}^k_{j+1})_i$. This situation would produce a too restrictive box-constraint on all levels $s = j-1, \dots, 0$. Thus, we can assume that, at least for this iteration, the i-th constraint on level $j+1$ has to be active, the effective value $(\mathbf{y}^k_{j+1})_i$ is known and no further correction from the coarser spaces is needed. In practice, this means there is no need for other corrections in the direction of the subspace related to the dof i of level $j+1$ and thus the corresponding basis function is not required anymore.

Therefore, we could temporarily remove this basis function from the basis of the level $j+1$. In other words, we would *truncate* the basis. As a consequence, the basis of the coarse level j would change, being built as a linear combination of the truncated basis from level $j+1$. Recursively, the same will happen for all other levels $0, \dots, j-1$. In this way, from the coarse levels there will be no corrections influencing the active dof i on level $j+1$. In Figure 6.2, we show the effect on a 1D example for the truncation of a shape function $\phi_{F,3}$. The corresponding coarse basis function $\phi_{C,2} = \frac{1}{2}\phi_{F,2} + \phi_{F,3} + \frac{1}{2}\phi_{F,4}$, that is a hat function, is consequently modified as $\phi_{C,2} = \frac{1}{2}\phi_{F,2} + \frac{1}{2}\phi_{F,4}$ and is no more a hat function.

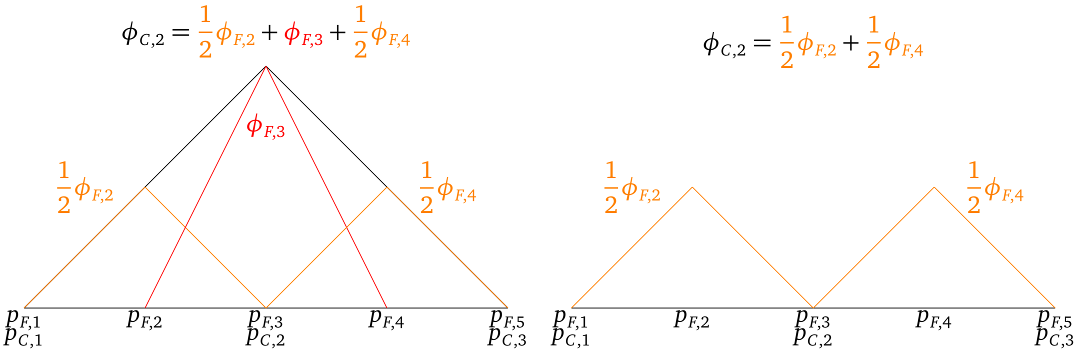


Figure 6.2. The coarse basis function $\phi_{C,2}$ in black is hat function given by the linear combination of $\phi_{F,2}$, $\phi_{F,3}$, $\phi_{F,4}$. If $\phi_{F,3}$ is truncated, the function $\phi_{C,2}$ is only a linear combination of $\phi_{F,2}$, $\phi_{F,4}$ and no more a hat function.

It is clear however that this process depends on the current iterate $\mathbf{y}^k_{j+1}$. For example, it can happen $(\mathbf{u}^k_{j+1})_i = (\mathbf{y}^k_{j+1})_i$ and $(\mathbf{u}^{k+1}_{j+1})_i \neq (\mathbf{y}^{k+1}_{j+1})_i$. The dependency of the basis functions at level j on the current iterate $\mathbf{y}^k_{j+1}$ implies that also the spaces $\mathbf{Y}_j$ depend on $\mathbf{y}^k_{j+1}$ as well. So we should write $\mathbf{Y}^k_j = \mathbf{Y}_j(\mathbf{y}^k_{j+1})$ for $j = 0, \dots, J-1$. Similarly, for the interpolation operator $\mathbf{\Pi}^{j+1}_j : \mathbf{Y}_j \to \mathbf{Y}_{j+1}$ introduced in section 6.7, i.e., $\mathbf{\Pi}^{k,j+1}_j = \mathbf{\Pi}^{j+1}_j(\mathbf{y}^k_{j+1})$. As a final consequence, the coarse matrix in (6.15a) has such dependency as well and we can write $\mathbf{A}^k_j = \mathbf{A}_j(\mathbf{y}^k_{j+1})$. The same arguments apply on $\mathbf{B}_j$, for $j = 0, \dots, J-1$. Therefore truncation makes necessary the Galerkin assembly everytime a dof is added or removed from the active set. If the set of active dofs on level $j+1$ does not change from iteration k to iteration $k+1$, then no re-assembly is needed.

A special case of always active dofs is related to boundary conditions, which are known from the beginning. So for linear problems, where the constraints are only given by the boundary conditions, known a priori, the coarse matrices can be assembled only once. In general, even for nested meshes, Galerkin assembly expressed in (6.15a) ensures a better description of the problem in contrast to a geometric assembly on $\mathcal{T}_j$. The reason for this, also for the linear case, is that in (6.15a) the information is projected from the finest level, where all the information of the problem is precisely described. So originally, before the projection, the shape functions belong to $\mathbf{X}_J$ and also possible space-dependent coefficients are represented on $\mathcal{T}_J$. On the other hand, a direct assembly on $\mathcal{T}_j$, for $j = 0, \ldots, J-1$, loses both of these advantages. Therefore we can say that the truncation strategy is, in general, useful. But it becomes even more important if monotone restrictions come into play as well.

6.9 Monotone restrictions

6.9.1 Introduction

In order to ensure the monotonicity in the reduction of the energy functional $\mathcal{J}_J$, it suffices to satisfy (6.16). Indeed a correction $\mathbf{c}_C \in \mathbf{Y}_C$ from the coarse level must satisfy:

$$\mathbf{l}_F(\mathbf{x}) - \mathbf{y}_F(\mathbf{x}) \le \Pi_C^F \mathbf{c}_C \le \mathbf{u}_F(\mathbf{x}) - \mathbf{y}_F(\mathbf{x}), \tag{6.27}$$

where by definition, if $\mathbf{y}_F$ is feasible, $\mathbf{l}_F(\mathbf{x}) - \mathbf{y}_F(\mathbf{x}) \le 0$ and $\mathbf{u}_F(\mathbf{x}) - \mathbf{y}_F(\mathbf{x}) \ge 0$. But then a comparison between functions at coarse and fine levels is required. In particular, as previously explained, this would make the constraints at a coarse level not box-constraints and the multigrid not-optimal anymore. For papers which explain this topic, see Kornhuber, Ralf [1994] and Kornhuber, Ralf [1996]. Optimality of the multigrid method could be recovered by defining $\mathbf{K}_j$ by means of box-constraints at a coarse level that fulfill (6.16) and that are also *close-enough* to the corresponding constraints at the fine level. To this purpose, monotone restriction operators are introduced. For coarse and fine meshes $\mathcal{T}_C$ and $\mathcal{T}_F$, defined in (6.21), and for coarse and fine spaces $\mathbf{X}_C$ and $\mathbf{X}_F$, defined in (6.22), monotone restriction operators on the lower and upper bounds are respectively operators $\mathbf{R}_{l,F}^C : \mathbf{Y}_F \to \mathbf{Y}_C$ and $\mathbf{R}_{u,F}^C : \mathbf{Y}_F \to \mathbf{Y}_C$ defined so that on the fine and coarse levels the relations (6.18) and (6.18) are respectively fulfilled. It is clear that, to this purpose, it is sufficient to choose $\mathbf{R}^C_{l,F}(\mathbf{l}_F - \mathbf{y}_F) = \mathbf{R}^C_{u,F}(\mathbf{u}_F - \mathbf{y}_F) = \mathbf{0}$. Nevertheless this choice is not smart because makes zero all the coarse corrections and the monotone multigrid boils down to a smoother on the fine level. For this reason, $\mathbf{R}^C_{l,F}(\mathbf{l}_F - \mathbf{y}_F)$ and $\mathbf{R}^C_{u,F}(\mathbf{u}_F - \mathbf{y}_F)$ have to be close-enough to $\mathbf{l}_F - \mathbf{y}_F$ and $\mathbf{u}_F - \mathbf{y}_F$. This property cannot be guaranteed in general, but it is for sure possible for functions which are piecewise constant or piecewise linear.

6.9.2 The 1D case

For simplicity, let us focus on a one dimensional monotone projection problem with coarse segment K_C, delimited by the points p_0 and p_{n_p}, and n_p nested fine elements $\{K_{F,1}, \ldots, K_{F,n_p}\}$, each of which is delimited by the points p_i and p_{i+1} for $i = 0, \ldots, n_p - 1$, where:

$$K_C = \bigcup_{\mathrm{i=1}}^{\mathrm{N}} K_{F,i} \qquad K_{F,i} \cap K_{F,j} = \emptyset \quad , i, j = 1, \ldots, n_p, \ i \neq j \, . \tag{6.28}$$

As example, see Figure 6.3.

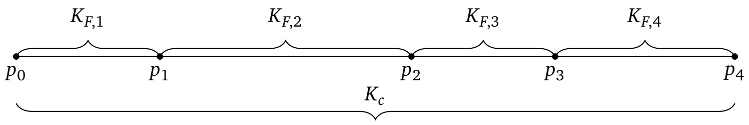


Figure 6.3. Coarse element $K_C = [p_0, p_4]$ subdivided into four fine elements $K_{F,1}, \ldots, K_{F,4}$.

Let us consider two piecewise constant functions on $\{K_{F,1}, \ldots, K_{F,n_p}\}$ that, with abuse of notation, assume respectively the values $l_{F,i}$ and $u_{F,i}$ on each $K_{F,i}$, for $i = 1, \ldots, n_p$, not to be confused with the degrees of freedom in (6.25). We define:

$$l_{F,i} = (\mathbf{l}_F)_i - (\mathbf{u}_F)_i \, , \tag{6.29a}$$

$$u_{F,i} = (\mathbf{u}_F)_i - (\mathbf{u}_F)_i \, . \tag{6.29b}$$

We want to define two constant functions on K_C, identified by their single value l_C and u_C, such that:

$$l_{F,i} \leq l_C \, , \quad u_C \leq u_{F,i} \qquad i = 1, \ldots, n_p \, . \tag{6.30}$$

It suffices to define the monotone restrictions so that:

$$l_C = \max_{i=1,\dots,n_p} \{l_{F,i}\}\,, \tag{6.31a}$$

$$u_C = \min_{i=1,\dots,n_p} \{u_{F,i}\}\,. \tag{6.31b}$$

This formula is equivalent to the ones in (6.20). Since this projection can be too pessimistic for piecewise linear functions, in Kornhuber, Ralf [1994] a better performing monotone restriction has been presented. Here we want to explain how it works also in the case of non-uniform refinement. For simplicity, we will consider only the projection of the upper bound. Its nodal values in $\{p_i\}_{i=0}^{n_p}$ are $\{u_i\}_{i=0}^{n_p}$. We use the same definition as in (6.29). The coarse nodal values $u_{C,0}$ and $u_{C,1}$, which refer to the nodes p_0 and p_{n_p}, can be computed by means of Algorithm 6. We initialize these values with the corresponding fine ones. We then loop on each fine element and consider its left and right node. We want to make the coarse bound, represented by the straight line between $u_{C,0}$ and $u_{C,1}$, below the given value $u_{F,i}$:

$$\alpha\, u_{C,0} + (1-\alpha) u_{C,1} \leq u_{F,i}\,, \quad \text{where } \alpha = \frac{|p_i - p_{n_p}|}{|p_{n_p} - p_0|}\,. \tag{6.32}$$

To this aim, the values $u_{C,0}$ and $u_{C,1}$ have to be modified accordingly:

$$\hat{u}_{C,k} = \min\left(u_{C,k}, \max\left(\frac{u_{F,i} - u_{C,j}}{\alpha} + u_{C,j}, u_{F,i} \right)\right)\,, \tag{6.33}$$

$$j = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{if } k = 1 \end{cases}\,, \qquad \alpha = \begin{cases} \dfrac{|p_i - p_{n_p}|}{|p_{n_p} - p_0|} & \text{if } k = 0 \\[2ex] \dfrac{|p_i - p_0|}{|p_{n_p} - p_0|} & \text{if } k = 1 \end{cases}\,, \tag{6.34}$$

where $\hat{u}_{C,0}$ and $\hat{u}_{C,1}$ are the updated values for $u_{C,0}$ and $u_{C,1}$. Given k and j, the update of $u_{C,0}$ and $u_{C,1}$ in Algorithm 6 is written, conceptually, as a parallel process, but the formula (6.34) permits also their sequential computation. We have stressed out this fact because (6.34) builds a monotone restriction independently of the order of the computation of $u_{C,0}$ and $u_{C,1}$.

Let us consider the Figure 6.4. The continuous black line represents the fine piecewise linear upper bound. The dashed orange horizontal line represents the constant value $u_{F,i}$. Then, on the left column, the dashed blu line passes through $u_{C,1}$ and $u_{F,i}$; the projected coarse upper bound $\hat{u}_{C,0}$ is highlighted in red. In the middle column, the dashed blu line passes from $u_{C,0}$ to $u_{F,i}$ and the projected

coarse upper bound $\hat{u}_{C,1}$ is depicted in red as well. Finally, in the right column, the two pieces of information are gathered into the resultant projected upper bound, which is always smaller than the upper bound on the finer level.
Thus in Figure 6.4, there are four cases to consider:

- The value $u_{F,i}$ is larger than $u_{C,0}$ and $u_{C,1}$i.e., $u_{F,i} > \max(u_{C,0}, u_{C,1})$; the line connecting $u_{C,0}$ and $u_{C,1}$ is below the upper bound.
- The value $u_{F,i}$ is smaller than $u_{C,0}$ and $u_{C,1}$, i.e., $u_{F,i} < \min(u_{C,0}, u_{C,1})$; the line connecting $u_{C,0}$ and $u_{C,1}$ is above the upper bound.
- The value $u_{F,i}$ is in between $u_{C,0}$ and $u_{C,1}$, in particular $u_{C,1} \leq u_{F,i} \leq u_{C,0}$; the slope of the line between $u_{C,0}$ and $u_{F,i}$ is larger, in absolute size, than the slope of the line between $u_{F,i}$ and $u_{C,1}$.
- The value $u_{F,i}$ is in between $u_{C,0}$ and $u_{C,1}$, in particular $u_{C,0} \leq u_{F,i} \leq u_{C,1}$; the slope of the line between $u_{C,0}$ and $u_{F,i}$ is smaller, in absolute value, than the slope of the line between $u_{F,i}$ and $u_{C,1}$.

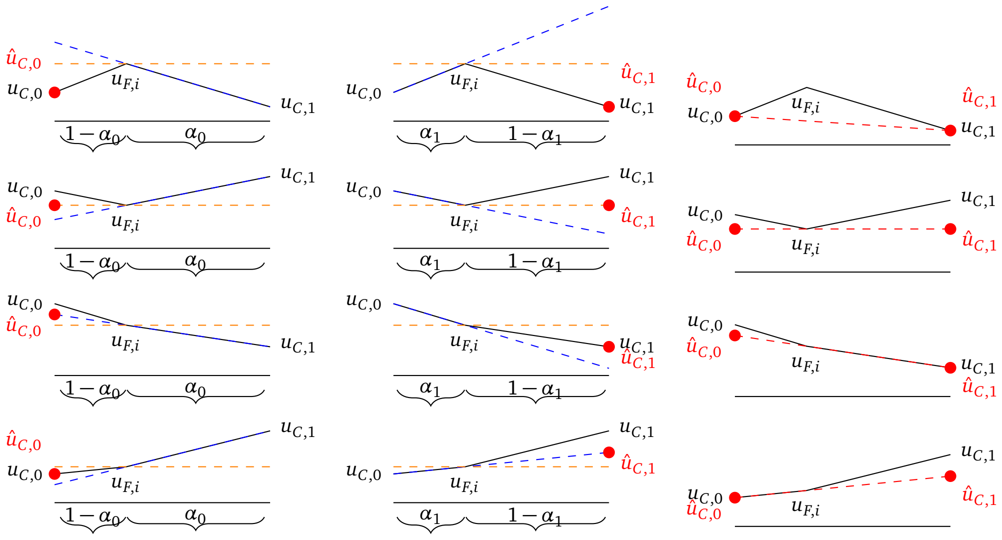


Figure 6.4. Continuous black line: the fine upper bound. Dashed orange line: the value $u_{F,i}$. On the left colum: computation of $\hat{u}_{C,0}$. In the middle colum: computation of $\hat{u}_{C,1}$. On the right column: the coarse upper bound represented as a red dashed line, always smaller than its fine continuous black version.

The formula in (6.34) can be modified if the truncation of the basis functions is used. Indeed, if the i-th dof is active due to the upper bound, i.e., $(\mathbf{y}_F)_i = (\mathbf{u}_F)_i$,

Algorithm 6: $u_{C,0}, u_{C,1}$=MonotoneRestricionUpperBound($p_0, \dots, p_{n_p}$, $u_{F,0}, \dots, l_{F,n_p}$)

```
$u_{C,0} = u_{F,0}$,
$u_{C,1} = u_{F,n_p}$
// loop on the elements
for $k = 1, \dots, n_p$ do
    // loop on the left and right node of the element $K_{F,k}$
    for $j = 1, 0$ do
        $i = k - j$
        $\alpha_0 \leftarrow \frac{|p_i - p_{n_p}|}{|p_{n_p} - p_0|}$
        $\alpha_1 \leftarrow \frac{|p_i - p_0|}{|p_{n_p} - p_0|}$
        $\hat{u}_{C,0} \leftarrow \min\left(u_{C,0}, \max\left(\frac{u_{F,i} - u_{C,1}}{\alpha_0} + u_{C,1}, u_{F,i}\right)\right)$
        $\hat{u}_{C,1} \leftarrow \min\left(u_{C,1}, \max\left(\frac{u_{F,k} - u_{C,0}}{\alpha_1} + u_{C,0}, u_{F,i}\right)\right)$
        $u_{C,0} \leftarrow \hat{u}_{C,0}$
        $u_{C,1} \leftarrow \hat{u}_{C,1}$
    end
end
```

then the value $(\mathbf{u}_F - \mathbf{y}_F)_i = 0$. The updated values of the coarse bounds would be consequently affected, making them too pessimistic. However the i-th dof should not influence the corrections at the coarse level anymore. Thus, we can redefine the values in (6.29) so that:

$$(\mathbf{l}_F)_i = (\mathbf{y}_F)_i \quad \Rightarrow \quad l_{F,i} = -\infty\,, \tag{6.35a}$$

$$(\mathbf{u}_F)_i = (\mathbf{y}_F)_i \quad \Rightarrow \quad u_{F,i} = +\infty\,. \tag{6.35b}$$

As an explanatory example, let us consider the positive piecewise linear function $(\mathbf{u}_F - \mathbf{y}_F)$ as in Figure 6.5 defined as:

$$\left[\mathbf{u}_F - \mathbf{y}_F\right](p) = \begin{cases} +1 & p = p_{F,1} = p_{C,1} \\ 0 & p = p_{F,2} \\ +1 & p = p_{F,3} = p_{C,2}\,, \end{cases} \tag{6.36}$$

By applying the standard definition of the monotone restriction $\mathbf{R}^C_{u,F}$ without truncation, we would recover:

$$[\mathbf{R}^C_{u,F}(\mathbf{u}_F - \mathbf{y}_F)](p) = 0 \qquad \text{for } p = p_{C,1}, p_{C,2} \,. \tag{6.37}$$

However, if the fine shape function corresponding to the node $p_{F,2}$ is removed, then no coarse correction can influence that point. And thus, before projection, (6.36) can be modified into:

$$\left[\mathbf{u}_F - \mathbf{y}_F\right](p) = \begin{cases} +1 & p = p_{F,1} \\ +\infty & p = p_{F,2} \\ +1 & p = p_{F,3} \,, \end{cases} \tag{6.38}$$

so that the coarse monotone restriction results in:

$$\left[\mathbf{R}^C_{u,F}(\mathbf{u}_F - \mathbf{y}_F)\right](p) = \begin{cases} +1 & p = p_{C,1} \\ +1 & p = p_{C,2} \,. \end{cases} \tag{6.39}$$

which guarantees for larger coarse corrections than the only zero function.

We have omitted the case of the lower bound. But it is very simple to take advantage of the same formula (6.34) for this case. It suffices to use $u_{F,i} = -l_{F,i}$ for $i = 1, \ldots, n_p$. Then we define $\hat{l}_{C,k} := -\hat{u}_{C,k}$.

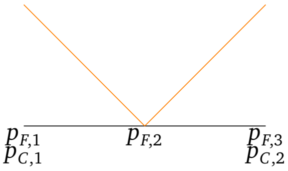


Figure 6.5. Example of representation of the piece-wise linear function $(\mathrm{u}_F - \mathrm{y}_F)$.

6.9.3 The 2D case

We would like to generalize the argument of the 1D case to triangle. The coarse points $p_{C,0}$, $p_{C,1}$ and $p_{C,2}$ coincide with the vertices of the triangle. The other $\{p_{F,i}\}$, for $i = 1, \ldots, n_p$, can be placed anywhere in the triangle. The fine elements satisfy the same relation (6.28). Then we loop on each fine element $K_{F,i}$ and on its internal nodes. For a given node, we update the values $u_{C,0}$, $u_{C,1}$, $u_{C,2}$. Let us focus, for example, on $u_{C,0}$. To recover the same argument used for the 1D case, we should trace a line from $p_{F,0}$, passing through $p_{F,i}$. In general, its prolongation does not intersect neither $p_{F,1}$ nor $p_{F,2}$. Thus, we need to consider the point of intersection between this line and the line connecting $p_{F,1}$ and $p_{F,2}$. We denote this point by $\hat{p}$. We denote by β the following normalized distance:

$$\beta = \frac{|p_{F,2} - \hat{p}|}{|p_{F,2} - p_{F,1}|}, \tag{6.40}$$

which we use to interpolate the value $u_{C,1}$ and $u_{C,2}$, obtaining the value:

$$\hat{u} = \beta u_{C,1} + (1 - \beta) u_{C,2} . \tag{6.41}$$

Now that $\hat{p}$ and $\hat{u}$ are known, the 1D argument can be replicated on the segment starting in $p_{F,0}$ and ending in $\hat{p}$. The procedure is repeated for all the other points.

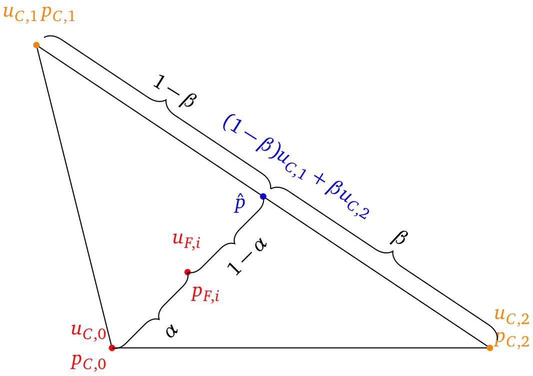


Figure 6.6. 2D projection example. Fixed the point $p_{F,i}$ and the value $u_{F,i}$, the line passing from $p_{F,0}$ to $p_{F,i}$ does not intersect neither $p_{F,1}$ nor $p_{F,2}$. Thus we interpolate the value $u_{C,1}$ and $u_{C,2}$ in the point $\hat{p}$, obtaining, $\hat{u} := \beta u_{C,1} + (1 - \beta) u_{C,2}$.

6.9.4 Monotone restrictions for the Signorini problem

It is clear that the monotone restrictions here defined can be applied only to piecewise constant or piecewise linear functions. Let us investigate if all the choices of the discrete spaces for the different formulations, (5.51), (5.54) and (5.58), fulfill this definition. To this aim, we first notice that (4.1a) and (4.1b) concern the trace of $\mathbf{u}_h$ and $\boldsymbol{\sigma}_h$.

In the primal formulation, $\mathbf{u}_h \in \mathbf{P}_1(\mathcal{T}_h)$ is piecewise linear on the domain and also on the boundary. In the dual formulation, $\boldsymbol{\sigma}_h \in \mathbf{RT}_1(\mathcal{T}_h)$ is a quadratic vector on the domain, but its trace is a piecewise linear vector. In the FOSLS formulation, $\mathbf{u}_h \in \mathbf{P}_1(\mathcal{T}_h)$ like for the primal formulation, while $\boldsymbol{\sigma}_h \in \mathbf{RT}_0(\mathcal{T}_h)$ is a piecewise linear vector on the domain and its trace is a piecewise constant vector. Therefore the monotone restrictions almost fit the framework of the discretized Signorini problem. We say "almost fit" because the monotone restrictions are defined for box-constraints, while (4.1a) and (4.1b) involve linear combinations of the corresponding dofs of $\mathbf{u}_h$ and $\boldsymbol{\sigma}_h$. We need to make those constraints box-constraints, by means of a proper transformation that we will discuss in the next section.

6.9.5 The Householder transformation

In general the conditions (4.1a) and (4.1b) are not box-constraints. Indeed all the components of the displacement $\mathbf{u}$ or the force $\boldsymbol{\sigma}\mathbf{n}$, for $i = 1, \dots, d$, are involved in a single inequality constraint. Therefore the set (6.5) can not really describe the formulations (5.50), (5.52), (5.57). However, by changing the local coordinate system from Cartesian to an orthogonal coordinate system with first component identified by the direction of the normal $\mathbf{n}$, (4.1a) and (4.1b) become box-constraints.

To this aim it is possible to define on Γ_C the Householder transformation $\mathbf{H}_\mathbf{x}$ relative to the outward normal $\mathbf{n}$ in the point $\mathbf{x} \in \Gamma_C$ as:

$$\mathbf{H}(\mathbf{x}) = \mathbf{I} - 2\,\mathbf{m}(\mathbf{x})\mathbf{m}(\mathbf{x})^T \tag{6.42a}$$

$$\mathbf{m}(\mathbf{x}) = \frac{\mathbf{n}(\mathbf{x}) - \mathbf{e}_1}{||\mathbf{n}(\mathbf{x}) - \mathbf{e}_1||}\,, \tag{6.42b}$$

where $\mathbf{e}_1 \in \mathbb{R}^d$ and $(\mathbf{e}_1)_i = \delta_{1i}$. Since the transformation (6.42) involves the normal $\mathbf{n}$, in the discrete setting this quantity must be computed with care. If the normal $\mathbf{n}$ is defined on a face, then it coincides with the normal of the face. This is the case for normals involved by $\mathbf{RT}_0(\mathcal{T}_h)$ and $\mathbf{RT}_1(\mathcal{T}_h)$ functions, like in (4.5b) and in (4.4b). On the other hand, a normal $\mathbf{n}$ on a vertex of the mesh is not well defined and so it is computed as the average of the normals defined on the surrounding faces. This can be the case for the displacement $\mathbf{u} \in \mathbf{P}_1(\mathcal{T}_h)$ in (4.5b) or in (4.3b).

The local Householder transformation has to be applied to the collection of dofs related to all the components of the local $\mathbf{u}_h$ or $\boldsymbol{\sigma}_h\mathbf{n}$. Given the point $\mathbf{p} \in \Gamma_{h,C}$ of normal $\mathbf{n}$, the vector of displacement dofs $\mathbf{u}_{h,\mathbf{p}} = (u_1, \dots, u_d)$ related to the vertex $\mathbf{p}$ can be expressed in the new coordinate system, obtaining $\overline{\mathbf{u}}_{h,\mathbf{p}}$:

$$\overline{\mathbf{u}}_{h,\mathbf{p}} := \mathbf{H}(\mathbf{p})\mathbf{u}_{h,\mathbf{p}}\,. \tag{6.43}$$

Its first component $(\overline{\mathbf{u}}_{h,\mathbf{p}})_1$ coincides with $\mathbf{u}_{h,\mathbf{p}} \cdot \mathbf{n}$. The same argument can be used also for $(\boldsymbol{\sigma}_h \cdot \mathbf{n})$, but we need to pay attention on the physical meaning of the quantities of interest. Indeed the set of dofs $\mathbf{u}_{h,\mathbf{p}}$ does actually represent the displacements in the Cartesian coordinate system, but the set of dofs $\boldsymbol{\sigma}_{h,\mathbf{p}}$ does not have the same meaning of the force $\underline{(\boldsymbol{\sigma}_h\mathbf{n})}$, which is what appears in (4.1b). Starting from $\boldsymbol{\sigma}_{h,\mathbf{p}}$, we need to recover $(\boldsymbol{\sigma}\mathbf{n})_{h,\mathbf{p}}$. Therefore, by using $\mathbf{H}(\mathbf{p})$, the new dofs will describe $(\boldsymbol{\sigma}\mathbf{n})_{h,\mathbf{p}}$ in the new coordinate system, not $\boldsymbol{\sigma}_{h,\mathbf{p}}$.

Once the local Householder transformations are built, the global Householder matrix can be built as well and the whole problem can be reformulated in terms of the new basis. Such a decision has a direct impact on the matrices $\mathbf{A}_j$, for $j = 0, \ldots, J$ and on the interpolation operators $\mathbf{\Pi}_j^{j+1}$, for $j = 0, \ldots, J-1$. From now on, we will assume the problem to be expressed using local Householder transformations, so that box-constraints are recovered.

6.10 Monotone multigrid for the primal formulation

In the previous sections we have introduced all the ingredients for the monotone multigrid method. We would like now to study its behavior in the case of the discrete primal formulation (5.50). The method applied to this discretization has already been investigated in Badea [2002], Badea [2014], Bader and Hoppe [1993], Badea and Krause [2012], Kornhuber, Ralf and Krause, Rolf [2001], Kornhuber, Ralf and Krause, Rolf and Sander, O. and Deuflhard, P. and Ertel, S. [2008], Krause [2009], Krause, Rolf and Rigazzi, Alessandro and Steiner, Johannes [2016]. In particular, we apply the Householder transformation of section 6.9.5, use truncation argument and monotone restrictions of sections 6.8 and 6.9. The smoother to be chosen is projected point-wise Gauss-Seidel. This means the spaces $\mathbf{Y}_{j,i}$, for $j = 0, \ldots, J$ and $i = 1, \ldots, n_j$, consists of all the dofs corresponding to a node. A subspace is thus referred to all components of a displacement in a node.

The MMG method for the primal formulation exhibits both theoretically and practically optimal convergence behavior. Nevertheless, as it can be shown in the experiments, its behavior deteriorates for increasing of the Lamé parameter λ. Indeed it is well known that the primal formulation is affected by the locking effects for $\lambda \rightarrow \infty$. We can notice this peculiarity in Figure (6.7), where we have applied MMG to the case of a square domain subject to a circular obstacle, as in Figure 8.22. The rate of convergence is independent of the dimension of the problem but is significantly influenced by λ. The difference becomes more and more evident by increasing λ and it is sufficient to choose $\lambda = 100$ to obtain a bad convergence rate. This is why we want to take advantage of the dual or the FOSLS formulations, which remain bounded even for $\lambda \rightarrow \infty$. However, special care is required since we need to solve also for the stress which belongs to $\mathbf{H}_{\mathrm{div}}(\Omega)$ and not to the standard Sobolev space $\mathbf{H}^1(\Omega)$.

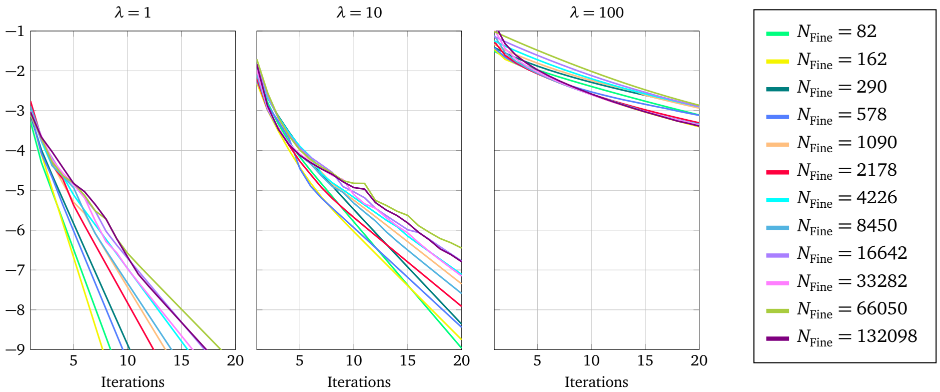


Figure 6.7. $\log_{10}$ of the Euclidean norm of the residual for the MMG applied to the primal formulation of the Signorini's problem. Parameters: $N_{\mathrm{coarse}} = 8$, $\mu = 1$, number of pre and post smoothing steps $= 3$.

6.11 Smoothers for H_{div} regular problems

The errors of the formulations (4.3), (4.4), (4.5), can be expressed as linear combinations of the eigenvectors of the corresponding differential operators and their high-frequency components are related to the large eigenvalues of the operators themselves. For H^1 regular problems (with the unknown belonging to H^1) like the weak form associated with the Laplacian problem, there is a bijective correspondence between large eigenvalues and high oscillatory eigenfunctions. This property makes Gauss-Seidel or the conjugate gradient methods, which only damp components related to large eigenvalues, also smoothers for H^1 regular problems.

Let us consider, for example, the following bilinear form $\Lambda_{H^1} : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$:

$$\Lambda_{H^1}(u, v) = (u, v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)} \quad \forall u, v \in H^1(\Omega)\,, \tag{6.44}$$

which also represents the H_1 dot product. The generalized eigenvalue problem can be written as: find the eigenfunctions $u \in H^1(\Omega)$ and the scalar eigenvalues $\lambda_{H^1} \geq 0$ such that:

$$\Lambda_{H^1}(u, u) = \lambda_{H^1}(u, u)_{L^2(\Omega)}\,. \tag{6.45}$$

Then the formula for λ_{H^1}, obtained by simply rearranging the terms, is:

$$\lambda_{H^1} = 1 + \frac{(\nabla u, \nabla u)_{L^2(\Omega)}}{(u,u)_{L^2(\Omega)}} = 1 + \frac{||\nabla u||^2_{L^2(\Omega)}}{||u||^2_{L^2(\Omega)}} \tag{6.46}$$

Since the second term is never negative, $\lambda_{H^1} \geq 1$. Then it is clear that low eigenvalues are related to functions with small variations, i.e., such that $||\nabla u||_{L^2(\Omega)}$ is very small. Indeed the smallest eigenvalue $\lambda_{H^1} = 1$ is related to the constant function $u = c$, for $c \in \mathbb{R}$, which defines the kernel of the gradient operator, $\text{Ker}(\nabla)$. Conversely, large eigenvalues correspond to eigenfunctions with large gradients. We know that Gauss-Seidel, for example, damp in few iterations all the eigenfunctions related to large eigenvalues. We need to determine if, in order to tackle *all oscillatory functions* on the mesh $\mathcal{T}_h$, it is sufficient to damp *only* the eigenfunctions related to large eigenvalues λ_{H^1}. Luckily, for a problem involving (6.44), this condition is sufficient and Gauss-Seidel and the conjugate gradient methods behave as smoothers.

However, for H_{div} regular problems (with the unknown belonging to H_{div}), there are low frequency components of the error which can be represented only on the fine mesh but which are not damped by standard smoothers for H^1 regular problems. For the sake of clarity, let us consider as an example the bilinear form $\Lambda_{H^{\text{div}}} : H_{\text{div}}(\Omega) \times H_{\text{div}}(\Omega)$ of the type :

$$\Lambda_{H_{\text{div}}}(\boldsymbol{\sigma}, \boldsymbol{\tau}) = (\boldsymbol{\sigma}, \boldsymbol{\tau})_{L^2(\Omega)} + (\text{div}\,\boldsymbol{\sigma}, \text{div}\,\boldsymbol{\tau})_{L^2(\Omega)} \quad \forall \boldsymbol{\sigma}, \boldsymbol{\tau} \in H_{\text{div}}(\Omega)\,. \tag{6.47}$$

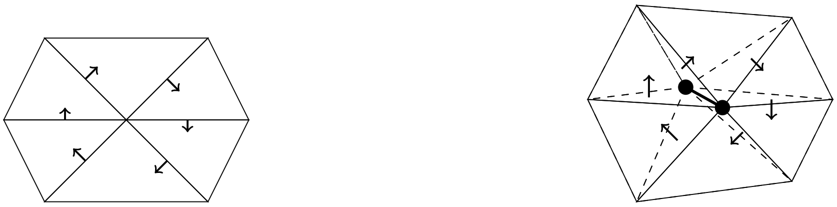
We can write the generalized eigenvalue problem as: find the eigenfunctions $\boldsymbol{\sigma} \in H_{\text{div}}(\Omega)$ and the scalar eigenvalues $\lambda_{\text{div}} > 0$ such that:

$$\Lambda_{H_{\text{div}}}(\boldsymbol{\sigma}, \boldsymbol{\sigma}) = \lambda_{\text{div}}(\boldsymbol{\sigma}, \boldsymbol{\sigma})_{L^2(\Omega)}\,. \tag{6.48}$$

The direct formula for λ_{div} is :

$$\lambda_{\text{div}} = 1 + \frac{(\text{div}\,\boldsymbol{\sigma}, \text{div}\,\boldsymbol{\sigma})_{L^2(\Omega)}}{(\boldsymbol{\sigma}, \boldsymbol{\sigma})_{L^2(\Omega)}} = 1 + \frac{||\text{div}\,\boldsymbol{\sigma}||^2_{L^2(\Omega)}}{||\boldsymbol{\sigma}||^2_{L^2(\Omega)}} \geq 1\,. \tag{6.49}$$

As we can see, the low eigenvalues are related to functions with small divergence, i.e., such that $||\text{div}\,\boldsymbol{\sigma}||$ is small. The set of all functions with zero-divergence corresponds to the eigenvalue $\lambda_{\text{div}} = 1$, identified by Ker(div), and is much larger than the one which contains only the constants, $\text{Ker}(\nabla)$. In particular, there are divergence-free oscillatory functions, corresponding to low eigenvalues of the operator $\Lambda_{H^{\text{div}}}$, which can be represented only on fine meshes. Therefore it loses the bijective relation, which characterizes symmetric positive definite operators like the one in (6.44), between eigenfunctions related to large eigenvalues and oscillatory functions. It is then clear that H_{div} smoothers must be able to damp also divergence-free components of the error which are not visible from coarser meshes. So it must tackle, in addition to the usual oscillatory functions related to large eigenvalues, also oscillatory divergence-free functions related to small eigenvalues. The divergence-free functions in 2D and in 3D for a simplicial mesh $\mathcal{T}_h$, that cannot be reproduced on coarser meshes, are depicted in Figure 6.8. They live on patches $\mathcal{P}$ of elements of $\mathcal{T}_h$ surrounding respectively a node, in 2D, and an edge, in 3D. These divergence-free functions are linear on each element and their normal component on the boundary of the patch is zero. In fact, for the divergence theorem $\int_{\mathcal{P}} \text{div}\boldsymbol{\sigma} = \int_{\partial\mathcal{P}} \boldsymbol{\sigma}\mathbf{n} = 0$, it follows they have, at least on average, zero-divergence.



(a) 2D node-related divergence-free function. (b) 3D edge-related divergence-free function.

Figure 6.8. Piecewise linear divergence-free functions.

In order to tackle these components of the error also on fine meshes, two main strategies have been proposed in the existing literature. The first one we are going to discuss has been developed by Hiptmair, Ralf [1997], Hiptmair, Ralf and Toselli, Andrea [2000] and it is based on the Helmholtz decomposition. The second one has been examined by Arnold-Falk-Winther and aims to directly capture the divergence-free components of the error. See Arnold Douglas [1998], Arnold et al. [2008], Arnold et al. [2000], Arnold et al. [1997]. However, these smoothers have been examined only for the linear case. *We aim to generalize them for the contact problems presented above.*

6.12 The generalized Hiptmair smoother to tackle inequality constraints

If a vector field $\mathbf{w} : \mathbb{R}^n \to \mathbb{R}^n$, for $n = 2, 3$, is sufficiently smooth, it can be decomposed in the sum of two components, one irrotational and the other solenoidal, by means of the Helmoltz decomposition. The irrotational field can be expressed as the gradient of a scalar potential ρ, whilst the solenoidal one can be expressed as the curl of a vector potential field $\boldsymbol{\psi}$, so that:

$$\mathbf{w} = \nabla \rho + \mathbf{curl}\, \boldsymbol{\psi} \,, \tag{6.50}$$

where $\mathbf{curl}\, \nabla \rho = 0$ and $\operatorname{div} \mathbf{curl}\, \boldsymbol{\psi} = 0$. In this way, we are able to distinguish between divergence free components ($\mathbf{curl}\, \boldsymbol{\psi}$) and the others ($\nabla \rho$). Hiptmair's idea is to smooth sequentially first the irrotational components of the error, then the solenoidal ones. In order to do so, the irrotational part can be damped by using a standard smoother for H^1 variables. As we already know, this would not work for the solenoidal term. This has to be examined in terms of its vector potential, meaning that we must recast the problem in the potential space. Therefore the bilinear form $\Lambda_{H^{\text{div}}}$ in (6.47) has to be recast into the space of the vector potential $\boldsymbol{\psi}$, obtaining $\Lambda_{H_{\text{curl}}} : H_{\text{curl}}(\Omega) \times H_{\text{curl}}(\Omega) \to \mathbb{R}$ defined as:

$$\Lambda_{H_{\text{curl}}}(\boldsymbol{\phi}, \boldsymbol{\psi}) := \Lambda_{H_{\text{div}}}(\mathbf{curl}\, \boldsymbol{\psi}, \mathbf{curl}\, \boldsymbol{\phi}) \qquad \forall \boldsymbol{\phi}, \boldsymbol{\psi} \in H_{\text{curl}}(\Omega) \,, \tag{6.51}$$

which results in:

$$\Lambda_{H_{\text{curl}}}(\boldsymbol{\phi}, \boldsymbol{\psi}) = (\mathbf{curl}\, \boldsymbol{\psi}, \mathbf{curl}\, \boldsymbol{\phi})_{L^2(\Omega)} \qquad \forall \boldsymbol{\phi}, \boldsymbol{\psi} \in H_{\text{curl}}(\Omega) \,. \tag{6.52}$$

As remarked in Hiptmair, Ralf [1997], "this time we do not have to worry about Ker(**curl**) because no zero order term is present in the potential space". Due to (6.50), we should also cast the bilinear form $\Lambda_{H^{\text{div}}}$ onto the space of the potential ρ, obtaining $\Lambda_{H_\Delta} : H^2(\Omega) \times H^2(\Omega) \to \mathbb{R}$ defined as:

$$\Lambda_{H_\Delta}(\rho, \eta) := \Lambda_{H_{\text{div}}}(\nabla \rho, \nabla \eta) \qquad \forall \rho, \eta \in H^2(\Omega) \,, \tag{6.53}$$

which results in:

$$\Lambda_{H_\Delta}(\rho, \eta) = (\nabla \rho, \nabla \eta)_{L^2(\Omega)} + (\Delta \rho, \Delta \eta)_{L^2(\Omega)} \qquad \forall \rho, \eta \in H^2(\Omega) \,, \tag{6.54}$$

where $\Delta := \operatorname{div} \nabla$ is the Laplace operator. However, such projection is not necessary, because, by smoothing oscillatory components in H^1, the components in H^2 are smoothed as well. Thus using smoothers like Gauss-Seidel or the conjugate gradients methods on the variable $\boldsymbol{\sigma} \in H_{\text{div}}(\Omega)$, in (6.47), is sufficient. After this, it is fundamental to compute corrections $\boldsymbol{\psi} \in H_{\text{curl}}(\Omega)$ by means of a smoother applied on the bilinear form (6.51). This mixed process is called *hybrid-smoother* and has been introduced in Hiptmair, Ralf [1997].

In the finite element framework, the spaces $H_{\text{div}}(\Omega)$ and $H_{\text{curl}}(\Omega)$ are discretized by means of $RT_p(\mathcal{T}_h)$ and $ND_{p+1}(\mathcal{T}_h)$, representing the Raviart-Thomas and Nédélec spaces of order p and $p+1$, with non-negative $p \in \mathbb{N}$. See sections 5.5.2 and 5.6 for details concerning their definition. Since the corrections in the potential space need to be added to the current iterate in $RT_p(\mathcal{T}_h)$, an interpolation operator between $ND_{p+1}(\mathcal{T}_h)$ and $RT_p(\mathcal{T}_h)$ is required. To this aim, the Stokes's theorem plays a preminent role with the following formula:

$$\int_{\Sigma} \mathbf{curl}\,\boldsymbol{\phi} \cdot \mathbf{n}\, d\sigma = \oint_{\partial\Sigma} \boldsymbol{\phi}\, d\mathbf{l}\,, \tag{6.55}$$

where Σ is a closed surface in $\mathbb{R}^d$ and $\partial\Sigma$ is its boundary. Assuming $\boldsymbol{\phi}_i = \mathbf{curl}\,\boldsymbol{\psi}$, the integral of the normal flux of $\boldsymbol{\phi}_i$ against the face F_i of a simplex K becomes:

$$\int_{F_i} \boldsymbol{\phi}_i \cdot \mathbf{n} = \sum_{\mathrm{e}\in\mathrm{F_i}} \int_e \boldsymbol{\psi} d\mathbf{l}_e\,, \tag{6.56}$$

where e is an edge of the face F_i. For simplicity, let us consider $d=2$ and the space $RT_0(\mathcal{T}_h)$. Then, recalling equation (2.14), $\boldsymbol{\phi}_i = \mathbf{curl}\,\psi$, with $\psi \in P_1(\mathcal{T}_h)$. We can relate the dofs of the two spaces as follows:

$$\int_{F_i} \boldsymbol{\phi}_i \cdot \mathbf{n} = \psi(p_{\text{end}}) - \psi(p_{\text{start}})\,, \tag{6.57}$$

where the potential continuous linear Lagrangian variable ψ is evaluated in the extreme points, p_{start} and p_{end}, of the oritented edge F_i. Thus the resulting interpolation between P_1 and RT_0 is characterized only by the values ±1. See also Kolev and Vassilevski [2012].

The aforementioned problems are linear, without constraints. Therefore, to understand the applicability of Hiptmair's smoother to the constrained case, let us consider, for example, the FOSLS formulation (4.5). The goal is to write the functional and the convex set in terms of the potential variable. By fixing $\mathbf{u}$ and $\boldsymbol{\sigma}$ and solving for corrections $\boldsymbol{\psi} \in \mathbf{H}^1(\Omega)$ in 2D (and $\boldsymbol{\psi} \in \mathbf{H}_{\mathbf{curl}}(\Omega)$ in 3D), the problem becomes:

$$\begin{aligned}
\mathcal{J}_f(\boldsymbol{\psi};\mathbf{u},\boldsymbol{\sigma}) =& \gamma||\mathcal{A}(\boldsymbol{\sigma}+\mathbf{curl}\,\boldsymbol{\psi}) - \boldsymbol{\varepsilon}(\mathbf{u})||^2_{L^2(\Omega)} + \delta||\mathrm{div}\boldsymbol{\sigma}+\mathbf{f}||^2_{L^2(\Omega)} \\
&+ \langle((\boldsymbol{\sigma}+\mathbf{curl}\,\boldsymbol{\psi})\mathbf{n})_n, u_n - g\rangle_{\Gamma_C}\,, && (6.58a)\\
\mathbf{K}_f =& \{\boldsymbol{\psi} \in \mathbf{H}^1(\Omega):\ \boldsymbol{\psi} = \mathbf{0} \ \text{ on } \Gamma_N, && (6.58b)\\
& (\mathbf{curl}\,\boldsymbol{\psi}\mathbf{n})_n \le -\sigma_n,\ \boldsymbol{\psi}_t = 0 \text{ on } \Gamma_C\}\,. && (6.58c)
\end{aligned}$$

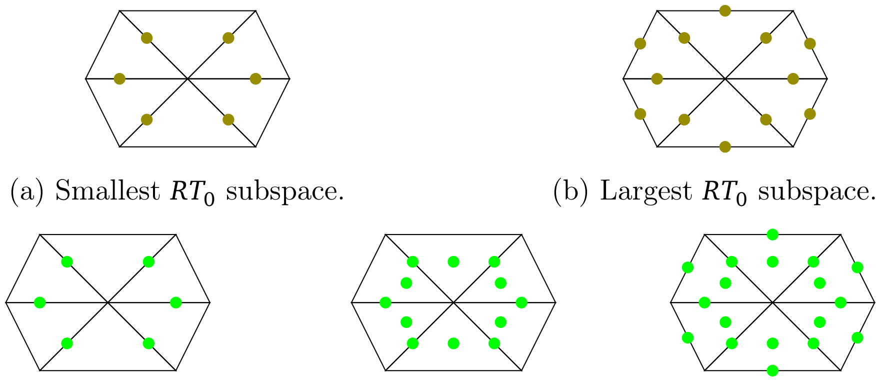
Without constraints, we recover a linear problem and the smoother reduces to the standard Hiptmair's smoother. Unfortunately, in the contact formulation, the correction of the potential variable $\boldsymbol{\psi}$ must fulfill global constraints, even though the original constraints are only local. One could proceed in three different ways:

- enforcing $\boldsymbol{\psi} = \mathbf{0}$ even on Γ_C, but then there would be no correction on the contact boundary;
- transforming, the global constraints into a more restricted dof-wise constraint, in the discrete FE formulation;
- satisfying the global constraints, making each dof satisfying contemporarily more than one inequality constraint.

As it will be explained, even though some numerical experiments have been carried out with these variants, they are still not able to solve the problem (7.2). A more practical approach is, on the other hand, represented by the Arnold-Falk-Winther's smoother

6.13 The generalized Arnold-Falk-Winther smoother

The Arnold-Falk-Winther's smoother tackles directly divergence-free components not visible from coarser meshes. See Arnold et al. [1997], Arnold et al. [2000], Arnold Douglas [1998]. To this aim, in contrast to projected point-wise Gauss-Seidel method, it does not act sequentially on one-dimensional subspaces, but on larger subspaces. In particular, the subspaces are defined on patches. A patch $\mathcal{P}_n$ related to a node n is the set of all elements which share that node. This kind of patch can be used for any dimension d. In 3D, we can also consider a patch $\mathcal{P}_{n,m}$ which is the set of all elements sharing the edge connecting the nodes m and n. A representation of these patches and of piecewise linear divergence-free components is given in Figure 6.8. For RT_p, the subspace has to be defined, at least, as the set of all the degrees of freedom on the internal faces of the patch. But, on such a patch, we can also consider other degrees of freedom, like the ones on the border of the patch. We can see a 2D example in Figure 6.9 for both RT_0 and RT_1 spaces.



(a) Smallest RT_0 subspace.

(b) Largest RT_0 subspace.

(c) Smallest RT_1 subspace.

(d) All internal RT_1 dofs.

(e) All RT_1 dofs.

Figure 6.9. The olive and green circles refer respectively to RT_0 and RT_1 dofs. For simplicity, in the RT_1 case, we count 2 dofs per each green circle, since we do have 2 dofs per face and 2 internal dofs. The Figure (a) represents the smallest patch-subspace for tackling divergence-free components of RT_0 functions, while Figure (b) is an extension to the border. The Figure (c) represents the smallest patch-subspace for tackling divergence-free components of RT_1 functions. The Figure (d) represents a possible extension of such subspace to all internal dofs. In Figure (e) all RT_1 dofs of the patch are represented.

6.14 Uncertainty Quantification

6.14.1 Introduction

The MMG method described in section 6.6 is designed to have the optimal convergence property. Therefore the number of iterations for its convergence should be independent of the dimension of the problem and of the number of levels used. This property is very important because it ensures a fast convergence of the algorithm for a constrained optimization problem. If many are the constrained optimization problems to be solved, then optimal convergence is crucial: fast convergence of a single simulation translates into the fast convergence of the overall simulations. This is the case of uncertainty quantification problems, which are solved by means of quasi-Monte Carlo or multilevel Monte Carlo methods that require the solution of thousands of simulations. In this context, the MMG method of section 6.6 for the dual formulation applied to the Signorini problem would be fundamental. In sections 6.14.2 and 6.14.3 we respectively introduce the uncertainty quantification problems and the methods for their resolution.

6.14.2 The uncertainty quantification problems

Continuum mechanics describes the mechanical problem from a macroscopic point of view. This approach is based on a simplified treatment of the surface and material coefficients of the body, in addition to the external sources, which are supposed to be known exactly. However, in practice, we do not know the real shape of the body and how its coefficients vary from point to point. To do that, we would need to deal with the microscale for every portion of the body itself, which in practice is not achievable. Since the microscale perspective does not pay off for standard engineering problems, but we still would want to take into account our lack of complete knowledge of the boundary, the coefficients and the external sources, an option is given by their stochastic treatment. With *uncertainty quantification methods*, all the quantities, which we cannot assume to know, are now considered stochastic variables and the knowledge we have of them is assumed to be their mean values. For further references regarding the stochastic treatment of some features of PDEs problems, we can mention Babuška [1961], Babuška [1971], Babuška et al. [2005], Babuška et al. [2007a], Babuška et al. [2007b], Schwab, Christoph [2003], Schwab, Ch and Stuart, AM [2011], Schwab, Ch and Todor, Radu Alexandru [2003], Gittelson, Claude Jeffrey [2011a], Gittelson, Claude Jeffrey [2011b], Multerer, MD [2019].

For simplicity, we study now a classic forward problem of uncertainty quantification for a PDE. The uncertainty resides only in the coefficient a of the constitutive law. Since the coefficient a is stochastic, also the unknown u, which depends on it, has to be stochastic as well. We characterize this fact by adding the dependency of a and u on the parameter $\theta \in \Theta$.

$$\begin{aligned} -\mathrm{div}\,(a(\mathbf{x},\theta)\nabla u(\mathbf{x},\theta)) &= f(\mathbf{x}) \qquad \mathbf{x}\in\Omega\,, && (6.59)\\ u(\mathbf{x},\theta) &= 0 \qquad \mathbf{x}\in\partial\Omega\,. && (6.60)\end{aligned}$$

We can separate the dependencies from the random field θ and the spatial dimension $\mathbf{x}$, by means of the Karhunen-Loève expansion of the random field $a(\mathbf{x},\theta)$. See Harbrecht, Helmut and Peters, Michael D. and Schmidlin, Marc [2017]. We can introduce a sequence of uniformly distributed and independent variables $y_j \in U([-0.5, 0.5])$ and express the coefficient a as:

$$a(\mathbf{x},\theta) = a_0(\mathbf{x}) + \sum_{j=1}^{\infty} y_j(\theta)a_j(\mathbf{x}), \tag{6.61}$$

where

$$\begin{aligned} a_0(\mathbf{x}) &= \int_\Theta a(\mathbf{x},\theta)d\theta, && j=0, && (6.62)\\ y_j(\theta) &= \int_\Omega (a(\mathbf{x},\mathbf{y}) - a_0(\mathbf{x}))\,a_j(\mathbf{x})d\mathbf{x}, && j\geq 1. && (6.63)\end{aligned}$$

The term $a_0(\mathbf{x})$ is the mean-value of $a(\mathbf{x},\theta)$ which we would use if we would consider the problem as full deterministic. The terms $a_j(\mathbf{x})$ are usually the eigenvectors of a Hilbert-Schmidt operator, induced from the kernel of a covariance operator. For non-trivial geometries, the computation of the eigenpairs of this operator is not an easy task. For strategies in this direction, we can mention Pezzuto, Simone and Quaglino, Alessio and Potse, Mark [2019].

In order to reduce the infinite dimensional stochastic problem to a finite one, the series has to be trucated up to the order N, so that we obtain:

$$a(\mathbf{x},\theta) \approx a_0(\mathbf{x}) + \sum_{j=1}^{N} y_j(\theta)a_j(\mathbf{x}), \tag{6.64}$$

Given the truncated distribution of $a(\mathbf{x},\theta)$, we would like to obtain information about the distribution of the solution $u = u(\mathbf{x},\theta)$, like the mean-value and the variance. To this aim, we must produce a sufficient amount of experimental data. In particular, we must somehow fix a $\bar{\theta}$ to which correspond

$y_1(\bar{\theta}), \ldots y_N(\bar{\theta})$, i.e., N random numbers in $[-0.5, 0.5]$. Then the PDE problem can be solved and we obtain $u = u(\mathbf{x}, \bar{\theta})$. We can repeat the process for varying θ_k, for $k = 1, \ldots, M$. Then, once we have computed a sufficiently large number of experiments, we can recover the mean-value and the variance of $u(\mathbf{x}, \theta)$:

$$\mathrm{E}[u(\mathbf{x})] = \frac{1}{M}\sum_{k=1}^{M} u(\mathbf{x}, \theta_k) \tag{6.65}$$

$$\mathrm{Var}[u(\mathbf{x})] = \frac{1}{M}\sum_{k=1}^{M} [\mathrm{E}[u(\mathbf{x})] - u(\mathbf{x}, \theta_k)]^2 \tag{6.66}$$

To reduce the overall computational cost, we can minimize the computational time of a single simulation by means of an optimal solver, like the MMG that we have presented. But we can also find smart strategies for reducing the number of simulations required. We discuss this point in the next paragraph.

6.14.3 Multilevel Monte Carlo

The first natural approach for generating each θ_k is given by the Monte Carlo method. We just need to produce, every time, N random numbers in the range $[-0.5, 0.5]$. This method converges with an error of order $O(N^{-0.5})$, meaning that we need to perform many numerical experiments to obtain convergence. It is clear that, if solving a PDE is, in general, a demanding task, solving N PDEs can be computationally overwhelming. The idea of quasi-Monte Carlo methods is to accelerate such convergence with an error of the order $O(N^{-1})$, so that a reduced number of experiments is needed for getting the same level of error obtained with the Monte Carlo method. For further reading about this topic, see Trefethen, Lloyd N. [2017], Gilbert, Alexander D. and Graham, Ivan G. and Kuo, Frances Y. and Scheichl, Robert and Sloan, Ian H. [2019]. In theory, this result has not been achieved and instead the order of the error is given by $O(\log(N)/N)$. However, in practice, the quasi-Monte Carlo method is much faster than its theoretical upper bound. Indeed, in contrast to Monte Carlo, it is purely deterministic. Instead of creating a sequence of random numbers, a low discrepancy sequence is built. Intuitively this means that, on average, the sequence of numbers is equidistributed.

Even though quasi-Monte Carlo can solve a smaller amount of PDEs, every experiment still consists of solving a large PDE. One can reduce the *computational burdern* by means of a coarsening strategy. Instead of solving the same PDE on the same mesh, we do few experiments on the fine mesh and we do many on coarser meshes. This is the strategy of the so-called multi-level Monte Carlo. See

Giles, Michael B. [2015]. As in multigrid methods, the convergence is ensured by the usage of the experiments on the fine mesh, while the coarser ones are needed for accelerating convergence. Nevertheless, such a strategy becomes impossible if the fine mesh cannot be coarsened. A further generalization is given by the multi-fidelity Monte Carlo methods. In this case, also low-fidelity models, which do not provide a good approximation in terms of the error bounds, but still exhibit high correlation with the finest model, can be used to accelerate convergence. See Quaglino, A. and Pezzuto, S. and Krause, R. [2018].

As mentioned at the beginning of this section, for the solution of both the finest and the coarsest models of a multi-level Monte Carlo or a multi-fidelity Monte Carlo method applied to the dual formulation of the Signorini problem, the MMG strategy presented in this thesis could be used. As it is clear, there are two levels of multigrid that would be exploited. An external multigrid process in the stochastic space (multi-level Monte Carlo) and an internal one for the deterministic minimization problem (our proposed algorithm). Of course, if the algorithms for the internal problems have the optimal convergence property, the external process is beneficially affected and the overall performance is improved.

Chapter 7

The FOSLS case

In section 7.1 of this chapter we present a proof for the ellipticity of the bilinear form related to the functional (4.5a). Since in the FOSLS formulation we seek for both variables, $\mathbf{u}$, and $\boldsymbol{\sigma}$, and not just for one of the two like in the primal or in the dual formulations, the ellipticity will be granted under certain conditions of the coefficients γ and δ. In section 7.2, we introduce the generalized Arnold-Falk-Winther smoother for the FOSLS formulation. In particular, we opt for a monolithic approach like in Starke [1999], where the subspaces are meant to capture local corrections for both $\mathbf{u}$ and $\boldsymbol{\sigma}$. Thus we will expand the subspace in Figure 6.9a so that also the displacement related to the internal node is considered. Even though the definition of the smoother is straightforward, in section 7.3 we will examine how the standard truncation of both the primal and dual variables does not accelerate the convergence as expected.

7.1 Ellipticity for the FOSLS Signorini formulation

Let us consider the functional (4.5a). For the sake of simplicity, we can assume that the unknowns satisfy homogeous boundary conditions. Indeed, we can write $\mathbf{u} = \mathbf{u}_0 + \mathbf{v}_{\mathbf{g}_D}$ and $\boldsymbol{\sigma} = \boldsymbol{\sigma}_0 + \boldsymbol{\tau}_{\mathbf{g}_N}$, where $\mathbf{u}_0 \in \mathbf{U}_0$, $\boldsymbol{\sigma}_0 \in \boldsymbol{\Sigma}_0$ and $\mathbf{v}_{\mathbf{g}_D} \in \mathbf{U}_{\mathbf{g}_D}$, $\boldsymbol{\tau}_{\mathbf{g}_N} \in \boldsymbol{\Sigma}_t$ are supposed to be known and to satisfy:

$$\mathbf{v}_{\mathbf{g}_D} \in \mathbf{U}_{\mathbf{g}_D} \qquad \mathbf{v}_{\mathbf{g}_D}|_{\Gamma_C} \cdot \mathbf{n} = 0\,, \tag{7.1a}$$

$$\boldsymbol{\tau}_{\mathbf{g}_N} \in \boldsymbol{\Sigma}_t \qquad \boldsymbol{\tau}_{\mathbf{g}_N}|_{\Gamma_C} \quad = \mathbf{0}\,. \tag{7.1b}$$

In this way, the functional in (4.5a) becomes:

$$\begin{aligned}\mathcal{J}_f(\mathbf{u}_0,\boldsymbol{\sigma}_0) = &+\gamma||\mathcal{A}\boldsymbol{\sigma}_0-\boldsymbol{\varepsilon}(\mathbf{u}_0)||^2_{L^2(\Omega)}+\gamma||\mathcal{A}\boldsymbol{\tau}_{\mathbf{g}_N}-\boldsymbol{\varepsilon}(\mathbf{v}_{\mathbf{g}_D})||^2_{L^2(\Omega)}\\ &+2\gamma(\mathcal{A}\boldsymbol{\tau}_{\mathbf{g}_N}-\boldsymbol{\varepsilon}(\mathbf{v}_{\mathbf{g}_D}),\mathcal{A}\boldsymbol{\sigma}_0-\boldsymbol{\varepsilon}(\mathbf{u}_0))_{L^2(\Omega)}\\ &+\delta||\mathrm{div}\boldsymbol{\sigma}_0||^2_{L^2(\Omega)}+\delta||\mathrm{div}\boldsymbol{\tau}_{\mathbf{g}_N}+\mathbf{f}||^2_{L^2(\Omega)}\\ &+2\delta(\mathrm{div}\boldsymbol{\sigma}_0,\mathrm{div}\boldsymbol{\tau}_{\mathbf{g}_N}+\mathbf{f})^2_{L^2(\Omega)}\\ &+\langle\sigma_{0,n},u_{0,n}\rangle_{\Gamma_C}-\langle\sigma_{0,n},g\rangle_{\Gamma_C}\,.\end{aligned}\tag{7.2}$$

From this formulation, we can remove the constant terms, so that the functional and the convex set are:

$$\begin{aligned}\mathcal{J}_f(\mathbf{u}_0,\boldsymbol{\sigma}_0) = &+\gamma||\mathcal{A}\boldsymbol{\sigma}_0-\boldsymbol{\varepsilon}(\mathbf{u}_0)||^2_{L^2(\Omega)}+2\gamma(\mathcal{A}\boldsymbol{\tau}_{\mathbf{g}_N}-\boldsymbol{\varepsilon}(\mathbf{v}_{\mathbf{g}_D}),\mathcal{A}\boldsymbol{\sigma}_0-\boldsymbol{\varepsilon}(\mathbf{u}_0))_{L^2(\Omega)}\\ &+\delta||\mathrm{div}\boldsymbol{\sigma}_0||^2_{L^2(\Omega)}+2\delta(\mathrm{div}\boldsymbol{\sigma}_0,\mathrm{div}\boldsymbol{\tau}_{\mathbf{g}_N}+\mathbf{f})^2_{L^2(\Omega)}\\ &+\langle\sigma_{0,n},u_{0,n}\rangle_{\Gamma_C}-\langle\sigma_{0,n},g\rangle_{\Gamma_C}\,,\end{aligned}\tag{7.3a}$$

$$\mathbf{K}_f=\{(\mathbf{u}_0,\boldsymbol{\sigma}_0)\in\mathbf{U}_0\times\boldsymbol{\Sigma}_0 : u_{0,n}\leq g,\ \sigma_{0,n}\leq 0,\ (\boldsymbol{\sigma}_0\mathbf{n})_t=\mathbf{0}\text{ on }\Gamma_C\}\,.\tag{7.3b}$$

All the corresponding bilinear and linear forms are continuous. However it is necessary to show the ellipticity of the bilinear forms with respect to the following norm:

$$||(\mathbf{u},\boldsymbol{\sigma})|| := \sqrt{||\boldsymbol{\sigma}||^2_{\boldsymbol{\Sigma}}+||\boldsymbol{\varepsilon}(\mathbf{u})||^2_{\mathbf{L}^2(\Omega)}}\,,\tag{7.4}$$

as done in Cai, Zhiqiang and Starke, Gerhard [2004]. We notice that the bilinear forms in (7.3a) are the same as the ones in (4.5a). Thus a proof regarding the bilinear forms holds anyway for homogeneous or non homogenous boundary conditions.

Theorem 7.1.1. *For any $(\boldsymbol{u},\boldsymbol{\sigma})\in\boldsymbol{K}_f$, there exists $C>0$ such that the inequality*

$$||\boldsymbol{\sigma}||^2_{\boldsymbol{\Sigma}}+||\boldsymbol{\varepsilon}(\boldsymbol{u})||^2_{L^2(\Omega)}\leq C\left(\gamma||\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\boldsymbol{u})||^2_{L^2(\Omega)}+\delta||div\,\boldsymbol{\sigma}||^2_{L^2(\Omega)}+\langle\sigma_n,u_n\rangle_{\Gamma_C}\right)\tag{7.5}$$

is fulfilled for

$$\begin{aligned}&\gamma\geq\left[\left(\frac{2\mu^2+1}{\mu}\right)\left(5\mu+16\mu C_k^2\right)+2\right]\\ &\delta\geq\left[\frac{4C_k^2}{\mu}+1\right]\\ &C\geq 2\,.\end{aligned}\tag{7.6}$$

where C_k is the Korn's constant of Theorem 2.4.8.

Proof. The proof is similar to the Theorem 3.1 in Cai, Zhiqiang and Starke, Gerhard [2004]. We will omit the $L^2(\Omega)$ subscript when it is obvious from the context. We also assume that $\boldsymbol{\sigma}$ and $\mathbf{u}$ satisfy homogeneous boundary conditions.

We start by add and subtract $\mathcal{A}\boldsymbol{\sigma}$ into the last norm of (7.4) and use triangle and Young's inequalities:

$$\|\boldsymbol{\sigma}\|^2 + \|\operatorname{div}\boldsymbol{\sigma}\|^2 + \|\boldsymbol{\varepsilon}(\mathbf{u})\|^2 \leq \|\boldsymbol{\sigma}\|^2 + \|\operatorname{div}\boldsymbol{\sigma}\|^2 + 2\|\boldsymbol{\varepsilon}(\mathbf{u}) - \mathcal{A}\boldsymbol{\sigma}\|^2 + 2\|\mathcal{A}\boldsymbol{\sigma}\|^2 \tag{7.7a}$$

$$\leq \left(\frac{2\mu^2+1}{2\mu^2}\right)\|\boldsymbol{\sigma}\|^2 + \|\operatorname{div}\boldsymbol{\sigma}\|^2 + 2\|\boldsymbol{\varepsilon}(\mathbf{u}) - \mathcal{A}\boldsymbol{\sigma}\|^2 \tag{7.7b}$$

$$\leq \left(\frac{2\mu^2+1}{\mu}\right)(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma}) + \|\operatorname{div}\boldsymbol{\sigma}\|^2 + 2\|\boldsymbol{\varepsilon}(\mathbf{u}) - \mathcal{A}\boldsymbol{\sigma}\|^2, \tag{7.7c}$$

where we use:

$$\|\mathcal{A}\boldsymbol{\tau}\| \leq \frac{1}{2\mu}\|\boldsymbol{\tau}\|, \tag{7.8}$$

to get (7.7b) and :

$$(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma}) \geq \frac{1}{2\mu}\|\boldsymbol{\sigma}\|^2 \tag{7.9}$$

to obtain (7.7c). Thus it is now sufficient to bound $(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})$. We add and subtract $\boldsymbol{\varepsilon}(\mathbf{u})$, exploit the Green's formula (2.19) and the Cauchy-Schwarz inequality:

$$(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma}) = (\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u}),\boldsymbol{\sigma}) + (\boldsymbol{\varepsilon}(\mathbf{u}),\boldsymbol{\sigma}) \tag{7.10a}$$

$$= (\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u}),\boldsymbol{\sigma}) + (\boldsymbol{\sigma} - \frac{1}{2}\left(\boldsymbol{\sigma} - \boldsymbol{\sigma}^T\right), \nabla\mathbf{u}) \tag{7.10b}$$

$$= (\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u}),\boldsymbol{\sigma}) - (\operatorname{div}\boldsymbol{\sigma},\mathbf{u}) - \frac{1}{2}\left(\boldsymbol{\sigma} - \boldsymbol{\sigma}^T, \nabla\mathbf{u}\right) + \int_{\partial\Omega} \boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{u} \tag{7.10c}$$

$$\leq \|\mathcal{A}\boldsymbol{\sigma} - \boldsymbol{\varepsilon}(\mathbf{u})\|\|\boldsymbol{\sigma}\| + \|\operatorname{div}\boldsymbol{\sigma}\|\|\mathbf{u}\| + \frac{1}{2}\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^T\|\|\nabla\mathbf{u}\| + \int_{\partial\Omega} \boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{u}. \tag{7.10d}$$

In order to bound the second and third addenda, we take advantage of the fact that the constitutive law bounds the symmetry condition, i.e., (3.35), and of the

Korn's inequality (2.35):

$$\begin{aligned}||\mathrm{div}\boldsymbol{\sigma}||||\mathbf{u}|| + \frac{1}{2}||\boldsymbol{\sigma}-\boldsymbol{\sigma}^T||||\nabla\mathbf{u}|| &\leq ||\mathbf{u}||_{\mathbf{H}^1(\Omega)}\left(||\mathrm{div}\boldsymbol{\sigma}|| + \frac{1}{2}||\boldsymbol{\sigma}-\boldsymbol{\sigma}^T||\right) \qquad (7.11a)\\ &\leq C_k||\boldsymbol{\varepsilon}(\mathbf{u})||\left(||\mathrm{div}\boldsymbol{\sigma}|| + 2\mu||\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})||\right), \qquad (7.11b)\end{aligned}$$

where C_k is the Korn's constant. In this way, we can define:

$$H(\boldsymbol{\sigma},\mathbf{u}) := ||\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})||||\boldsymbol{\sigma}|| + C_k||\boldsymbol{\varepsilon}(\mathbf{u})||||\mathrm{div}\boldsymbol{\sigma}|| + 2\mu C_k||\boldsymbol{\varepsilon}(\mathbf{u})||||\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})|| \qquad (7.12)$$

and obtain:

$$(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma}) \leq H(\boldsymbol{\sigma},\mathbf{u}) + \int_{\partial\Omega}\boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{u}\,. \qquad (7.13)$$

We apply three times the Young's inequality to (7.12), with parameters $\tilde{\alpha},\ \tilde{\beta},\ \tilde{\gamma} > 0$, collect the common terms and use (7.9):

$$\begin{aligned}H(\boldsymbol{\sigma},\mathbf{u}) \leq &\left(\frac{1}{4\tilde{\alpha}} + \frac{\mu C_k}{2\tilde{\gamma}}\right)||\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})||^2 + \tilde{\alpha}||\boldsymbol{\sigma}||^2 + \\ &+\left(C_k\tilde{\beta} + 2\mu C_k\tilde{\gamma}\right)||\boldsymbol{\varepsilon}(\mathbf{u})||^2 + \frac{C_k}{4\tilde{\beta}}||\mathrm{div}\boldsymbol{\sigma}||^2 \qquad (7.14a)\\ \leq &\left(\frac{1}{4\tilde{\alpha}} + \frac{\mu C_k}{2\tilde{\gamma}}\right)||\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})||^2 + 2\mu\tilde{\alpha}(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma}) + \\ &+\left(C_k\tilde{\beta} + 2\mu C_k\tilde{\gamma}\right)||\boldsymbol{\varepsilon}(\mathbf{u})||^2 + \frac{C_k}{4\tilde{\beta}}||\mathrm{div}\boldsymbol{\sigma}||^2\,. \qquad (7.14b)\end{aligned}$$

Now we can repeat the same argument of (7.7a), (7.7b), (7.7c)

$$
\begin{aligned}
H(\boldsymbol{\sigma},\mathbf{u}) \leq & \left(\frac{1}{4\tilde{\alpha}}+\frac{\mu C_k}{2\tilde{\gamma}}+2\left(C_k\tilde{\beta}+2\mu C_k\tilde{\gamma}\right)\right)\|\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})\|^2+2\mu\tilde{\alpha}(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})+ \\
& +2\left(C_k\tilde{\beta}+2\mu C_k\tilde{\gamma}\right)\|\mathcal{A}\boldsymbol{\sigma}\|^2+\frac{C_k}{4\tilde{\beta}}\|\mathrm{div}\boldsymbol{\sigma}\|^2 \qquad (7.15a) \\
\leq & \left(\frac{1}{4\tilde{\alpha}}+\frac{\mu C_k}{2\tilde{\gamma}}+2C_k\tilde{\beta}+4\mu C_k\tilde{\gamma}\right)\|\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})\|^2+2\mu\tilde{\alpha}(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})+ \\
& +\left(\frac{C_k\tilde{\beta}}{2\mu^2}+\frac{C_k\tilde{\gamma}}{\mu}\right)\|\boldsymbol{\sigma}\|^2+\frac{C_k}{4\tilde{\beta}}\|\mathrm{div}\boldsymbol{\sigma}\|^2 \qquad (7.15b) \\
\leq & \left(\frac{1}{4\tilde{\alpha}}+\frac{\mu C_k}{2\tilde{\gamma}}+2C_k\tilde{\beta}+4\mu C_k\tilde{\gamma}\right)\|\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})\|^2+ \\
& +\left(2\mu\tilde{\alpha}+\frac{C_k\tilde{\beta}}{\mu}+2C_k\tilde{\gamma}\right)(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})+\frac{C_k}{4\tilde{\beta}}\|\mathrm{div}\boldsymbol{\sigma}\|^2\,. \qquad (7.15c)
\end{aligned}
$$

Now by letting $\tilde{\alpha}=\frac{1}{8\mu}$, $\tilde{\beta}=\frac{\mu}{8C_k}$, $\tilde{\gamma}=\frac{1}{16C_k}$, it follows that:

$$\left(2\mu\tilde{\alpha}+\frac{C_k\tilde{\beta}}{\mu}+2C_k\tilde{\gamma}\right)=\frac{1}{2}\,, \qquad (7.16)$$

$$\left(\frac{1}{4\tilde{\alpha}}+\frac{\mu C_k}{2\tilde{\gamma}}+2C_k\tilde{\beta}+4\mu C_k\tilde{\gamma}\right)=\frac{5}{2}\mu+8\mu C_k^2\,, \qquad (7.17)$$

$$\frac{C_k}{4\tilde{\beta}}=\frac{2C_k^2}{\mu}. \qquad (7.18)$$

Thus:

$$\frac{1}{2}(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})\leq\left(\frac{5}{2}\mu+8\mu C_k^2\right)\|\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})\|^2+\frac{2C_k^2}{\mu}\|\mathrm{div}\boldsymbol{\sigma}\|^2+\int_{\partial\Omega}\boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{u}\,, \qquad (7.19)$$

a result which can be finally used in (7.7c):

$$\|\boldsymbol{\sigma}\|^2+\|\operatorname{div}\boldsymbol{\sigma}\|^2+\|\boldsymbol{\varepsilon}(\mathbf{u})\|^2\leq\left(\frac{2\mu^2+1}{\mu}\right)(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\sigma})+\|\operatorname{div}\boldsymbol{\sigma}\|^2+2\|\boldsymbol{\varepsilon}(\mathbf{u})-\mathcal{A}\boldsymbol{\sigma}\|^2 \tag{7.20}$$

$$\leq+\left[\left(\frac{2\mu^2+1}{\mu}\right)\left(5\mu+16\mu C_k^2\right)+2\right]\|\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u})\|^2+ \tag{7.21}$$

$$+\left[\frac{4C_k^2}{\mu}+1\right]\|\operatorname{div}\boldsymbol{\sigma}\|^2+ \tag{7.22}$$

$$+2\int_{\partial\Omega}\boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{u} \tag{7.23}$$

The integral $\int_{\partial\Omega}\boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{u}$ can be splitted into the integrals on Γ_C, Γ_D, Γ_N. Since $\boldsymbol{\sigma}$ and $\mathbf{u}$ satisfy homogenous boundary conditions, the integral boils down to the one on Γ_C. Then we use the frictionless contact condition (4.1d):

$$\int_{\partial\Omega}(\boldsymbol{\sigma}\cdot\mathbf{n})\cdot\mathbf{u}=\int_{\Gamma_C}(\boldsymbol{\sigma}\cdot\mathbf{n})\cdot\mathbf{u}+\int_{\Gamma_D}(\boldsymbol{\sigma}\cdot\mathbf{n})\cdot\mathbf{u}+\int_{\Gamma_N}(\boldsymbol{\sigma}\cdot\mathbf{n})\cdot\mathbf{u} \tag{7.24a}$$

$$=\int_{\Gamma_C}(\boldsymbol{\sigma}\cdot\mathbf{n})\cdot\mathbf{u} \tag{7.24b}$$

$$=\int_{\Gamma_C}(\boldsymbol{\sigma}\cdot\mathbf{n})_n\mathbf{n}\cdot\mathbf{u} \tag{7.24c}$$

$$=\langle\sigma_n,u_n\rangle_{\Gamma_C} \tag{7.24d}$$

Therefore (7.6) follows. □

Remark 7.1.1. *The condition (7.6) derives from the fact that the only bilinear forms have to be elliptic on the convex set. However, the non-negativity of the complementarity term depends also on the linear form in which the gap function g appears. Thus, even though on the convex set the functional is non-negative for any positive choice of γ and δ, the same cannot be guaranteed for the bilinear forms alone. To obtain ellipticity, the weights of the FOSLS functional have to be chosen carefully. As it is clear from (7.6) the constants γ and δ depend on the Korn's constant C_k and on μ. The former is known for simple problems, but in general is difficult to compute exactly, even though it can be approximated. On the other hand, γ and δ are at least independent of λ.*

Remark 7.1.2. *In the FOSLS formulation, the displacement and the stress are both main variables. On the other hand, in the primal and the dual formulations, the stress and the displacement are respectively Lagrange multipliers. For this reason, the complementarity term is not involved in any bilinear form but, at least in the dual formulation, is just represented by the linear form with the gap function g. Consequently, there are no problems of ellipticity regarding this term.*

To avoid problems concerning its ellipticity, the complementarity term could be enforced directly into the convex set $\boldsymbol{K}_f$, but then the constraints would be non-linear. Therefore it is still more practical to consider the augmented functional $\mathcal{J}_f$ and then find a way to correctly choose γ and δ.

7.2 The generalized Arnold-Falk-Winther smoother for the FOSLS formulation

In a monolithic approach like the one used in Starke [1999], the Arnold-Falk-Winther smoother for the FOSLS formulation defines the subspaces to capture local corrections for both $\mathbf{u}$ and $\boldsymbol{\sigma}$. There are no restrictions in terms of the dofs for the stress $\boldsymbol{\sigma}$, except for the fact divergence-free components must be captured. Thus between Figure 6.9a and Figure 6.9b, we can opt for the first choice, which introduces the smallest subspace for $\boldsymbol{\sigma}$. However, since we want to tackle at the same time also the displacement $\mathbf{u}$, the same subspace has to be enriched. With the same philosophy, we would like to introduce the minimum number of displacement dofs and this is why the only displacement dofs related to the internal node are considered. Therefore, the monolithic patch subspace can be defined by considering all dofs corresponding to the internal components for $\mathbf{RT}_0$ and to the internal node for $\mathbf{P}_1$, as in Figure 7.1. The same argument holds also for $d = 3$, if the patch is built on a node. Otherwise, if the patch is built on an edge like in Figure 6.8b, all the displacement dofs related to the two nodes on the internal edge must belong to the local subspace.

Once the subspace of Figure 7.1 is defined, then a local non-linear problem has to be solved. Since the dimension of the subspace is relatively small, we can compute corrections for both $\mathbf{u}$ and $\boldsymbol{\sigma}$ by using an active-set strategy. In this regard, we mention Algorithm 1. Typically few iterations of the method are necessary for its convergence. In particular, if the local set of active dofs is the exact one, then just a local linear problem has to be solved.

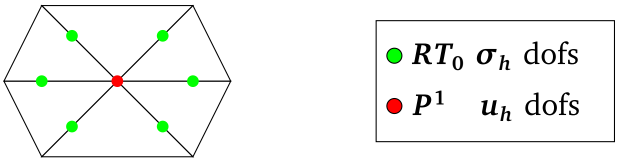


Figure 7.1. FOSLS patch subspace.

7.3 Numerical Experiments with the Arnold-Falk-Winther smoother

In the following numerical experiments, the material is chosen to be incompressible, so that $\lambda = \infty$, which is the problematic case for the primal formulation. For simplicity, we opt also for $\mu = 1$. Furthermore, after the FE discretization, we obtain a linear system of the type:

$$\mathbf{A}_h \mathbf{y}_h = \mathbf{f}_h \,, \tag{7.25}$$

where $\mathbf{A}_h$ and $\mathbf{f}_h$ are the discretized bilinear and linear forms related to the functional (4.5a), while $\mathbf{y}_h$ is the vector of the coefficients of $[\mathbf{u}_h, \boldsymbol{\sigma}_h]^T$. In order to take into consideration the boundary conditions, $\mathbf{A}_h$ and $\mathbf{f}_h$ are also modified accordingly. Then we define the residual of the problem as:

$$\mathbf{r}_h = \mathbf{f}_h - \mathbf{A}_h \mathbf{y}_h \,. \tag{7.26}$$

The MMG applied to the formulation in FOSLS linear elasticity framework works for standard linear elasticity problems, where the Neumann and the Dirichlet boundaries are strictly separated on both the fine and the coarse levels. This is not the case for the Signorini problem, even when it boils down to a linear problem. If we already know which dofs are effectively active on Γ_C, the contact boundary reduces to portions of Dirichlet and Neumann boundaries, at least for the normal components of the stress and displacement. However, it can happen that on the coarse levels the separation between the two kinds of boundaries is not strict.

For example, let us consider the linear problem of Figure 7.2, where a coarse mesh of two triangles on level $j = 0$ is refined, using a bisection algorithm, up to level $J = 6$. On the finest mesh, all Neumann and Dirichlet boundary conditions are zero except for a uniform negative vertical displacement set on the top. The boundary conditions are defined on different elements on the fine mesh of Figure 7.2a, but they intersect on the coarse level $j = 5$ of Figure 7.2b, meaning that there exists a coarse element where both Γ_D and Γ_N are present.

We have discovered that the MMG method is very sensitive to truncations that involve both the primal and the dual variables. The convergence property of MMG can be very poor if standard truncation techniques are used. In this case, the truncation is carried out like in Figure 7.3a. All dofs related to $\mathbf{u}_h$ and $\boldsymbol{\sigma}_h$, that must satisfy boundary conditions, are truncated. Then, in Figure 7.4a, we see that after an initial decrease, the residual is almost constant. This happens for a multilevel algorithm with levels $j = 0, \dots, J$, but also for a two levels method, where the fine problem is on level $J = 6$ and the coarsest one is on level $J - 1$. This means that the coarse correction is too restrictive and thus the effect of truncation is too drastic.

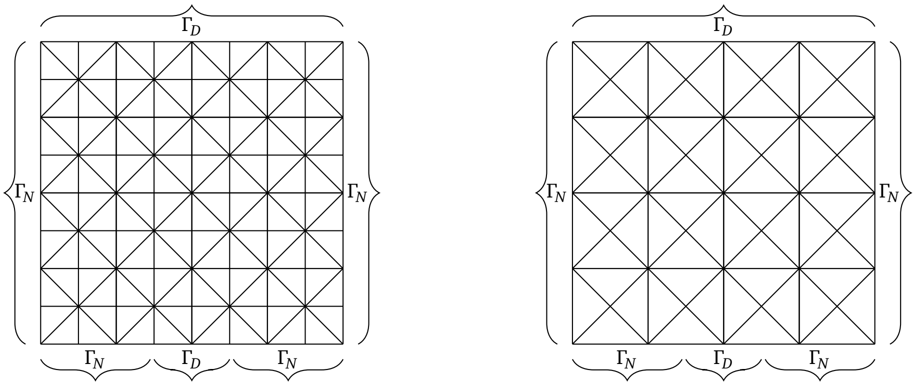
Then we can imagine to partially truncate the displacement $\mathbf{u}_h$ or the stress $\boldsymbol{\sigma}_h$. The overall truncation is standard except for the portions of the coarse boundary where Γ_D and Γ_N intersect. In particular, for the partial truncation of $\mathbf{u}_h$, we do not truncate the *only fine* displacement dofs $\mathbf{u}_h$ which belong to a coarse face where Γ_D and Γ_N intersect. See Figure 7.3b. On the other hand, for the partial truncation of $\boldsymbol{\sigma}_h$, we do not truncate the *only fine* stress dofs $\boldsymbol{\sigma}_h$ which belong to a coarse face where Γ_D and Γ_N intersect. See Figure 7.3c.

We see that for a multilevel strategy applied to the case of Figure 7.3b, the MMG behaviour does not differ from the standard truncation argument: Figure 7.4a and Figure 7.4b are very similar. For the truncation of Figure 7.3c, the results in 7.4c are more promising, at least for few levels. Since the problem is linear, the truncation is applied only at the fine level and no monotone restriction is necessary. Thus all the coarse levels are influenced by the only truncation at the fine level since no other truncation occurs. Then the interpolation operators between all the other levels do not change. So we would expect that, if the fine truncation positively influences the coarse corrections from level $j = 5$, the same will happen for the ones coming from $j = 0, \dots, 4$ as well. Unfortunately, this happens only up to level $j = 4$, and, for all other scenarios, the behavior is still close to the one of Figure 7.4a. This fact suggests that somehow, even the partial truncation is truncating too much the stress components on the coarsest levels.

It is still not known the real root of this weird behavior of the truncation, but we can affirm that:

- the order of the Raviart-Thomas space does not directly influence the MMG method: if we use $\boldsymbol{\Sigma}_h = \mathbf{RT}_1$ instead of $\boldsymbol{\Sigma}_h = \mathbf{RT}_0$, the overall performance of the MMG method is poor as the one in Figure 7.4a;

- the problem is linear, no monotone restriction is used and the cause of an eventual bad convergence rate has to be found elsewhere;

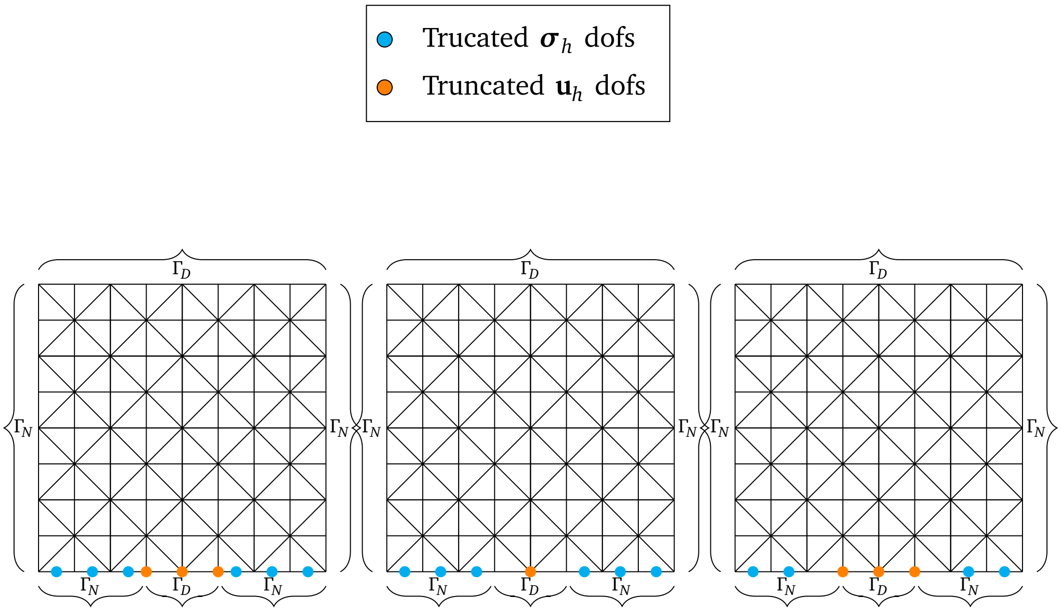


(a) Fine mesh at level $J = 6$. (b) Coarse mesh at level $J - 1 = 5$.

Figure 7.2. In the fine mesh (a) Γ_D and Γ_N are separated boundaries, while on the coarse mesh (b) Γ_D and Γ_N intersect on the same elements.

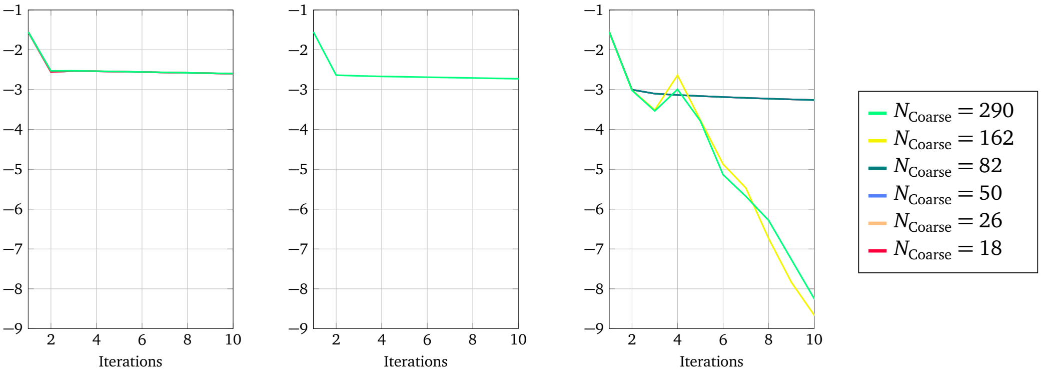
- if Γ_D and Γ_N do intersect on the coarser levels, then
  - the standard truncation does not work;
  - the standard truncation works better if boundary conditions on the stress at the bottom face are enforced weakly as in (3.38) with a small parameter ξ, but, when it increases, the convergence still deteriorates;
  - the partial truncation of $\mathbf{u}_h$ does not work;
  - the partial truncation of $\boldsymbol{\sigma}_h$ sometimes works as in Figure 7.3c;

In conclusion, the main difficulty is connected to the simultaneous presence of essential boundary conditions for both the displacement and the stress on the coarser levels. In particular, the problem is connected to the coarse representation of the stress variable in the case of mixed boundary conditions. We can say that to fully represent the coarse divergence-free components, we need all the fine Raviart-Thomas shape functions. But, if some fine Raviart-Thomas shape functions are truncated, this is not possible anymore. Then, from the MMG perspective, we should truncate, but from the H_{div} smoother point of view, we should not.


● Trucated $\boldsymbol{\sigma}_h$ dofs
● Truncated $\mathbf{u}_h$ dofs
Γ_D
Γ_N
Γ_N
Γ_N
Γ_D
Γ_N

(a) Standard truncation. (b) Partial truncation in $\boldsymbol{u}$. (c) Partial truncation in $\boldsymbol{\sigma}$.

Figure 7.3. Different truncation choices for the dofs in beteewn Γ_D and Γ_N.



(a) Standard truncation. (b) Partial truncation in $\boldsymbol{u}$. (c) Partial truncation in $\boldsymbol{\sigma}$.

Figure 7.4. $\log_{10}$ of the Euclidean norm of the residual for the MMG applied to the FOSLS formulation for the problem in Figure 7.2. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$, $N_{\text{Fine}} = 578$. We do 1000 smoothing steps on the coarsest level.

7.4 The generalized Hiptmair smoother for the FOSLS case

In section 7.3, we have discovered that, whenever Γ_D and Γ_N share the same element on the coarser levels, the MMG method has a bad convergence behavior. In particular, due to the MMG method, we should truncate all the fine shape functions related to the active dofs. But in this way, it is not possible to represent the coarse divergence-free components that belong to the portion of the coarse boundary where Γ_D and Γ_N intersect and consequently the coarse correction is negatively affected.

With respect to the Arnold-Falk-Winther smoother, a different way to tackle the divergence-free components is by means of the Hiptmair's smoother of section 6.12. However, as we have noticed, the issue resides not in the smoother, but in the truncation process. So it is not sufficient to change the type of smoother. Nevertheless, we could take advantage of the idea of computing corrections in the potential space. In particular, we could do a standard V-cycle on the main variables $\mathbf{u}_h$ and $\boldsymbol{\sigma}_h$ and then do a full V-cycle on the potential variable $\boldsymbol{\psi}_h$ of equation (6.58). The standard Hiptmair's smoother damps, on each level, the divergence-free components related to $\boldsymbol{\sigma}_h$ right afterward a pointwise Gauss-Seidel on $\boldsymbol{\sigma}_h$. In our case, we would take care of the divergence-free components in an appropriate V-cycle, where truncation would be carried out directly on the potential variable. Nonetheless, even in this case, the method does not converge properly.

Chapter 8

The dual case

In the previous chapter, we have discovered that the FOSLS formulation is not suited for truncation. The presence of essential boundary conditions for both the primal and the dual variables makes the truncation process difficult to handle. For this reason, in this chapter we examine the dual formulation, that enforces essential boundary conditions only on the stress variable. Therefore, we can assume truncation of the basis would work as in the primal case.

In section 8.1, we first introduce well known approaches for solving saddle point problems: the Schur complement and the Uzawa's methods. Since both require to invert a single-block matrix, they are not suited for the solution of the discrete problem (5.52) in the incompressible limit. For this reason, in section 8.2 local monolithic subspaces for the Arnold-Falk-Winther smoother are introduced. In order to satisfy the constraints, rigid body motions have to be permitted. However, in this way, the local constraints are linear dependent. In sections 8.3 and 8.4, two strategies for solving this issue are examined. Unfortunately, both approaches are not satisfactory. For this reason, the subspace patch is enlarged so that the local constraints are not satisfied exactly. The results in section 8.5 suggests that this direction is more promising. In particular, as shown section 8.6, this smoother makes the MMG method with standard truncation convergent for the problem of Figure 7.2. Nevertheless, it is desirable to reduce the effect of the constraints outside the patch, that cannot be satisfied. For this reason, in section 8.7 the same local problem is modified so that boundary stress dofs are damped. In 8.8 the connection between damping and local Robin conditions is studied. Then the MMG with damping is examined for both the linear case, in section 8.9, and the Signorini case, in section 8.10. Two-levels and multilevels cases are compared as well. Finally, since all the results depend on the local Robin parameter, in section 8.11 some strategies for its automatic computation are proposed.

8.1 The Schur complement method and the Uzawa's method

Let us consider the discrete problem (5.52). For the sake of simplcity, we can assume for the moment $\Gamma_C = \emptyset$, so that the problem (5.52) boils down to a linear one. Thus, in (6.5), we are enforcing either $(\mathbf{l}_h)_i = (\mathbf{u}_h)_i$ for the boundary conditions or $(\mathbf{l}_h)_i = -\infty$ and $(\mathbf{u}_h)_i = +\infty$. In order to simplify the notation, we will assume the boundary conditions to be full Dirichlet or to be directly enforced in the linear system, so that $\mathbf{A}_h$ and $\mathbf{B}_h$ are modified accordingly. Then we obtain:

$$\begin{bmatrix} \mathbf{A}_h & \mathbf{B}_h^T \\ \mathbf{B}_h & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ \mathbf{z}_h \end{bmatrix} = \begin{bmatrix} \mathbf{f}_h \\ \mathbf{h}_h \end{bmatrix} . \tag{8.1}$$

We can also define the residuals corresponding to the whole system, to its first and to its second components:

$$\mathbf{r} := \begin{bmatrix} \mathbf{f}_h \\ \mathbf{h}_h \end{bmatrix} - \begin{bmatrix} \mathbf{A}_h & \mathbf{B}_h^T \\ \mathbf{B}_h & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ \mathbf{z}_h \end{bmatrix} , \tag{8.2a}$$

$$\mathbf{r}_a := \mathbf{f}_h - \mathbf{A}_h \mathbf{y}_h - \mathbf{B}_h^T \mathbf{z}_h , \tag{8.2b}$$

$$\mathbf{r}_b := \mathbf{h}_h - \mathbf{B}_h \mathbf{y}_h . \tag{8.2c}$$

Different are the methods for solving this kind of problem. See Braess, Dietrich [2007]. The most known are for sure the Schur complement technique and Uzawa's method. The first one is a staggered approach, meaning that we first solve for $\mathbf{z}_h$ and then for $\mathbf{y}_h$. We rearrange the system (8.1) so that $\mathbf{y}_h$ is written explicitly in the first equation and substituted in the second one:

$$\begin{cases} \mathbf{A}_h \mathbf{y}_h = (\mathbf{f}_h - \mathbf{B}_h^T \mathbf{z}_h) \\ \mathbf{B}_h \mathbf{y}_h = \mathbf{h}_h \end{cases} \Rightarrow \begin{cases} \mathbf{y}_h = \mathbf{A}_h^{-1}(\mathbf{f}_h - \mathbf{B}_h^T \mathbf{z}_h) \\ \mathbf{B}_h \mathbf{A}_h^{-1} \mathbf{B}_h^T \mathbf{z}_h = \mathbf{B}_h \mathbf{A}_h^{-1} \mathbf{f}_h - \mathbf{h}_h \end{cases} , \tag{8.3}$$

obtaining Algorithm 7.

The matrix $\mathbf{B}_h \mathbf{A}_h^{-1} \mathbf{B}_h^T$, called *the Schur complement*, has to be inverted in the first equation, in order to get $\mathbf{z}_h$. Once $\mathbf{z}_h$ is known, $\mathbf{y}_h$ can be found from the second equation, by inverting $\mathbf{A}_h$. For large systems, the matrices $\mathbf{B}_h \mathbf{A}_h^{-1} \mathbf{B}_h{}^T$ and $\mathbf{A}_h$ can not be inverted directly and iterative solvers must be used. In particular, to compute $(\mathbf{B}_h \mathbf{A}_h^{-1} \mathbf{B}_h^T)^{-1}$, the inverse matrix $\mathbf{A}_h^{-1}$ must be known. Therefore two

Algorithm 7: Schur's complement algorithm

```
Result: $\mathbf{y}_h$, $\mathbf{z}_h$
Input: $\mathbf{f}_h, \mathbf{h}_h$
Solve $\mathbf{z}_h \leftarrow (\mathbf{B}_h\mathbf{A}_h^{-1}\mathbf{B}_h^T)^{-1}(\mathbf{B}_h\mathbf{A}_h^{-1}\mathbf{f}_h - \mathbf{h}_h)$
Solve $\mathbf{y}_h \leftarrow \mathbf{A}_h^{-1}(\mathbf{f}_h - \mathbf{B}_h^T\mathbf{z}_h)$
```

levels of iterations will be required: one inner iteration for inverting $\mathbf{A}_h$ and an outer one for inverting the whole Schur complement.

On the other hand, Uzawa's method is monolithic, meaning that we solve contemporarily for both $\mathbf{y}_h$ and $\mathbf{z}_h$. The saddle point system (8.1) is solved iteratively like in Algorithm 8. As we can see, both approaches require at least to invert the block $\mathbf{A}_h$. Unfortunately, such block is not always well posed. Indeed $\mathbf{A}_h$ is the discretization of the bilinear form $(\mathcal{A}\boldsymbol{\sigma}, \boldsymbol{\sigma})_{L^2(\Omega}$. We remind the reader the definition of $\mathcal{A}\boldsymbol{\sigma}$:

Algorithm 8: Uzawa's algorithm

```
Result: $\mathbf{y}_h$, $\mathbf{z}_h$
Input: $\mathbf{z}_h^0 \in \mathbb{R}^m$, $K_{\max}$, tol
$\mathbf{y}_h^0 = \mathbf{A}_h^{-1}(\mathbf{f}_h - \mathbf{B}_h^T\mathbf{z}_h^0)$
$\mathbf{r}_h^0 = \left[\mathbf{f}_h - \mathbf{A}_h\mathbf{y}_h^0 - \mathbf{B}_h^T\mathbf{z}_h^0, \quad \mathbf{h}_h - \mathbf{B}_h\mathbf{y}_h^0\right]^T$
$k \leftarrow 0$
while $k < K_{\max}$ or $\|\mathbf{r}_h^k\| > tol$ do
    $\mathbf{q}_h^k = \mathbf{h}_h - \mathbf{B}_h\mathbf{y}_h^k$
    $\mathbf{p}_h^k = \mathbf{B}_h^T\mathbf{q}_h^k$
    $\mathbf{s}_h^k = \mathbf{A}_h^{-1}\mathbf{p}_h^k$
    $\alpha^k = \dfrac{{\mathbf{q}_h^k}^T\mathbf{q}_h^k}{{\mathbf{p}_h^k}^T\mathbf{s}_h^k}$
    $\mathbf{z}_h^{k+1} = \mathbf{z}_h^k - \alpha_k\mathbf{q}_h^k$
    $\mathbf{y}_h^{k+1} = \mathbf{y}_h^k + \alpha_k\mathbf{s}_h^k$
    $k \leftarrow k+1$
    $\mathbf{r}_h^k = \left[\mathbf{f}_h - \mathbf{A}_h\mathbf{y}_h^k - \mathbf{B}_h^T\mathbf{z}_h^k, \quad \mathbf{h}_h - \mathbf{B}_h\mathbf{y}_h^k\right]^T$
end
$\mathbf{y}_h \leftarrow \mathbf{y}_h^k$
$\mathbf{z}_h \leftarrow \mathbf{z}_h^k$
```

$$\mathcal{A}\boldsymbol{\sigma} := \frac{1}{2\mu}\left[\boldsymbol{\sigma} - \frac{\lambda}{\lambda+2\mu}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I}\right], \tag{8.4}$$

where, for $\lambda \to \infty$, it reduces to:

$$\lim_{\lambda\to\infty} \mathcal{A}\boldsymbol{\sigma} = \frac{1}{2\mu}\left[\boldsymbol{\sigma} - \mathrm{tr}(\boldsymbol{\sigma})\mathbf{I}\right] . \tag{8.5}$$

It is easy to verify that, in the incompressible limit, the kernel of $\mathcal{A}$ is the set of all tensors which are a multiple of the identity matrix $\mathbf{I}$:

- In 2D:

$$\mathrm{Ker}\left(\lim_{\lambda\to\infty} \mathcal{A}\right) = \{\boldsymbol{\sigma} : \sigma_{xx} = \sigma_{yy}, \quad \sigma_{xy} = \sigma_{yx} = 0\} \tag{8.6}$$

$$= \{\boldsymbol{\sigma} : \boldsymbol{\sigma} = \alpha\mathbf{I}, \quad \alpha \in \mathbb{R}\} \tag{8.7}$$

- In 3D:

$$\begin{aligned}\mathrm{Ker}\left(\lim_{\lambda\to\infty} \mathcal{A}\right) = \{\boldsymbol{\sigma} : \ & \sigma_{xx} = \sigma_{yy} = \sigma_{zz}, \\ & \sigma_{xy} = \sigma_{yx} = \sigma_{xz} = \sigma_{zx} = \sigma_{yz} = \sigma_{zy} = 0\} \\ = \{\boldsymbol{\sigma} : \ & \boldsymbol{\sigma} = \alpha\mathbf{I}, \quad \alpha \in \mathbb{R}\}\end{aligned} \tag{8.8}$$

So, for incompressible materials, but even for large Lamè parameter λ, the discretized matrix $\mathbf{A}_h$ can be only symmetric semi-positive definite. Therefore it cannot be inverted with standard techniques. In addition, the dimension of its kernel increases with its dimension n.

The fulfillment of the discrete LBB conditions (5.53) makes the whole system (8.1) solvable, but there is no guarantee on the invertibility of the single block $\mathbf{A}_h$. The same argument applies for the problem (8.1) represented on smaller subspaces, like the ones in Figure 6.9c, Figure 6.9d and Figure 6.9e. This is the reason for considering a monolithic, and not a staggered, version of the Arnold-Falk-Winther smoother of section 6.13. The main unknown $\mathbf{y}_h$, related to the variable $\boldsymbol{\sigma}_h$, and the Lagrange multiplier $\mathbf{z}_h$, related to $[\mathbf{u}_h, \boldsymbol{\theta}_h]^T$, need to belong to the same subspace, so that the local system is invertible. Let us consider the local i-th subspace on level j. On level j the current iterate is given by $\mathbf{y}_j$ and $\mathbf{z}_j$, with right-hand side $\mathbf{f}_j$ and $\mathbf{h}_j$. Then we define:

$$\mathbf{A}_{j,i} := \mathbf{\Pi}_{j,i}^T \mathbf{A}_j \mathbf{\Pi}_{j,i} , \tag{8.9a}$$

$$\mathbf{B}_{j,i} := \mathbf{Q}_{j,i}^T \mathbf{B}_j \mathbf{\Pi}_{j,i} , \tag{8.9b}$$

$$\mathbf{f}_{j,i} := \mathbf{\Pi}_{j,i}^T (\mathbf{f}_j - \mathbf{A}_j \mathbf{y}_j) , \tag{8.9c}$$

$$\mathbf{h}_{j,i} := \mathbf{Q}_{j,i}^T (\mathbf{h}_j - \mathbf{B}_j \mathbf{z}_j) , \tag{8.9d}$$

where $\mathbf{A}_{j,i} \in \mathbb{R}^{n_{j,i},n_{j,i}}$ and $\mathbf{B}_{j,i} \in \mathbb{R}^{m_{j,i},n_{j,i}}$. See section 6.6 for the definitions of the other operators. The local correction $\mathbf{y}_{j,i} \in \mathbb{R}^{n_{j,i}}$ and $\mathbf{z}_{j,i} \in \mathbb{R}^{m_{j,i}}$ can be computed by solving the following system:

$$\begin{bmatrix} \mathbf{A}_{j,i} & \mathbf{B}_{j,i}^T \\ \mathbf{B}_{j,i} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{j,i} \\ \mathbf{z}_{j,i} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{j,i} \\ \mathbf{h}_{j,i} \end{bmatrix}. \tag{8.10}$$

However, as we will discover, the choice of the subspace is not trivial and will require special care. Fortunately, with respect to the MMG method for the FOSLS formulation of section 7.3, the truncation will not be problematic, since the only essential boundary condition is given by the stress variable. From now on the discussion will concern the 2D case.

8.2 The Arnold-Falk-Winther smoother for the dual formulation

In Figure 6.9c is given the minimum required $\mathbf{RT}_1$ dofs on the patch subspace, to tackle divergence-free components of the error. On the other hand, in Figure 6.9e, all the dofs on the patch are represented. In between the two figures, we can consider Figure 6.9d where the internal momentum dofs are added to the internal dofs of Figure 6.9c. In this way, it would not be necessary to damp separately the error related to those components. Due to the discussion in section 8.1, in addition to $\boldsymbol{\sigma}_h$, the Lagrange multipliers $\mathbf{u}_h$ and θ_h have to be considered as well. Then, the subspace of the patch is enriched as in Figure 8.1. Since only internal dofs of $\boldsymbol{\sigma}_h$ are present, all the corresponding constraints, enforced by means of the associated Lagrange multipliers, live inside the patch and can be fulfilled.

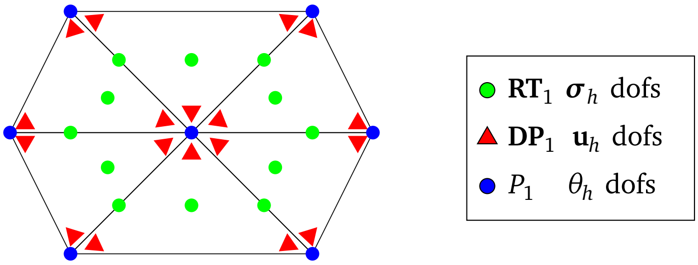

● $\mathbf{RT}_1$ $\boldsymbol{\sigma}_h$ dofs
▲ $\mathbf{DP}_1$ $\mathbf{u}_h$ dofs
● P_1 θ_h dofs

Figure 8.1. Subspace referred to a patch without considering $\boldsymbol{RT}_1$ dofs on the border.

Neglecting the $\boldsymbol{\sigma}_h$ dofs on the boundary is equivalent to enforce local homogeneous Neumann boundary conditions. In linear elasticity, a body that on the boundary is subject only to external forces, can freely translate and rotate. For $d = 2$ the rigid body motions are three in total, given by two translations and one rotation, while for $d = 3$ there are three rotations and three translations. Focusing on the case $d = 2$ with only Neumann boundary conditions, we would expect a local system that can be solved only up to the rigid body motions. In the saddle point problem (8.10) related to the subspace of Figure 8.1, such property shows up in the linear dependency of the constraints. Indeed the local matrix $\mathbf{B}_{j,i} \in \mathbb{R}^{m_{j,i} \times n_{j,i}}$ has not full rank if $m_{j,i} > n_{j,i}$. In particular, if free rigid body motions are permitted, it should happen that $m_{j,i} = n_{j,i} + 3$. However, this is not always the case. In Figure 8.2, we notice that for a single element, $n_{j,i} = 4$, $m_{j,i} = 9$ and $m_{j,i} - n_{j,i} = 5$. For two elements, $n_{j,i} = 12$, $m_{j,i} = 16$ and $m_{j,i} - n_{j,i} = 4$. Finally, for three elements $n_{j,i} = 20$, $m_{j,i} = 23$ and $m_{j,i} - n_{j,i} = 3$. Thus a patch of the kind of Figure 8.1 needs to be built at least on three elements. Otherwise the local matrix $\mathbf{B}_{j,i}$ of the problem (8.10) has too many linear dependent rows, even if we accept free rigid body motions.

Up to now, we have considered, as patches related to a node, the set of elements that share that node. For fine enough meshes, the patches, that do not have the internal node on the boundary, have always at least three triangles. However, on the border, no matter how fine the mesh is, a patch can consist of only one or two elements. Since this is the situation we want to avoid, we must generalize our definition of patch. For a patch related to the node ν_1, we do the following

- if it consists of at least three elements, then it is accepted;
- otherwise, we add all the elements of the patch related to the node ν_2, where $\nu_2 \neq \nu_1$ and ν_2 belongs to one of the element of the patch; if the new patch does not have at least three elements, the procedure is repeated for another node ν_3 of the new patch, with $\nu_3 \neq \nu_2, \nu_1$, until the patch consists of three elements; see Figure 8.3 for an example of this procedure.

In the case of patches built on the Dirichlet boundary, it could be not necessary to enrich the local patch. Indeed, one or two elements could have enough $\boldsymbol{\sigma}_h$ dofs. Nevertheless, since it cannot be known in advance if it is necessary or not, it is easier to extend the above argument for enriching the patches to all patches of the mesh.

Once the enlarged patches have been defined, the local problem can not sill be solved, because rigid body motions are allowed. To overcome this difficulty, different are the strategies we could think of:

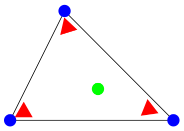
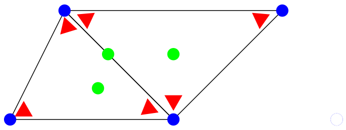
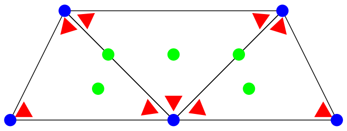
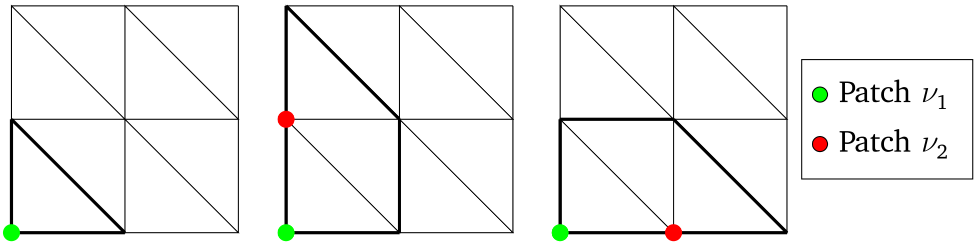
Legend dofs	Subspace	# 🟢	# 🔺	# 🔵	# 🔺+🔵
🟢 $\mathbf{RT}_1$ $\boldsymbol{\sigma}_h$ 🔺 $\mathbf{DP}_1$ $\mathbf{u}_h$ 🔵 P_1 θ_h		4	6	3	9
		12	12	4	16
		20	18	5	23

Figure 8.2. The table shows how many dofs are related to the unknowns $\boldsymbol{\sigma}_h$ and how many to the Lagrange multipliers $\boldsymbol{u}_h$ and θ_h. For full Neumann boundary conditions, only rigid body motions should be allowed, so the difference of Lagrange multiplier and stress dofs should be equal to 3. This happens if we consider a patch of at least 3 elements, where we do have $\#\text{dofs}(\boldsymbol{u}_h + \theta_h) = 23$ and $\#\text{dofs}(\boldsymbol{\sigma}_h) = 20$ and $\#\text{dofs}(\boldsymbol{u}_h + \theta_h) - \#\text{dofs}(\boldsymbol{\sigma}_h) = 3$.

1. Removing rigid body motions from the local system, not considering the dofs related to one displacement, in both directions, and one rotation; of course, on a single patch, many combinations can be taken into consideration. See section 8.3.

2. Solving the system up to rigid body motions, searching for the local displacement and rotation which have zero-average on the patch. See section 8.4.

3. Enriching the patch by adding also the stress boundary dofs. See section 8.5.



(a) Starting patch. (b) Enlarged patch. (c) Enlarged patch.

Figure 8.3. Starting from the patch built on the node ν_1 of Figure 8.3a, we can have two possible enlarged patches, Figure 8.3b or Figure 8.3c, depending on the choice of ν_2.

8.3 Removal of one displacement and one rotation

Let us examine Cook's membrane problem, which is represented in Figure 8.4. In figure 8.5, we represent the case in which, from the subspace of Figure 8.1, a rotation and a displacement dofs are removed. Since, on a given patch, many are the rotation and displacement dofs, as many combinations of removal of dofs can be made. For simplicity, after having locally numbered all the dofs, the first two components of the displacement and the last rotation of the patch are removed. The result of this specific case is represented in Figure 8.6, where the definition of the residual is given in (8.2). However, no matter how the removal process of the dofs is realized, a divergent smoother is recovered. Even for a simple small problem of pure rigid translation, the method, even though it is not divergent, oscillates and does not converge.

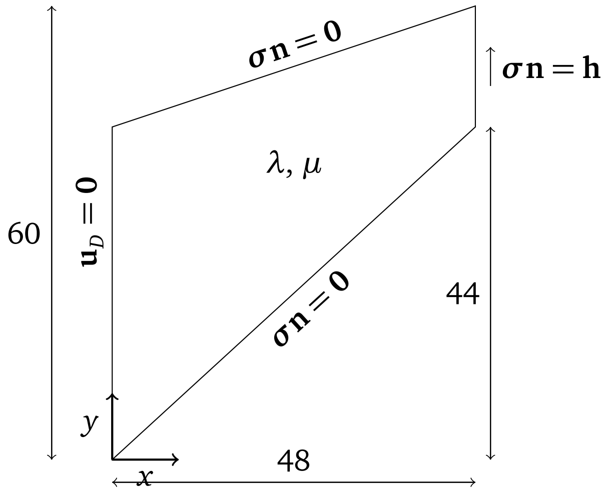

$\sigma\mathbf{n} = \mathbf{0}$
$\sigma\mathbf{n} = \mathbf{h}$
λ, μ
60
$\mathbf{u}_D = \mathbf{0}$
$\sigma\mathbf{n} = \mathbf{0}$
44
y
x
48

Figure 8.4. Cook's membrane problem.

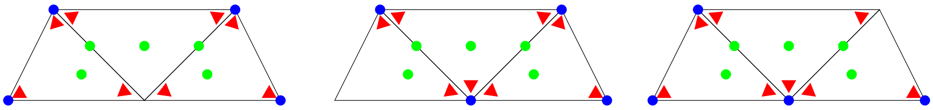


Figure 8.5. Examples of patch built on the boundary where the displacement and rotation dofs related to a given node are removed from the subspace.

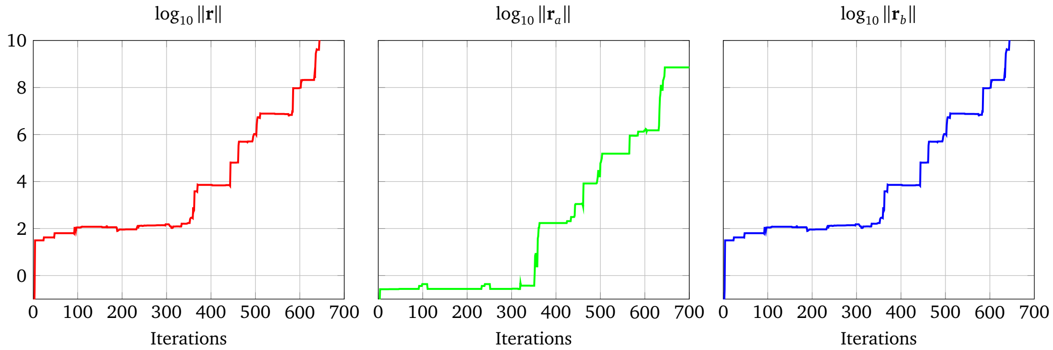


Figure 8.6. $\log_{10}$ of the Euclidean norm of the residual for the Arnold-Falk-Winther smoother applied to dual formulation with subspaces of the type of Figure 8.5 for the Cook's problem in Figure 8.4.

8.4 Enforcing zero-average multipliers

For enlarged patches, the system is solvable up to rigid body motions. A strategy to make the system solvable is to enforce that the local displacement and the local rotation have zero-average on the patch. For $d = 2$:

$$\int_{\text{Patch}} u_{h,x} = 0\,, \qquad \int_{\text{Patch}} u_{h,y} = 0\,, \qquad \int_{\text{Patch}} \theta_h = 0\,, \tag{8.11}$$

where we denote the first and second components of $\mathbf{u}_h$ by $u_{h,x}$ and $u_{h,y}$, while the rotation θ_h is a scalar quantity. It is clear that the test function is the function which is constant on the whole patch. So the same expressions can be assembled globally into three vectors, from which it is possible to extract the local vectors related to the only dofs on the patch:

$$\int_{\Omega} u_{h,x}\,1 = 0\,, \qquad \int_{\Omega} u_{h,y}\,1 = 0\,, \qquad \int_{\Omega} \theta_h\,1 = 0\,. \tag{8.12}$$

By applying the Arnold-Falk-Winther smoother which locally enforces (8.11) to the Cook's problem of Figure 8.4, we obtain the results in Figure 8.7. In the first iterations, the norm of the residual increases. Then it starts to decrease relatively fast and, after some iterations, very slowly. It is clear that the norm $||\mathbf{r}||$ is governed by $||\mathbf{r}_b||$, since $||\mathbf{r}_b|| \gg ||\mathbf{r}_a||$. So, by adding the condition (8.11) the system is now solvable, but the constraints are not properly captured. It is not true that locally the solution must satisfy the zero-average condition for the displacement and the rotation. This is just a trick to make the local system solvable, but it also suggests that the choice of the subspace is not optimal. Besides, the most important iterations of a smoother are the initial ones. But here we see that many iterations are required to have a decrease in the residual. For this reason, another strategy is necessary.

8.5 Patch enriched with stress dofs on the boundary patch

The subspace of Figure 8.1 gives rise to a local system that allows for rigid body motions, but which, in principle, satisfies all the constraints belonging to the patch. In order to make the system (8.10) solvable, on one hand, in section 8.3 it is required to neglect some of these constraints, by removing one rotation and one displacement. On the other hand, the idea of section 8.4 is to consider all the

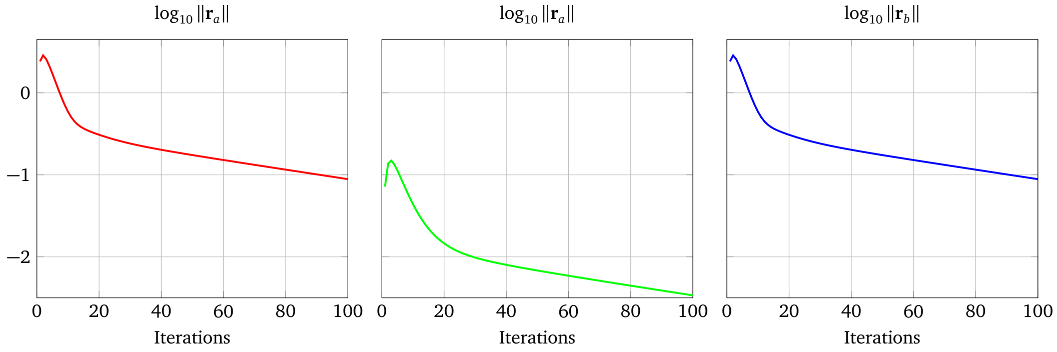


Figure 8.7. $\log_{10}$ of the Euclidean norm of the residual for the Arnold-Falk-Winther smoother applied to the dual formulation for the Cook's problem in Figure 8.4. Locally (8.11) is enforced. Parameters: $\mu = 1$, $\lambda = 1$, number of smoothing steps $= 100$.

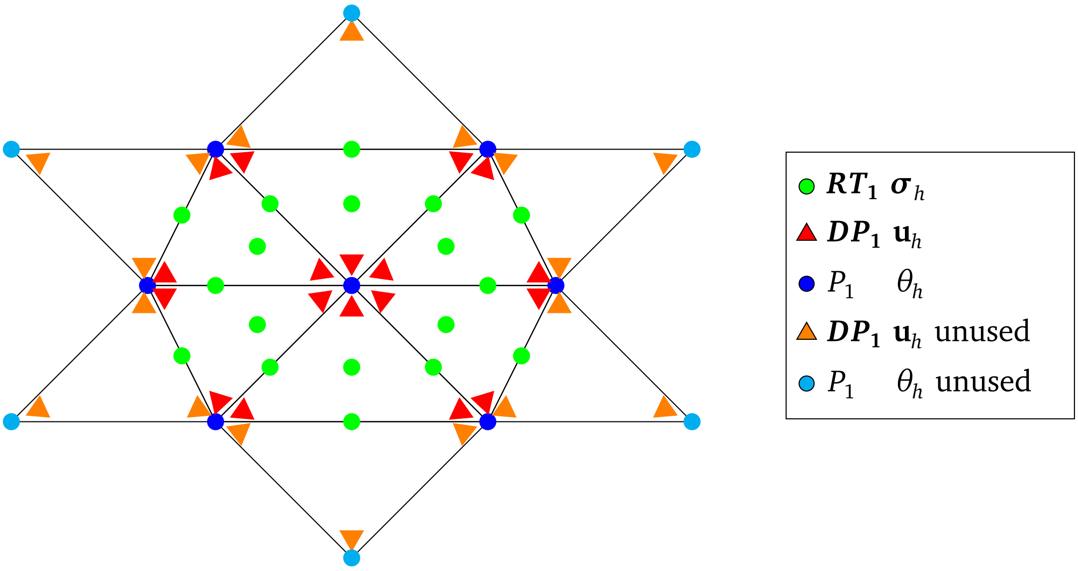

● $\boldsymbol{RT}_1$ $\boldsymbol{\sigma}_h$
▲ $\boldsymbol{DP}_1$ $\mathbf{u}_h$
● P_1 θ_h
▲ $\boldsymbol{DP}_1$ $\mathbf{u}_h$ unused
● P_1 θ_h unused

Figure 8.8. Subspace referred to a patch considering $\boldsymbol{RT}_1$ dofs on the border. All the constraints outside the patch, related to the orange and cyan dofs, are unused but should somehow influence the local solution, since they communicate with the $\boldsymbol{RT}_1$ dofs on the border.

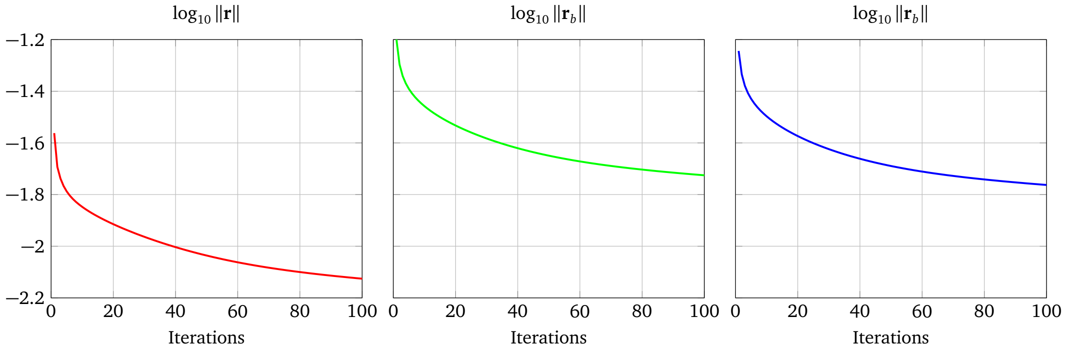


Figure 8.9. $\log_{10}$ of the Euclidean norm of the residual for the Arnold-Falk-Winther smoother applied to dual formulation for the Cook's problem in Figure 8.4. The patch is extended as in Figure 8.8. Parameters: $N_{\text{dofs}} = 4089$, $\mu = 1$, $\lambda = 1$, number of smoothing steps $= 100$.

constraints plus the zero-average constraints for the rotation and the displacement. Therefore none of the two strategies can solve (8.10) on the subspace of Figure 8.1 without some modifications of the constraints.

The main issue derives from the choice of the stress dofs, which are not enough to avoid rigid body motions. Therefore, we now enrich the subspace of Figure 8.1 by considering also $\boldsymbol{\sigma}_h$ dofs on the boundary patch. The full Neumann problem of Figure 8.1 is transformed into the full Dirichlet problem of Figure 8.8. All the constraints, corresponding to displacement (orange triangles) and rotation (cyan circles) dofs outside the patch but communicating with the $\boldsymbol{\sigma}_h$ dofs on the border, cannot be fulfilled. Hence $||\mathbf{r}_b||$ can be expected sometimes to increase after the addition, to the current solution, of the local correction. Nevertheless, since the patches overlap and communicate with each other, it is important the behavior of the residual, not after single local corrections, but after a whole smoothing step. In particular, in Figure 8.9, we can notice that the norm of the residual is decreasing after each smoothing step. In contrast to Figure 8.7, now $\log_{10}||\mathbf{r}||$ is decreasing monotonically with a much better rate of the decay. This fact suggests that the enrichment of the boundary $\boldsymbol{\sigma}_h$ dofs is the strategy to focus on. We will generalize the smoother proposed here in section 8.7. But first, we would like to check if the problem of Figure 7.2 can be easily solved by means of the MMG method applied to the dual formulation, with standard truncation techniques and with the smoother presented here.

8.6 Standard truncation works for the dual formulation

In section 7.3 the problem of Figure 7.2 has been investigated. It is known that standard truncation works in the primal case, but we have shown that many difficulties can arise for the FOSLS formulation. In this section, we show that the primal and the dual formulations behave similarly in terms of truncation if it is adopted a smoother with local subspaces like the ones in Figure 8.8. Indeed using the MMG method for the dual formulation applied to the problem of Figure 7.2, we get the results of Figure 8.10. It is clear that, even for the multilevel case, the MMG method does not have any problems due to the truncation. This is true even in the incompressible limit, $\lambda \to \infty$. This fact strengthens the idea that the truncation in the FOSLS formulation is problematic for the simultaneous presence of essential boundary conditions for both the primal and the dual variables.

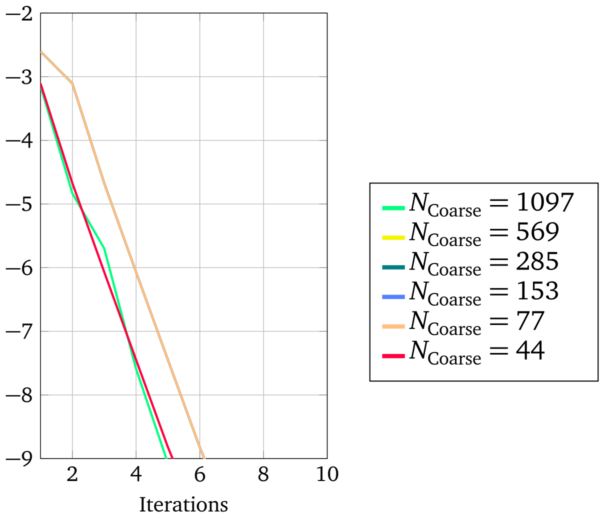


Figure 8.10. $\log_{10}$ of the Euclidean norm of the residual for the MMG applied to the dual formulation for the problem in Figure 7.2. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$, $N_{\text{fine}} = 2193$.

8.7 Damping of the stress boundary dofs (Robin conditions)

Among the choices examined so far, the best performance of the smoother, which also fits the MMG method with truncation, is given for a patch subspace which tackles also the boundary dofs of the stress. Nevertheless, in this way, all the constraints outside the patch cannot be fulfilled, since the corresponding Lagrange multipliers are neglected from the system to be solved. For this reason, it could be profitable to damp the components of the local correction related to stress boundary dofs: the constraints outside the patch would be violated in a minor way and maybe the convergence could be beneficially affected. The easier way to obtain this penalization is by adding to $\mathbf{A}_{j,i}$ in (8.10) a semi-positive definite diagonal matrix that is non-zero only in the positions related to stress boundary dofs. The new system is not arbitrary and unrelated to the original one, but instead can be interpreted as an average between the cases of Figure 8.1 and Figure 8.8.

By keeping in mind that we are always referring to a local system, the problem (8.10) can be rewritten in a simpler form by removing the indices of the level j and of the subspace i. The local dofs of $\mathbf{y}_{j,i}$ can be decomposed into the internal and boundary dofs. Thus we obtain the local vectors $\mathbf{y}_{\text{int}}$ and $\mathbf{y}_{\text{ext}}$ for the correction and $\mathbf{f}_{\text{int}}$ and $\mathbf{f}_{\text{ext}}$ for the right-hand side. All other local quantities will be denoted with the subscript “loc”. The full Dirichlet problem related to Figure 8.8 becomes:

$$\begin{bmatrix} \mathbf{A}_{\text{ext,ext}} & \mathbf{A}^T_{\text{int,ext}} & \mathbf{B}^T_{\text{int,ext}} \\ \mathbf{A}_{\text{int,ext}} & \mathbf{A}_{\text{int,int}} & \mathbf{B}^T_{\text{int,int}} \\ \mathbf{B}_{\text{int,ext}} & \mathbf{B}_{\text{int,int}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\text{ext}} \\ \mathbf{y}_{\text{int}} \\ \mathbf{z}_{\text{loc}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\text{ext}} \\ \mathbf{f}_{\text{int}} \\ \mathbf{h}_{\text{loc}} \end{bmatrix}, \tag{8.13}$$

while the full Neumann problem of Figure 8.1 is:

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\text{int,int}} & \mathbf{B}^T_{\text{int,int}} \\ \mathbf{0} & \mathbf{B}_{\text{int,int}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\text{ext}} \\ \mathbf{y}_{\text{int}} \\ \mathbf{z}_{\text{loc}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_{\text{int}} \\ \mathbf{h}_{\text{loc}} \end{bmatrix}. \tag{8.14}$$

Since the boundary conditions of (8.14) are homogeneous, the problem (8.14) is also equivalent to the following linear system:

$$\begin{bmatrix} \mathbf{G}(\alpha) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\text{int,int}} & \mathbf{B}^T_{\text{int,int}} \\ \mathbf{0} & \mathbf{B}_{\text{int,int}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\text{ext}} \\ \mathbf{y}_{\text{int}} \\ \mathbf{z}_{\text{loc}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_{\text{int}} \\ \mathbf{h}_{\text{loc}} \end{bmatrix}, \tag{8.15}$$

where, given the positive scalar $\alpha > 0$, $\mathbf{G}(\alpha)$ is a positive definite diagonal matrix defined in the following way:

$$\mathbf{G}_{p,q}(\alpha) = \begin{cases} \alpha \max\limits_{s} \max\Big(|(\mathbf{A}_{\text{ext,ext}})_{p,s}|, |(\mathbf{A}^T_{\text{int,ext}})_{p,s}|, |(\mathbf{B}^T_{\text{int,ext}})_{p,s}| \Big) & p = q \\ 0 & p \neq q \end{cases}. \tag{8.16}$$

As already mentioned, the system (8.15) is not well-posed. However, we can create a system as the average of (8.15) and (8.13). In particular, in (8.15) the system is decoupled: $\mathbf{y}_{\text{ext}}$ and $[\mathbf{y}_{\text{int}}, \mathbf{z}_{\text{loc}}]^T$ are independent. For this reason, we can leave the homogeneous boundary conditions as they are and multiply the remaining part of the system by a scalar ϵ. Due to the decoupling, the system is still equivalent for any $\epsilon > 0$. Thus we can sum up the modified problem (8.15) and (8.13), make ϵ tend to zero and get the following solvable system:

$$\begin{bmatrix} \mathbf{A}_{\text{ext,ext}} + \mathbf{G}(\alpha) & \mathbf{A}^T_{\text{int,ext}} & \mathbf{B}^T_{\text{int,ext}} \\ \mathbf{A}_{\text{int,ext}} & \mathbf{A}_{\text{int,int}} & \mathbf{B}^T_{\text{int,int}} \\ \mathbf{B}_{\text{int,ext}} & \mathbf{B}_{\text{int,int}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\text{ext}} \\ \mathbf{y}_{\text{int}} \\ \mathbf{z}_{\text{loc}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\text{ext}} \\ \mathbf{f}_{\text{int}} \\ \mathbf{h}_{\text{loc}} \end{bmatrix}. \qquad (8.17)$$

The new subspace is represented in Figure 8.11.

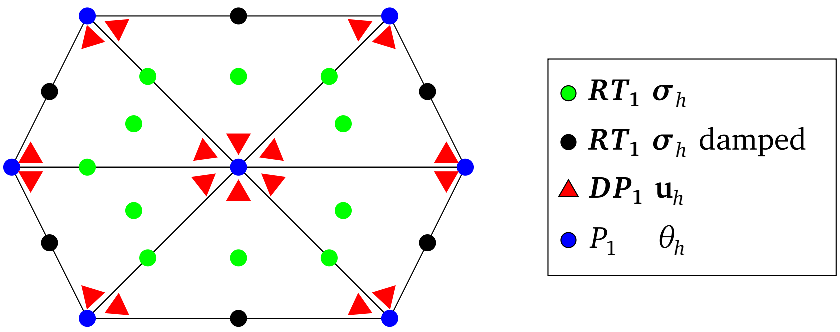

● $\boldsymbol{RT}_1$ $\boldsymbol{\sigma}_h$
● $\boldsymbol{RT}_1$ $\boldsymbol{\sigma}_h$ damped
▲ $\boldsymbol{DP}_1$ $\mathbf{u}_h$
● P_1 θ_h

Figure 8.11. Subspace referred to a patch considering damped $\boldsymbol{RT}_1$ dofs on the border.

8.8 Discrete Robin conditions

In section 8.7, a local system that damps the boundary stress dofs of the patch has been introduced as the average of the full Dirichlet and the full Neumann approaches. Intuitively, in the new problem, discrete Robin conditions are enforced. The choice of using Robin, instead of Dirichlet or Neumann boundary conditions, in a domain decomposition approach, has been proved to be an efficient choice. See Efstathiou, Evridiki and Gander, Martin J. [2003], Gander, Martin Jakob [2008], Gander, Martin J. and Halpern, Laurence and Magoules, Frédéric [2007], Gander, Martin and Halpern, Laurence and Magoulès, Frédéric and Roux, François-Xavier [2007], St-Cyr, Amik and Gander, Martin J. and Thomas, Stephen J. [2007], Gander, Martin J. and Vanzan, Tommaso [2019].

Since boundary conditions are usually imposed at a continuous level and then discretized, it is interesting to understand if the local discrete boundary conditions in (8.17) can be somehow related to the discrete or the continuous Robin boundary conditions. For the sake of simplicity, the standard Poisson problem will be examined. Since this digression is based on the computation of local corrections that have homogeneous boundary conditions, homogeneous boundary conditions will be assumed in the following.

8.8.1 Discrete Robin conditions for the primal formulation

Let $u_1, u_2, u_3 \in H^1(\Omega)$, $f \in H^{-1}(\Omega)$. Let us consider the full Dirichlet Poisson problem:

$$\begin{aligned} -\Delta u_1 &= f \quad \text{on}\,\Omega\,, \\ u_1 &= 0 \quad \text{on}\,\partial\Omega\,, \end{aligned} \tag{8.18}$$

the full Neumann Poisson problem:

$$\begin{aligned} -\Delta u_2 &= f \quad \text{on}\,\Omega\,, \\ \nabla u_2 \cdot \mathbf{n} &= 0 \quad \text{on}\,\partial\Omega\,, \end{aligned} \tag{8.19}$$

and the full Robin Poisson problem:

$$\begin{aligned} -\Delta u_3 &= (1+\alpha) f \quad && \text{on}\,\Omega\,, \\ u_3 + \alpha \nabla u_3 \cdot \mathbf{n} &= 0 && \text{on}\,\partial\Omega\,. \end{aligned} \tag{8.20}$$

If we multiply (8.19) by α and add it to (8.18), we can define $w = u_1 + \alpha u_2$. Then the volumetric equation can be written as $-\Delta w = (1+\alpha)f$, like in (8.20), if $w = u_3$. However, the linear combination of the boundary conditions cannot be represented in terms of the new variable w. Therefore at a continuous level, we can represent the only Robin boundary conditions as the average of the Dirichlet and Neumann boundary conditions, but a full Robin problem cannot be seen as the average of a full Dirichlet problem and a full Neumann problem.

However at a discrete level, it is possible to determine a connection between (8.20) and the average of (8.18) and (8.19). To this purpose, let us consider the following weak form:

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f\,v + \int_{\partial\Omega} \nabla u \cdot \mathbf{n}\,v\,, \tag{8.21}$$

where $u, v \in H^1(\Omega)$. By applying homogeneous Robin boundary conditions $u + \alpha \nabla u \cdot \mathbf{n} = 0$, we get:

$$\int_\Omega \nabla u \cdot \nabla v + \int_{\partial\Omega} \alpha\,u\,v = \int_\Omega f\,v \tag{8.22}$$

If we descretize the problem by means of Lagrangian finite elements and we collect in $\mathbf{u}_1$ and in $\mathbf{u}_2$ respectively all the boundary and all the internal dofs, we obtain the following system:

$$\begin{bmatrix} \mathbf{D}_{1,1} + \alpha \mathbf{M} & \mathbf{D}_{2,1}^T \\ \mathbf{D}_{2,1} & \mathbf{D}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}, \tag{8.23}$$

where the matrices $\mathbf{D}$ and $\mathbf{M}$ are respectively the discretizations of the bilinear forms $a(u, v) = \int_\Omega \nabla u \cdot \nabla v$ and $m(u, v) = \int_{\partial\Omega} uv$, while the vector $\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2]^T$ discretize the linear form $f(v) = \int_\Omega f\ v$.

Similarly, by enforcing homogeneous Dirichlet boundary conditions in (8.21), we get:

$$\begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_2 \end{bmatrix}, \tag{8.24}$$

where $\mathbf{G}$ is a positive definite diagonal matrix. If we enforce homogenous Neumann conditions, we obtain:

$$\begin{bmatrix} \mathbf{D}_{1,1} & \mathbf{D}_{2,1}^T \\ \mathbf{D}_{2,1} & \mathbf{D}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \tag{8.25}$$

As explained in section 8.7, the system (8.24) is decoupled and thus we can multiply the second part by ϵ and making it tend to zero. The obtained system can be added to (8.24), giving rise to:

$$\begin{bmatrix} \mathbf{D}_{1,1} + \mathbf{G} & \mathbf{D}_{2,1}^T \\ \mathbf{D}_{2,1} & \mathbf{D}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \tag{8.26}$$

In contrast to the continuous case, we notice a relation between (8.23) and (8.26). In the first block, equation (8.23) has a boundary mass matrix, while equation (8.26) has just a diagonal matrix. By defining each diagonal entry of $\mathbf{G}$ as a scaled sum of the corresponding row in $\mathbf{M}$:

$$\mathbf{G}_{ii} = \left(\frac{\mathbf{G}_{ii}}{\sum_j \mathbf{M}_{ij}} \right) \sum_j \mathbf{M}_{i,j} = \beta_i \sum_j \mathbf{M}_{i,j}, \tag{8.27}$$

the matrix $\mathbf{G}$ now represents a lumping of $\mathbf{M}$, scaled by the coefficients β_i. Therefore the discrete Robin conditions in (8.26) represent the lumped discretized version of the continuous Robin conditions in (8.23), where varying coefficients β_i are used instead of α.

8.8.2 Discrete Robin conditions for the dual formulation

In section 8.8.1, we have determined for the primal case a relation between the discrete Robin conditions explained in section 8.7 and the lumped discretized version of the continuous Robin conditions. The same argument can be generalized to the dual formulation. Let us consider the strong formulation of the Poisson problem:

$$\begin{aligned} -\mathrm{div}\boldsymbol{\sigma} &= \mathbf{f} \qquad \text{on } \Omega\,, \\ \boldsymbol{\sigma} &= \nabla u \quad\ \text{on } \Omega\,. \end{aligned} \tag{8.28a}$$

We multiply the equilibrium equation by a test function $v \in L^2(\Omega)$ and the constitutive equation by a test $\boldsymbol{\tau} \in H_{\mathrm{div}}(\Omega)$. Then we integrate the second equation by parts and obtain:

$$\begin{aligned} \int_\Omega \boldsymbol{\sigma}\cdot\boldsymbol{\tau} + \int_\Omega u\,\mathrm{div}\boldsymbol{\tau} - \int_{\partial\Omega} u\ (\boldsymbol{\tau}\cdot\mathbf{n}) = 0 \qquad & \forall \boldsymbol{\tau} \in H_{\mathrm{div}}(\Omega)\,, \\ \int_\Omega \mathrm{div}\boldsymbol{\sigma}\ v = \int_\Omega f\,v \quad & \forall v \in L^2(\Omega)\,. \end{aligned} \tag{8.29a}$$

By enforcing homogeneous Robin boundary conditions for the dual formulation, i.e:

$$u + \alpha\boldsymbol{\sigma}\cdot\mathbf{n} = 0\,, \tag{8.30}$$

we seek for $(u, \boldsymbol{\sigma}) \in L^2(\Omega) \times H_{\mathrm{div}}(\Omega)$ such that:

$$\begin{aligned} \int_\Omega \boldsymbol{\sigma}\cdot\boldsymbol{\tau} + \int_\Omega u\,\mathrm{div}\boldsymbol{\tau} + \int_{\partial\Omega} \alpha\ (\boldsymbol{\sigma}\cdot\mathbf{n})\ (\boldsymbol{\tau}\cdot\mathbf{n}) = 0 \qquad & \forall \boldsymbol{\tau} \in H_{\mathrm{div}}(\Omega)\,, \\ \int_\Omega \mathrm{div}\boldsymbol{\sigma}\ v = \int_\Omega f\,v \quad & \forall v \in L^2(\Omega)\,. \end{aligned} \tag{8.31a}$$

If we descretize the problem by means of Raviart-Thomas finite elements and we collect in $\boldsymbol{\sigma}_1$ and in $\boldsymbol{\sigma}_2$ respectively all the boundary and all the internal dofs, we obtain the following system:

$$\begin{bmatrix} \mathbf{S}_{1,1} + \alpha\mathbf{N} & \mathbf{S}_{2,1}^T & \mathbf{T}_1^T \\ \mathbf{S}_{2,1} & \mathbf{S}_{2,2} & \mathbf{T}_2^T \\ \mathbf{T}_1 & \mathbf{T}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_3 \end{bmatrix} \tag{8.32}$$

where the matrices $\mathbf{S}$, $\mathbf{N}$ and $\mathbf{T}$ are respectively the discretizations of the bilinear forms $s(\boldsymbol{\sigma},\boldsymbol{\tau}) = \int_\Omega \boldsymbol{\sigma}\cdot\boldsymbol{\tau}$, $n(\boldsymbol{\sigma},\boldsymbol{\tau}) = \int_{\partial\Omega}(\boldsymbol{\sigma}\cdot\mathbf{n})\ (\boldsymbol{\tau}\cdot\mathbf{n})$ and $t(\boldsymbol{\sigma}, v) = \int_\Omega \mathrm{div}\boldsymbol{\sigma}\ v$, while the vector $\mathbf{f}_3$ discretizes $f(v) = \int_\Omega f\, v$. Also in the dual formulation, Robin conditions give rise to a mass matrix on the border, $\mathbf{N}$. In principle, some components of the volumetric mass matrix obtained by means of Raviart-Thomas function can be negative. However, the trace of the Raviart-Thomas shape functions for simplicial meshes reduces to discontinuous Lagrange finite elements on the border. For this reason, the mass matrix **N** on the border is still similar to the one of the primal case, **M**.

As for the primal case, the two discrete problems, with homogenous Dirichlet and homogenous Neumann boundary conditions, can be introduced. For the full Dirichlet case, we get:

$$\begin{bmatrix} \mathbf{S}_{1,1} & \mathbf{S}_{2,1}^T & \mathbf{T}_1^T \\ \mathbf{S}_{2,1} & \mathbf{S}_{2,2} & \mathbf{T}_2^T \\ \mathbf{T}_1 & \mathbf{T}_2 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_3 \end{bmatrix}, \tag{8.33}$$

while for Neumann boundary conditions, we get:

$$\begin{bmatrix} \mathbf{G} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{2,2} & \mathbf{T}_2^T \\ \mathbf{0} & \mathbf{T}_2 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mathrm{f}_3 \end{bmatrix}. \tag{8.34}$$

With the same trick explained in section 8.7, we average between (8.35) and (8.34), obtaining:

$$\begin{bmatrix} \mathbf{S}_{1,1} + \mathbf{G} & \mathbf{S}_{2,1}^T & \mathbf{T}_1^T \\ \mathbf{S}_{2,1} & \mathbf{S}_{2,2} & \mathbf{T}_2^T \\ \mathbf{T}_1 & \mathbf{T}_2 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_3 \end{bmatrix}, \tag{8.35}$$

The comparison between (8.35) and (8.32) is identical to the one of the primal case. Thus, even for the dual formulation, the discrete Robin conditions are a lumped version of the discretized continuous Robin conditions. The matrix **G** for both the primal and the dual cases can be implemented more easily than the continuous corresponding version, i.e., **N** and **M**. It is not necessary to assemble the boundary mass matrix, but it is sufficient to know which are the boundary dofs. This is especially true if the assembly has to be repeated for patch subdomains which have a boundary that does not necessarily coincide with $\partial\Omega$.

8.9 Numerical examples in dual linear elasticity

In this section, we will investigate the convergence behavior of the smoother of section 8.7 for the dual elasticity problem. Due to the equation (8.17), the smoother now depends on an input parameter $\alpha \geq 0$. For $\alpha = 0$, we recover the smoother of section 8.5. For $\alpha > 0$, the smoother will behave differently and here we want to examine how the value of α can affect the MMG method convergence. The initial guess is always zero except for the boundary conditions, which are satisfied exactly. The residual is computed as in (8.2).

8.9.1 MMG for the Cook's membrane problem

We now consider the Cook's problem, as depicted in Figure 8.4. On the left edge, we enforce a zero displacament. On the right edge, a vertical force is applied: $\boldsymbol{\sigma}\mathbf{n} = [0, 0.01]^T$. Everywhere else, we impose homogeneous Neumann conditions, $\boldsymbol{\sigma}\mathbf{n} = \mathbf{0}$. The material has the following parameters: $\mu = 1$ and $\lambda = \infty$. We will analyze the cases $\alpha = 0$, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 2, 5, 10, 100. The solutions can be found in Figure 8.16.

Figure 8.12 illustrates a convergent, even though slow, behavior of the smoother for different values of α. The parameter $\alpha \leq 1$ does not seem to affect too much the performance. Values of $\alpha \in [0.25, 1]$ makes the method a little bit faster than for simply $\alpha = 0$. On the other hand, for $N_{\text{fine}} = 819$ and $\alpha = 2$, the method is slowly diverging. For $\alpha = 10$, 100, the convergence is faster for small meshes, but it gets worse for larger meshes. So by inspecting the only smoother, we can assume the multigrid method will not be affected for $\alpha \in [0, 1]$.
We use the multigrid method with a coarse mesh of $N_{\text{coarse}} = 819$ dofs which we refine, with a bisection algorithm, up to $N_{\text{fine}} = 421294$ dofs. For each level, we do 5 pre-smoothing steps and 5 post-smoothing steps. On the coarsest level, we solve exactly. Again, we have repeated the experiments for different values of α. Similarly to Figure 8.12, Figure 8.13 shows a rate of convergence that is not much influenced by $\alpha \in [0, 1]$. In this case, optimal convergence is achieved, meaning that the number of iterations is independent of the number of dofs and of the number of levels used. On the other hand, for $\alpha > 1$, the rate of convergence is no more optimal and depends on the size of the problem. For too large values of α, the method does not even converge.

The impact of $\alpha > 0$ does not seem so necessary for a linear multigrid method that considers all the levels. However, if we inspect a 2-grid method as in Figure 8.14, its role becomes more and more important with the aggressivity of the coarsening. The best performance is attained for $\alpha = 1$. We recover a convergence rate that is almost independent of the dimension of the problem, even though only two levels are used. Thus, if the right value of α is chosen, the Robin boundary conditions can damp different frequency components of the error. As we move away from this value, the method becomes slower and slower and can also not converge. We finally understand that the choice of α is very important and could affect more complicated non-linear problems.

In Figure 8.15, we see how the norm of the residual behaves after each fine and coarse correction for a 2-grid method. This means we evaluate the global fine residual after the smoothing steps and after the coarse correction. Every time the latter is added to the current approximate solution, the norm of the residual increases. Indeed the equality constraints at the fine level are just projected onto the coarser space. Then on the coarse space, an exact coarse correction is computed and it is interpolated to the fine space. Such correction, however, just satisfies the coarse and not the fine constraints. This is why, after its addition to the current solution, we can see an increment of the norm of the residual. Nevertheless, at the same time, the coarse correction helps in the global communication process and, thanks to the post-smoothing steps, in accelerating the overall convergence. In conclusion, we can state that the parameter α does not only govern the communication among the subdomains, but it is also important for damping the error after the computation of coarse corrections that do not fully satisfy the fine constraints. In particular, the value of α is properly chosen for the convergence if $||\mathbf{r}_a||$ and $||\mathbf{r}_b||$ are comparable and the latter does not dominate the whole process.

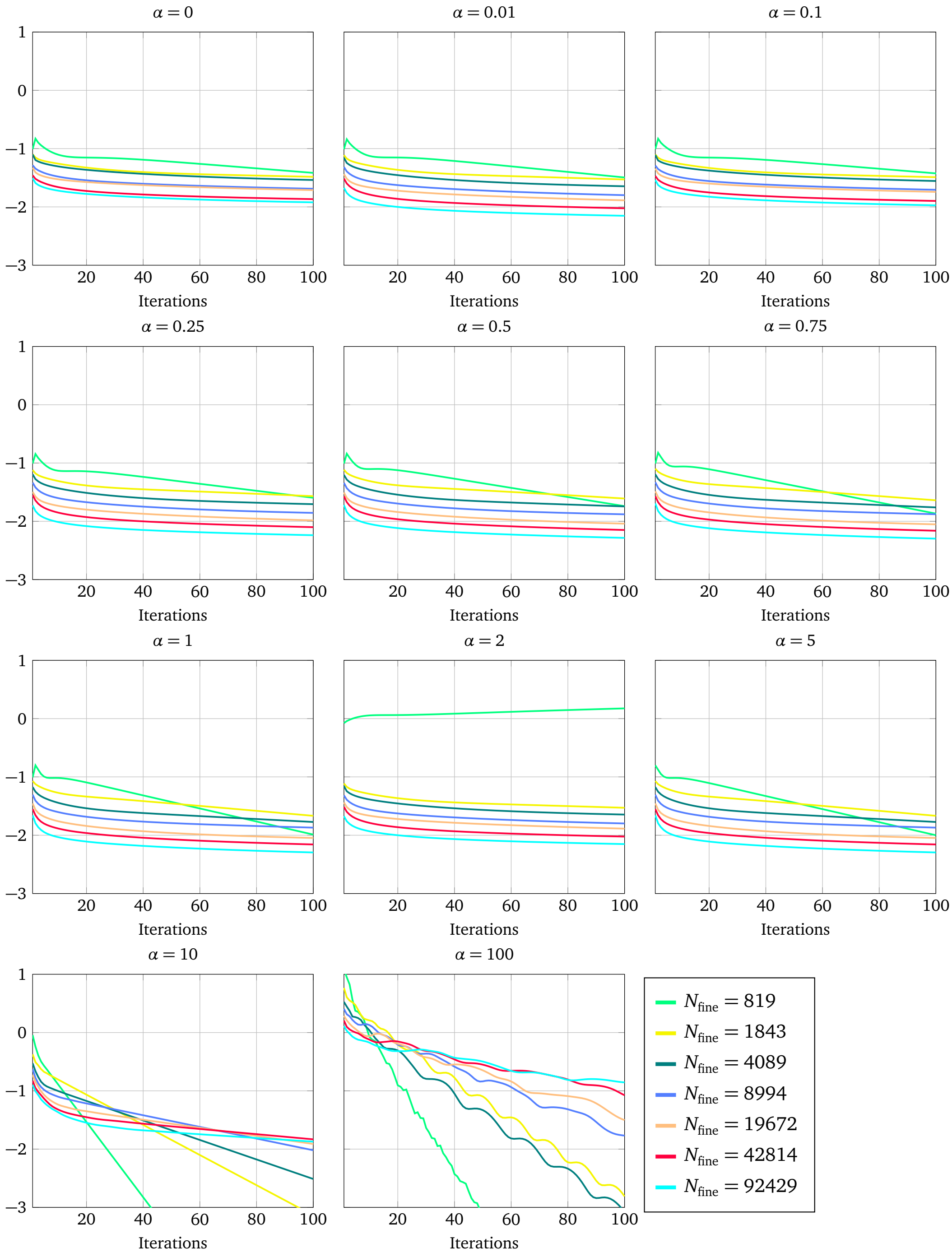


Figure 8.12. $\log_{10}$ of the Euclidean norm of the residual for the smoother of section 8.7 applied to the dual formulation for the problem in Figure 8.4. Parameters: $\mu = 1$, $\lambda = \infty$. The residuals have been computed after each smoothing step.

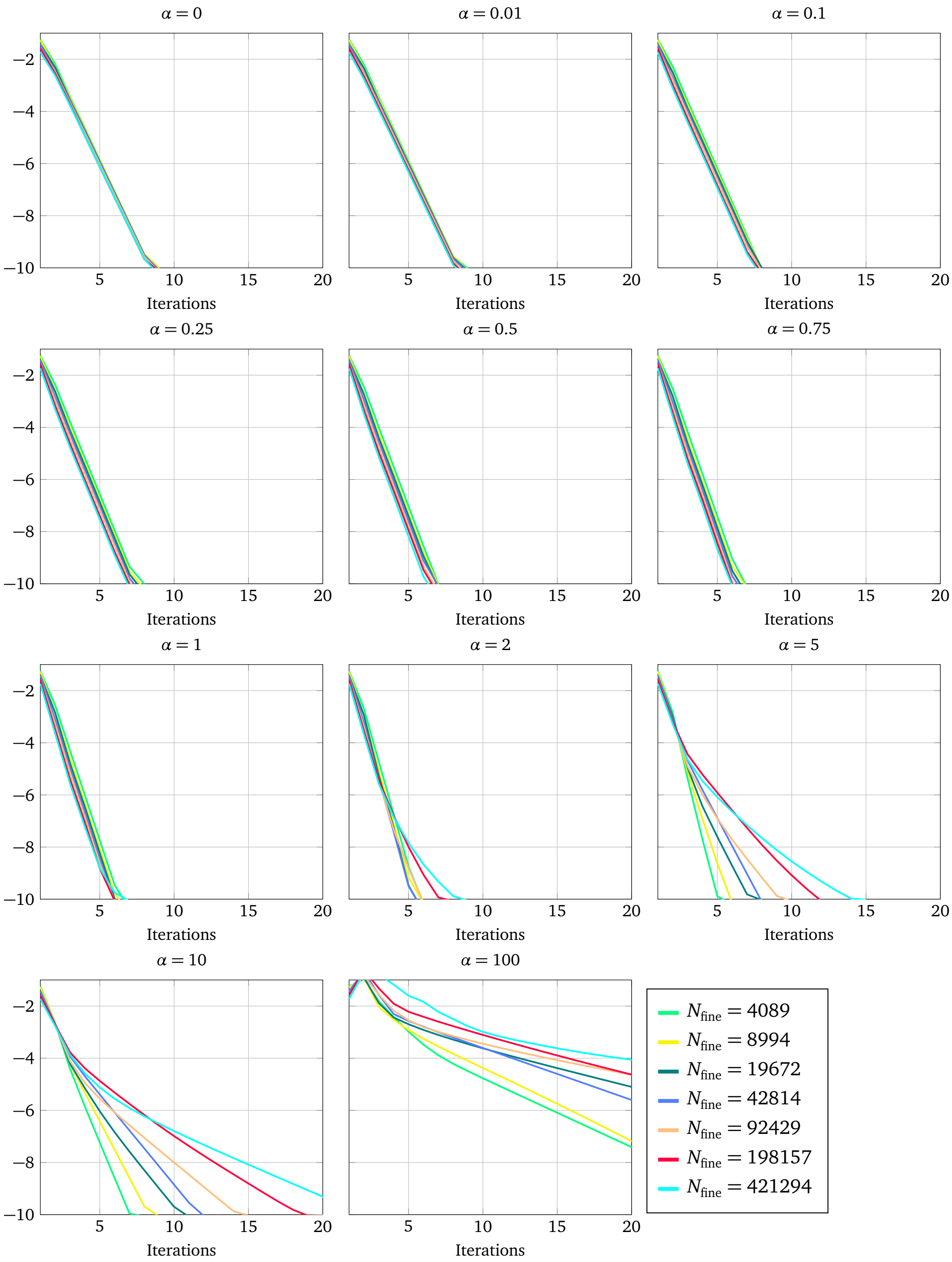


Figure 8.13. $\log_{10}$ of the Euclidean norm of the residual for the multigrid method applied to the dual formulation for the Cook's problem in Figure 8.4. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$. The coarsest level has dimension $N_{\text{coarse}} = 82$. Then bisection on each element is used to refine the mesh. The residuals have been computed after each V-cycle.

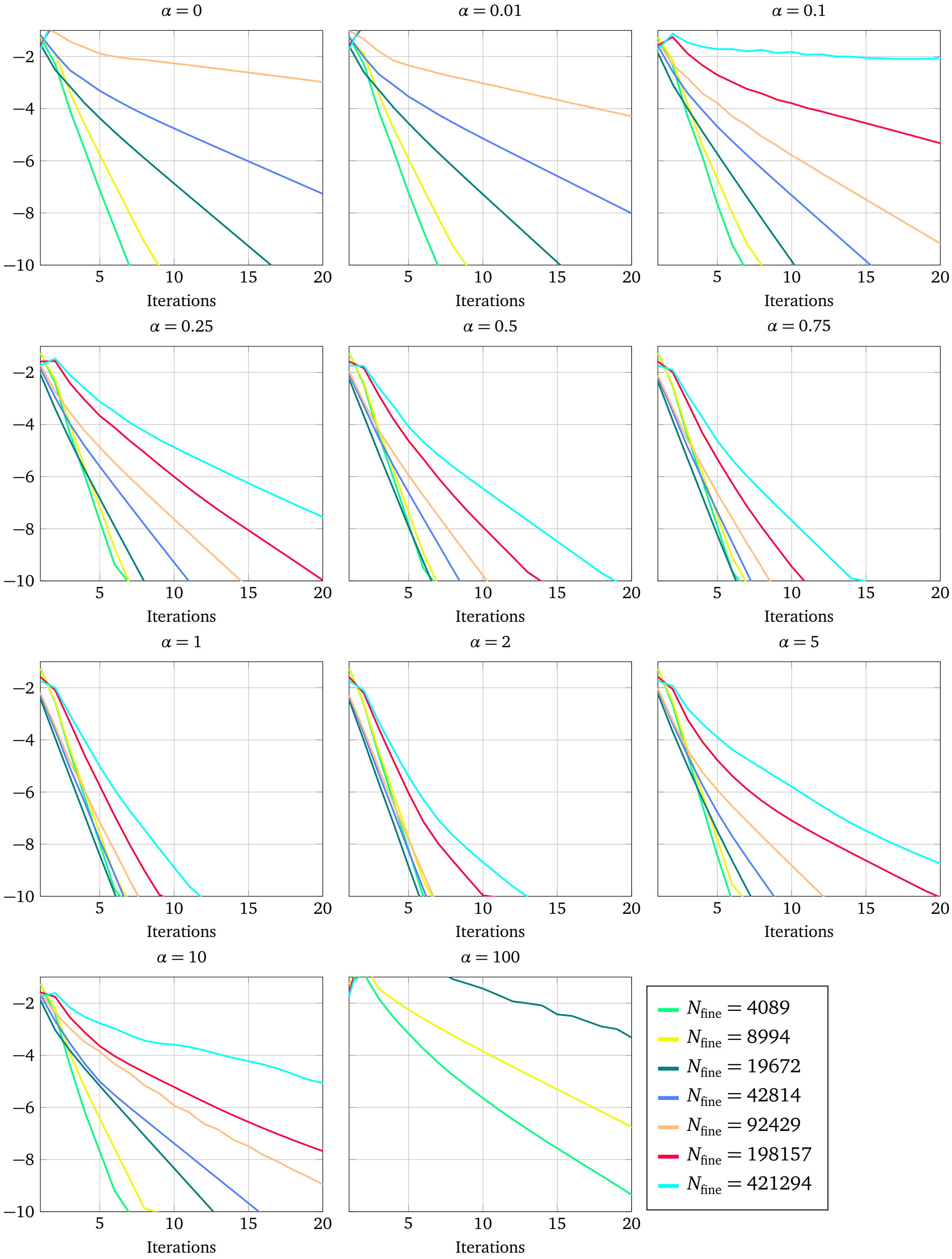


Figure 8.14. $\log_{10}$ of the Euclidean norm of the residual for the 2 grids method applied to the dual formulation for the Cook's problem in Figure 8.4. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$. The coarsest level has dimension $N_{\text{coarse}} = 819$. Then bisection on each element is used to refine the mesh, but only the finest level is considered. The residuals have been computed after each V-cycle.

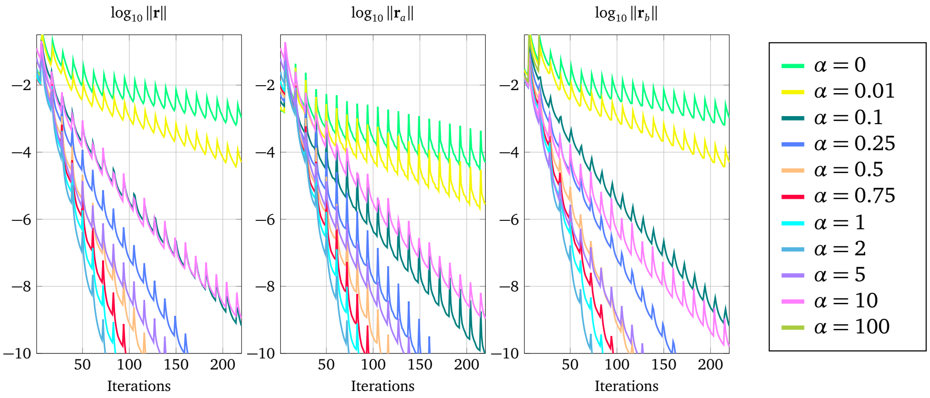
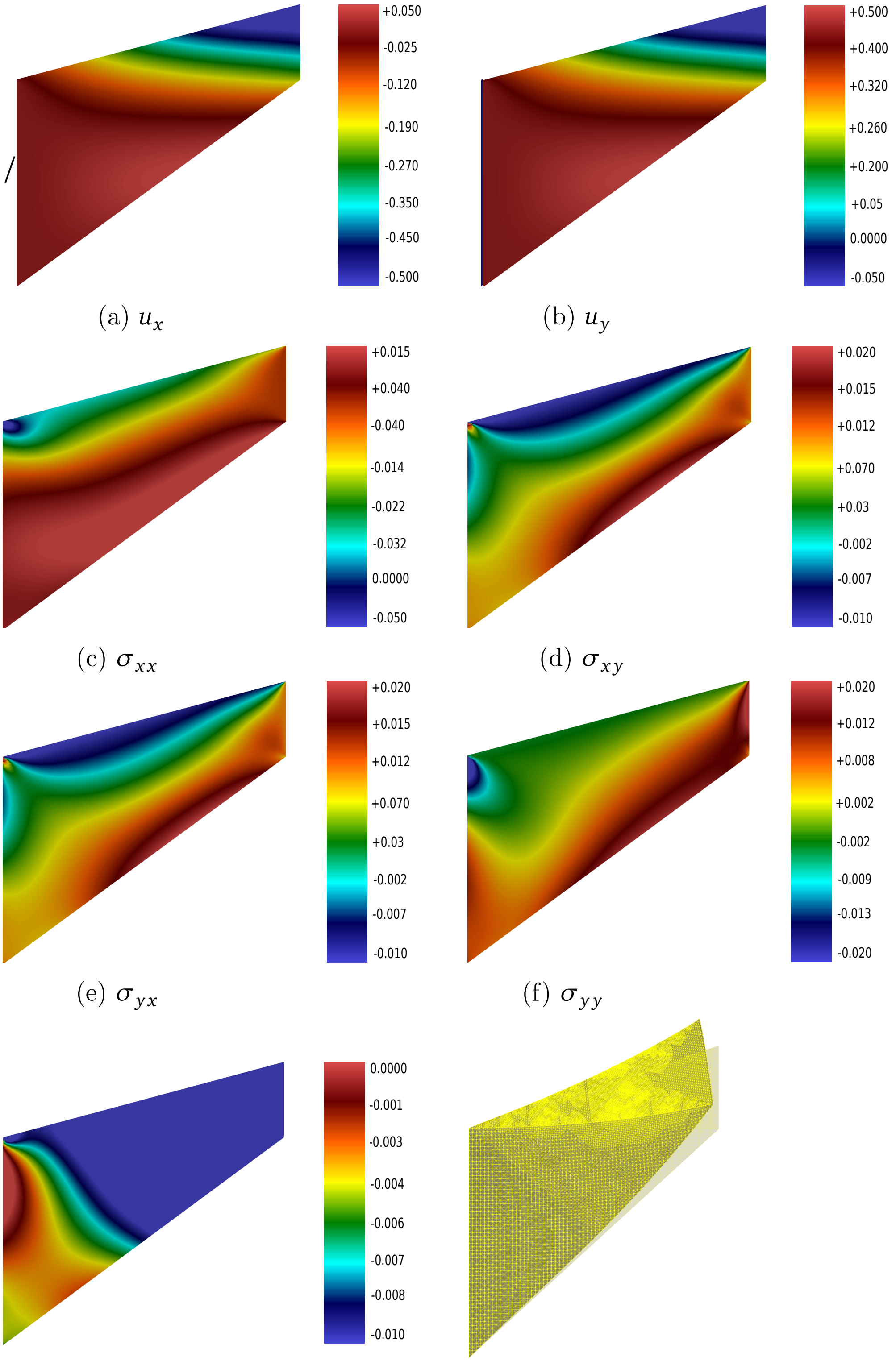


Figure 8.15. $\log_{10}$ of the Euclidean norm of the residual for the 2 grids method applied to the dual formulation for the Cook's problem in Figure 8.4. Parameters: $N_{\text{coarse}} = 819$, $N_{\text{fine}} = 92429$, $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$. The residuals have been computed after each smoothing step and each coarse correction addition to the current solution. In particular, after the coarse correction addition, the residual suddenly increases. This is due to the coarse representation of equality constraints. Nevertheless, the coarse correction still ensures a faster convergence than for the only smoother case of Figure 8.12.



(a) u_x

(b) u_y

(c) σ_{xx}

(d) σ_{xy}

(e) σ_{yx}

(f) σ_{yy}

(g) θ

(h) 5 times larger deformed mesh

Figure 8.16. Results for the Cook's problem in Figure 8.4. Parameters: $\mu = 1$, $\lambda = \infty$.

8.9.2 MMG for the face problem

In this section we want to examine how non-convexity of the geometry can eventually affect the multigrid convergence. Let us consider a square-shaped domain Ω with four different holes: two squares, one triangle and one rectangle, as depicted in Figure 8.17. On the bottom edge, we enforce a quadratic displacement in the y-component, i.e., $\mathbf{g}_D|_{\text{bottom}} = [0, 0.05x^2]^T$. On the triangle, we enforce zero displacement, i.e., $\mathbf{g}_D|_{\text{triangle}} = [0, 0]^T$. Everywhere else, we impose homogeneous Neumann conditions, $\mathbf{g}_N = [0, 0]^T$. The material has the following parameter: $\mu = 1$ and $\lambda = \infty$. Solutions are represented in Figure 8.21.

In Figure 8.18, the smoother as a stand-alone solver is examined for $\alpha = 0$, 0.01, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 2, 5, 10, 100. We can state that $\alpha = 0.01$ and $\alpha = 0.25$ give rise to the best convergence results. On the other hand, for increasing $\alpha > 1$ the convergence starts to deteriorate. The plots in Figure 8.19 are obtained using a multigrid method with a coarse mesh of $N_{\text{coarse}} = 769$ dofs which we refine, with a bisection algorithm, up to $N_{\text{fine}} = 234387$ dofs. For each level we do 5 pre-smoothing steps and 5 post-smoothing steps. On the coarsest level, we solve exactly. We have repeated the experiments for different values of $\alpha = 0$, 0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 2, 5, 10. Since we know from the analysis of the smoother that too large values of α are not a promising choice, we have removed $\alpha = 100$ and added $\alpha = 0.001$. Results are represented in Figure 8.19. In contrast to the Cook's membrane problem, the parameter α has to be chosen carefully. If $\alpha \leq 0.01$ results are always good, but for larger values of α, the rate of convergence is no more optimal and depends on the size of the problem. For too large α, the method does not even converge.

By inspecting the Figure 8.20, it is clear again that, for aggressive coarsening, the full Dirichlet case $\alpha = 0$ does not give rise to a convergent method. On the other hand, the optimal behaviour is reached for $\alpha = 0.1$. Indeed the rate of convergence is independent of the dimension of the problem. This means that, given a coarse level, independently of the fine level considered, the Robin boundary conditions with a proper value of α can damp all the frequency components of the error in between. By moving far away from this value, the convergence deteriorates or is lost.

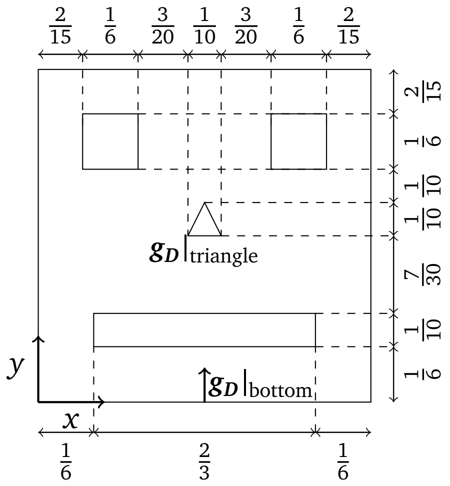


Figure 8.17. The face geometry is a square domain $[0,1] \times [0,1]$ with four holes, three rectangles and one triangle. The displacament condition on the bottom side is $\boldsymbol{g_D}|_{\text{bottom}} = [0, 0.05x^2]^T$. On the triangle-shaped hole $\boldsymbol{g_D}|_{\text{triangle}} = [0,0]^T$. Everywhere else $\boldsymbol{g_N} = [0,0]^T$.

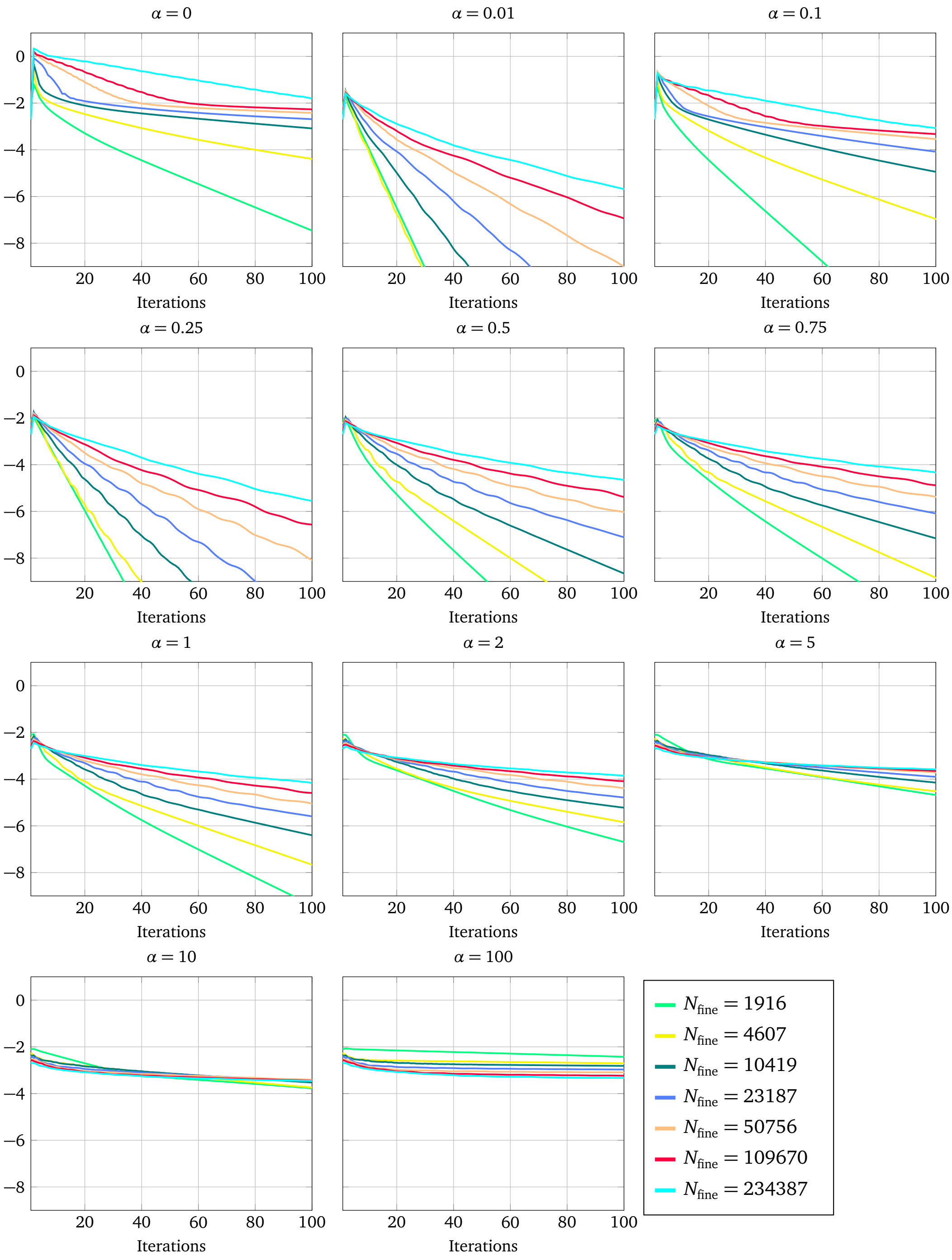


Figure 8.18. $\log_{10}$ of the Euclidean norm of the residual for the smoother of section 8.7 applied to the dual formulation for the problem in Figure 8.17. Parameters: $\mu = 1$, $\lambda = \infty$. The residuals have been computed after each smoothing step.

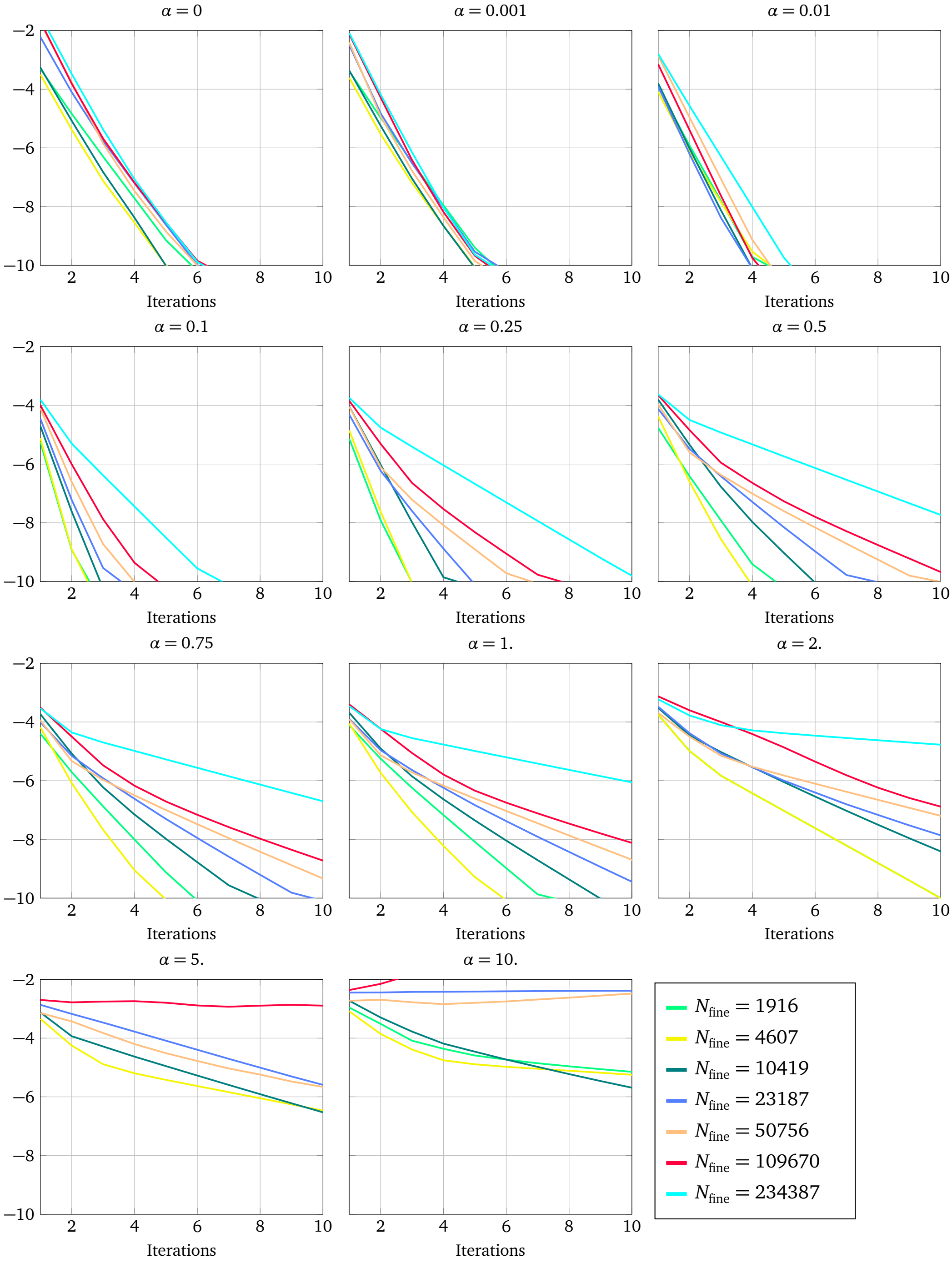


Figure 8.19. $\log_{10}$ of the Euclidean norm of the residual for the multigrid method applied to the dual formulation for the face problem in Figure 8.17. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$. The coarsest level has dimension $N_{\text{coarse}} = 769$. Then bisection on each element is used to refine the mesh. The residuals have been computed after each V-cycle.

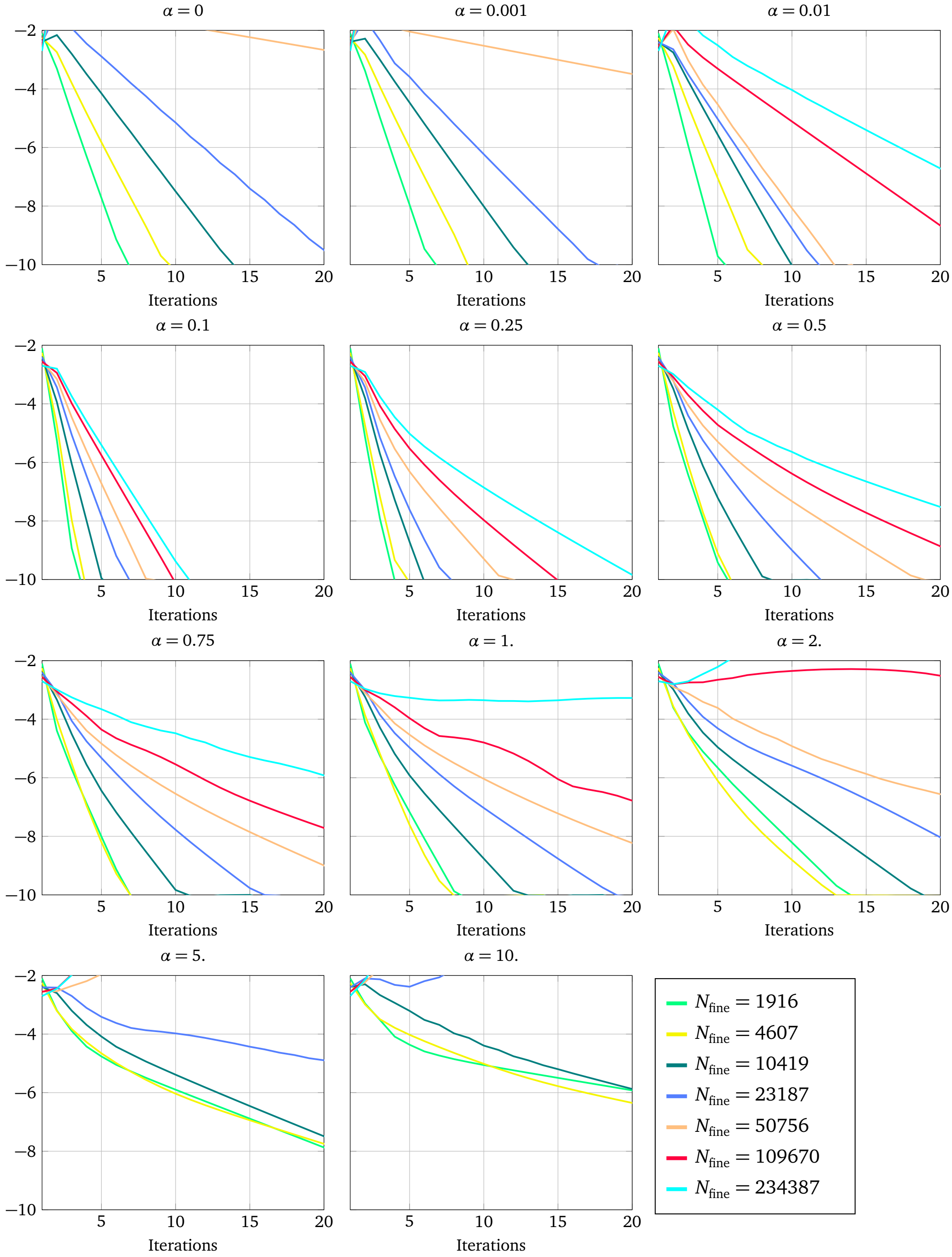
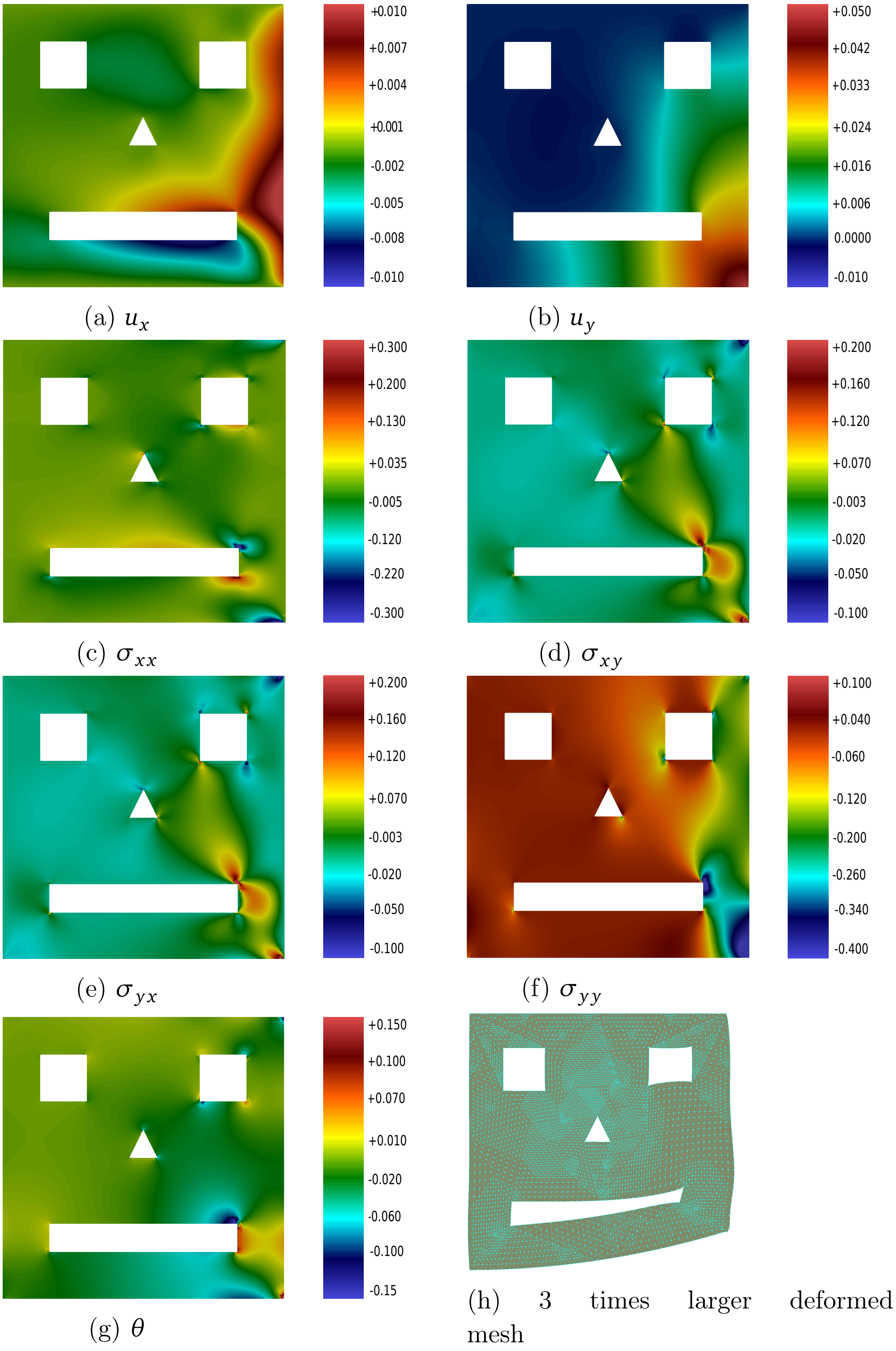


Figure 8.20. $\log_{10}$ of the Euclidean norm of the residual for the 2 grids method applied to the dual formulation for the face problem in Figure 8.17. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 5$. The coarsest level has dimension $N_{\text{coarse}} = 769$. Then bisection on each element is used to refine the mesh. The residuals have been computed after each V-cycle.



(a) u_x

(b) u_y

(c) σ_{xx}

(d) σ_{xy}

(e) σ_{yx}

(f) σ_{yy}

(g) θ

(h) 3 times larger deformed mesh

Figure 8.21. Results for the face problem. Parameters: $\mu = 1$, $\lambda = \infty$.

8.10 Numerical examples for the dual Signorini problem

In this section, the MMG method applied to the Signorini problem for the dual formulation of equation (5.52) is examined. We always consider incompressible materials with the following parameters $\mu = 1$ and $\lambda = \infty$. In section 8.10.1, the domain is a square, discretized by means of a uniform mesh, while the rigid obstacle is a semicircle. We first analyze the smoother and then the whole MMG. We will discover that the choice of the optimal parameter α is more strict than for the linear elastic case. For example, $\alpha = 0$ does not make the MMG method convergent. Of course, the parameter does not only depend on the continuous problem, but also on the mesh discretization: in section 8.10.2, a non uniform mesh for the same problem of section 8.10.1 will be used. In conclusion, in section 8.10.3, we study the effect of a non-convex geometry. In particular, the problem is given by is a square with a hole and with two rigid semicircular obstacles, one internal and one external.

The smoother introduced in section 8.7 solves for local *modified* non-linear problems that can be ill-posed for the active set method, meaning that the method enters in an infinite loop. In this case, the simulation can not end and it is not represented at all in the next figures. The same happens for simulations for which the MMG method did not converge or that were not able to arrive at 20 iterations in a reasonable amount of time of two days (time limit of the cluster used). The initial guess is always zero except for the boundary conditions, which are satisfied exactly. The residual is computed as in (8.2), but the dofs on the contact boundary which are active are neglected.

The results obtained by using the monotone restrictions in (6.34) gave rise to some problems of convergence, especially for the aggressive coarsening case. For this reason, we opted for using truncation on the fine level and for not enforcing inequality constraints on the coarser levels.

8.10.1 MMG for one body contact problem with a uniform mesh

We consider a square-shaped body $\Omega = [0,1] \times [0,1]$, with $\mu = 1$ and $\lambda = \infty$. We set free Neumann boundary conditions on the left and right sides, i.e., $\mathbf{g}_N|_{\text{left}} = \mathbf{g}_N|_{\text{right}} = [0,0]^T$. On the bottom we enforce a uniform vertical displacement $\mathbf{g}_D|_{\text{bottom}} = [0,0.01]^T$. The contact boundary Γ_C is on the top, where the obstacle is described by a semicircle of center $\mathbf{c} = [0.5,1.5]^T$ and radius $r = 0.5$. See Figure 8.22. All the tests are carried out with different values of α, i.e., $\alpha = 0$, 0.001, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1. Parameters $\alpha > 1$ have not been tried, because already in the linear case they make the

MMG method divergent.

We start by examining the smoother as a stand-alone solver up to 100 iterations. See Figure 8.24. In contrast to the linear case of Figure 8.12, now the smoother can be positively influenced by the parameter α. We see that $\alpha = 0.5$ gives very good results, for both coarse and fine problems. In particular, for sufficiently coarse meshes, the smoother is a real solver. Regarding the MMG method, we start from a coarse mesh of $N_{\text{coarse}} = 77$ dofs, which we refine with a bisection algorithm. We consider a MMG method with all the levels, doing 3 pre-smoothing steps and 3 post-smoothing steps at each level, except for the coarsest one, where an exact solver is used. Solutions of the problem can be found in Figure 8.23. The results are represented in Figure 8.25. The convergence is optimal for $\alpha = 0.1$, since it does not depend on the number of levels and the number of dofs. At least among the values used, we can state $\alpha_{\text{opt}} = 0.1$. The more α is chosen far away from this value, the more iterations are required for the MMG method to converge, and the more the number of levels has an important effect. Furthermore, if α is too far away, then the method can even not converge. In particular, it is interesting to notice that for too small values of α ($\alpha = 0,\ 0.001$) and for sufficiently fine meshes ($N_{\text{fine}} \geq 17057$), the MMG method does not converge. Thus, in contrast to the linear case, where $\alpha = 0$ for a multilevel strategy could be chosen with no problem, now the local Robin conditions are fundamental for the convergence of the method. If instead we use a two-level method, where the coarse space has $N_{\text{coarse}} = 1097$ dofs, qualitatively the results in Figure 8.26 are not so far away from the one in Figure 8.25. Also in this case $\alpha_{\text{opt}} = 0.1$ and for too small values of α, the MMG method does not converge. In particular, for fine meshes and large enough α, the method diverges as well.

If we focus on $\alpha_{\text{opt}} = 0.1$ for both Figure 8.25 and Figure 8.26, we see that the number of iterations is independent of the dimension of the fine problem and on the number of levels. Thus we can state that optimal convergence is achieved. Furthermore, it is interesting to notice that the same number of iterations is required for a multilevel strategy or a two levels strategy with aggressive coarsening. The difference is that, in this last case, all intermediate levels can be omitted and some computational time can be saved. Therefore, if α_{opt} is known, it is also possible to directly go for an aggressive coarsening.

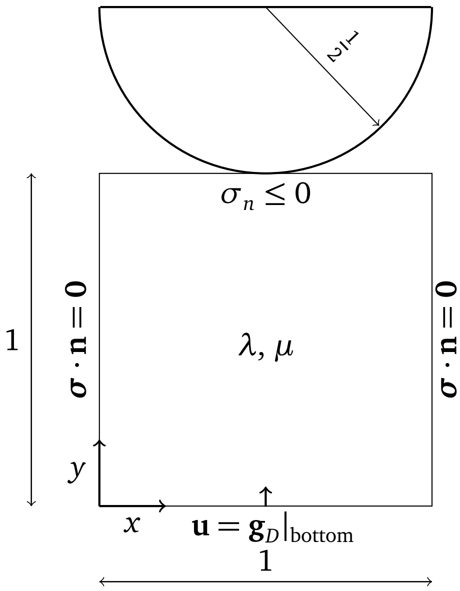
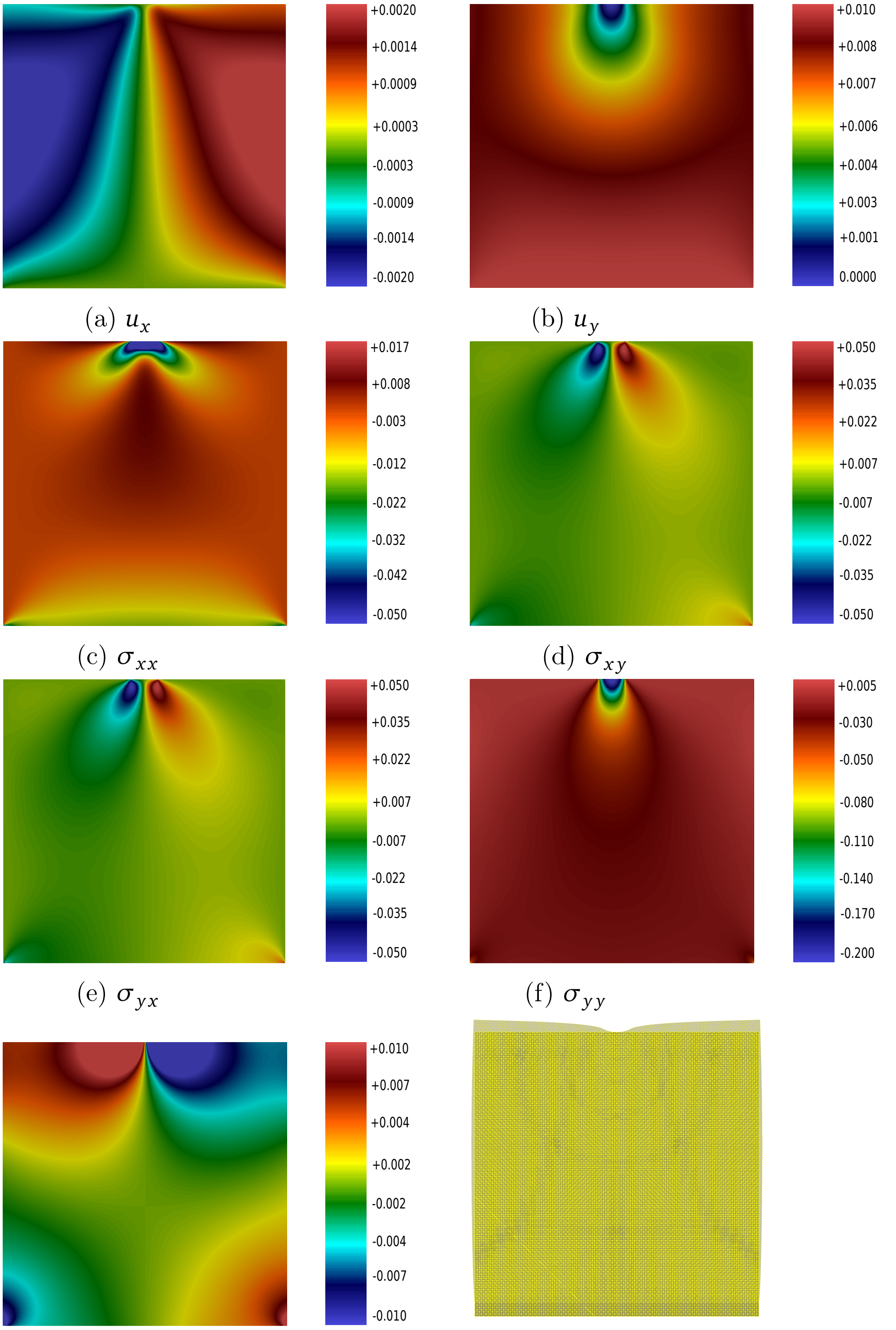

$\frac{1}{2}$
$\sigma_n \leq 0$
1
$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0}$
λ, μ
$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0}$
y
x
$\mathbf{u} = \mathbf{g}_D|_{\text{bottom}}$
1

Figure 8.22. One body contact problem with rigid body obstacle.



(a) u_x

(b) u_y

(c) σ_{xx}

(d) σ_{xy}

(e) σ_{yx}

(f) σ_{yy}

(g) ρ

(h) 5 times deformed mesh

Figure 8.23. Results for the problem in Figure 8.22 with a uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$.

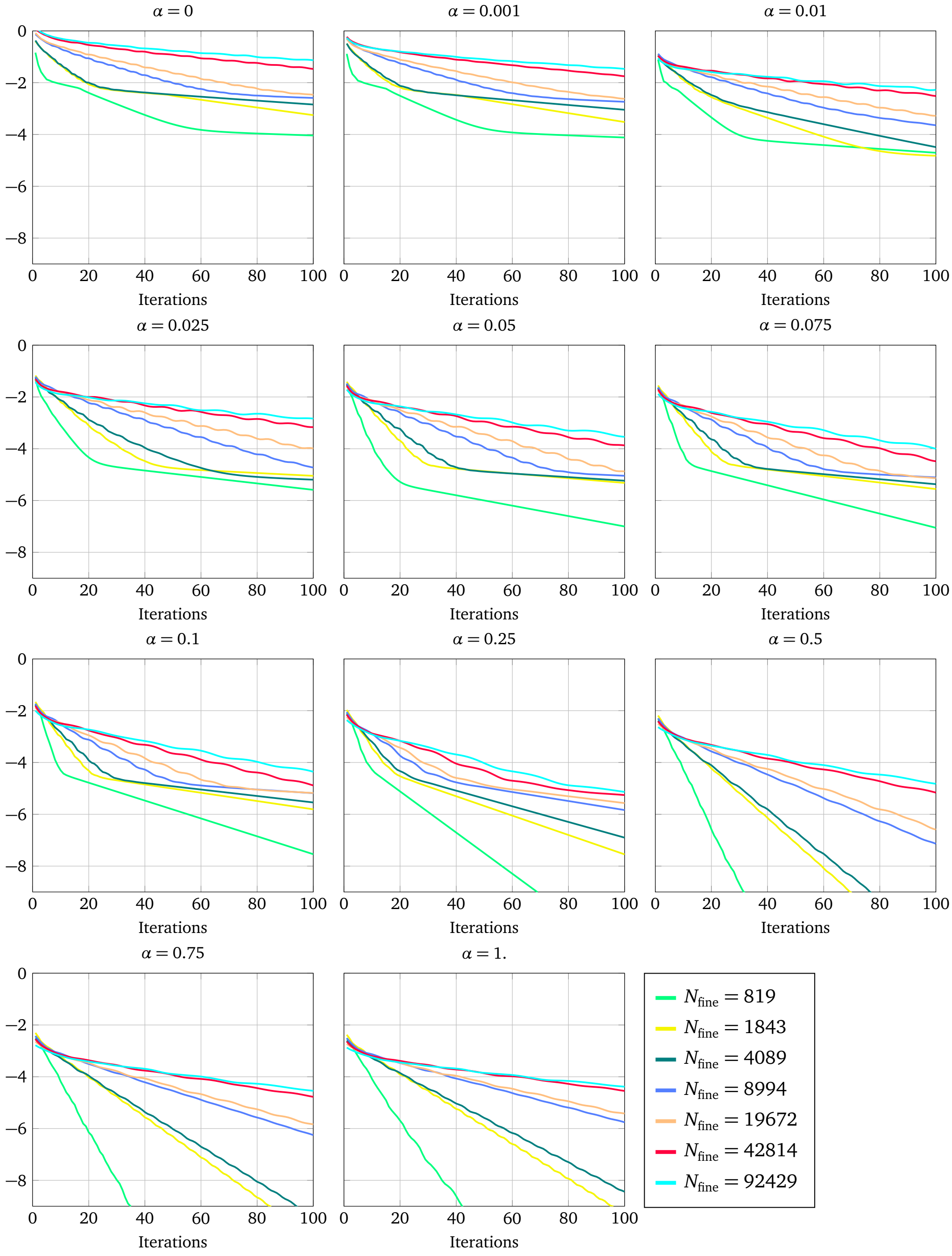


Figure 8.24. $\log_{10}$ of the Euclidean norm of the residual for the smoother applied to the dual formulation for the Signorini problem of Figure 8.22 with a uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 100$.

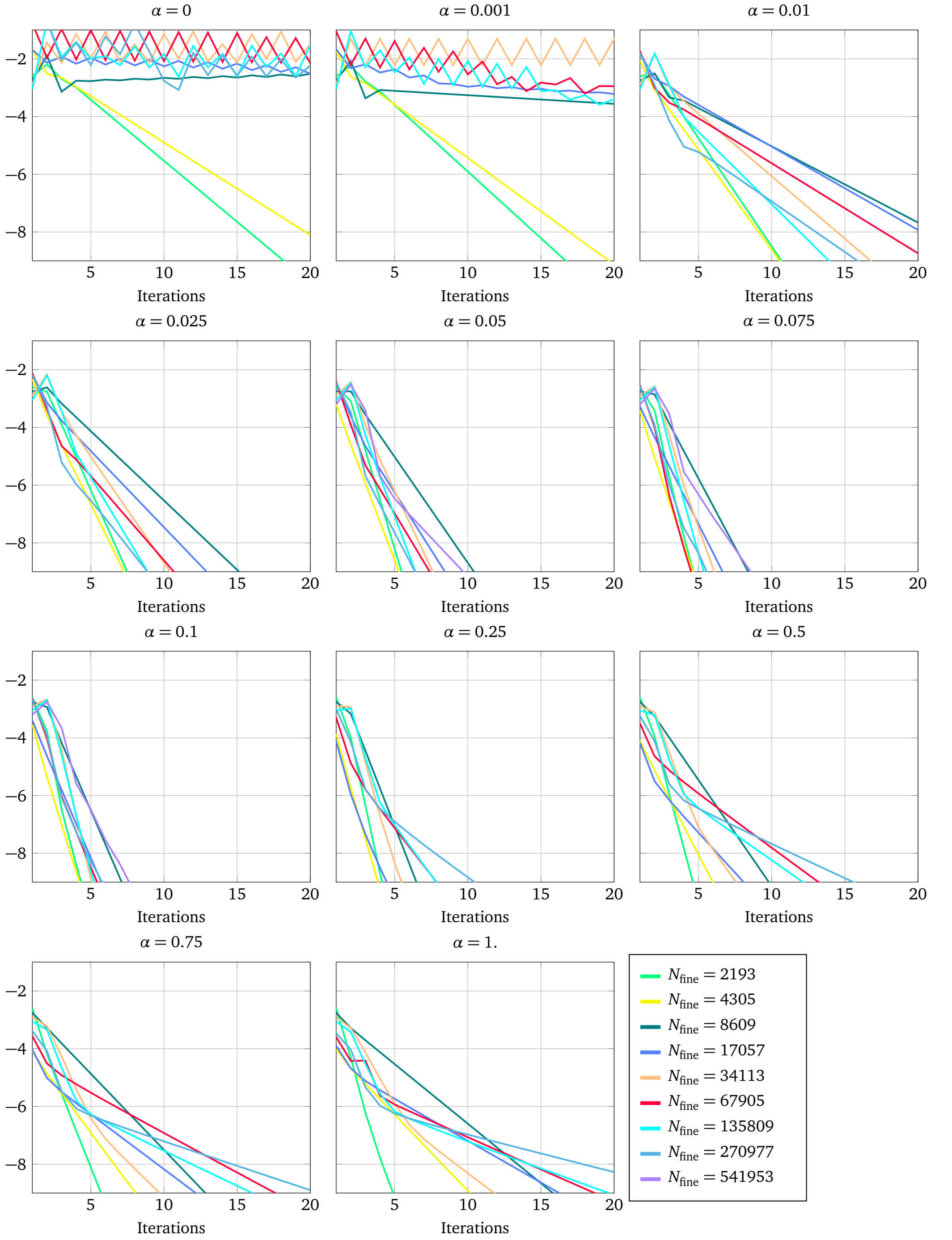


Figure 8.25. $\log_{10}$ of the Euclidean norm of the residual for the MMG method applied to the dual formulation for the Signorini problem of Figure 8.22 with a uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 3$. Coarse level dimension $N_{\text{coarse}} = 77$.

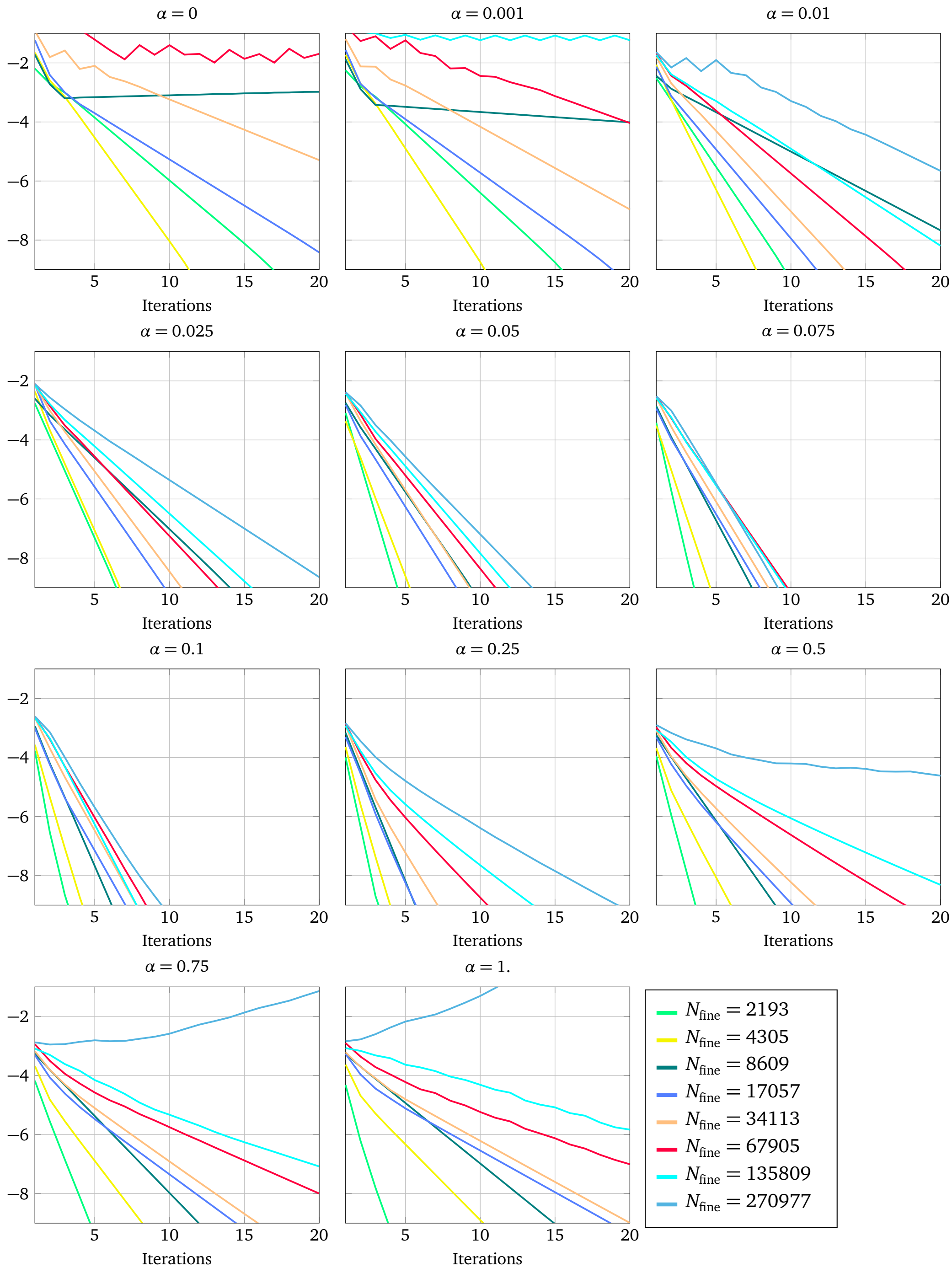


Figure 8.26. $\log_{10}$ of the Euclidean norm of the residual for the 2-levels MMG method applied to the dual formulation for the Signorini problem of Figure 8.22 with a uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 3$. Coarse level dimension $N_{\text{coarse}} = 1097$.

8.10.2 MMG for one body contact problem with a non uniform mesh

We now consider the same problem as in section 8.10.1, but with a non uniform mesh. The starting and the deformed meshes can be seen in Figure 8.27. In the contact zone, we have refined more the starting mesh. The results for 100 smoothing steps are represented in Figure 8.28. The convergence is more problematic. The finest case $N_{\text{fine}} = 209739$ is not convergent for $\alpha = 0.075, 0.25, 0.75$. For $\alpha = 0$ and $N_{\text{fine}} = 43368$, the method oscillates and the residual is not decreasing. Furthermore, now there is no α making the smoother solving even the coarser problems. We can expect the MMG method's convergence to be challenging as well, in this case. Among the values chosen, we can state that α_{opt} is in between $\alpha = 0.001$ and $\alpha = 0.01$, which is at least an order of magnitude less than the one we get for the uniform mesh. Thus all values of α far from α_{opt} and closer to zero or one are not good enough for the MMG method, which can diverge. Similar results, as in Figure 8.30, are obtained with a 2-levels MMG method. Indeed, even in this case, $\alpha_{\text{opt}} = 0.01$, with a number of iterations similar to the one of the multilevel case.

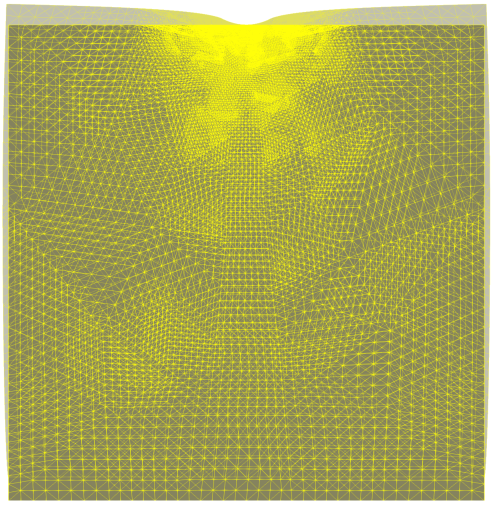


Figure 8.27. Undeformed and deformed non-uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$.

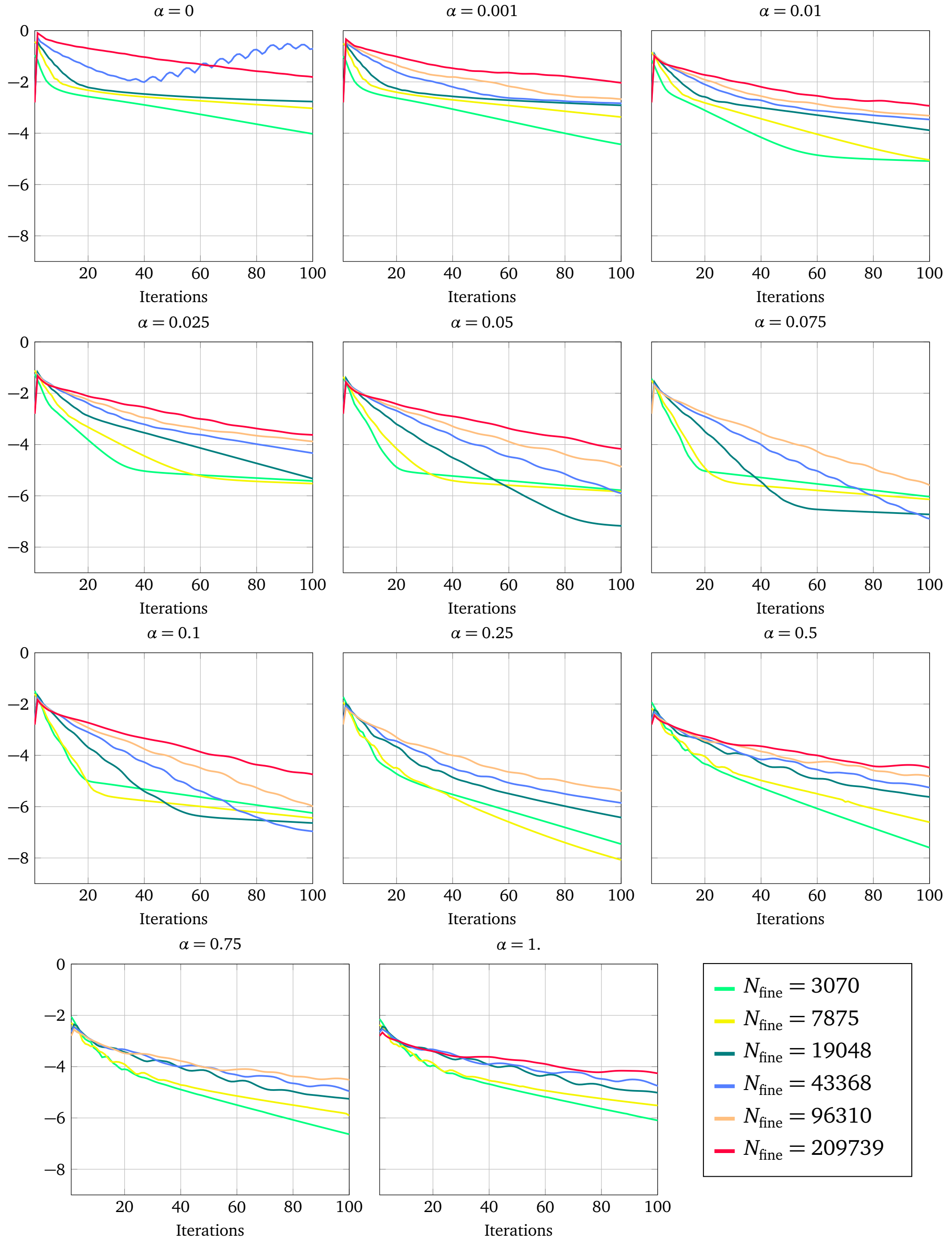


Figure 8.28. $\log_{10}$ of the Euclidean norm of the residual for the smoother applied to the dual formulation for the Signorini problem of Figure 8.22 with a non uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 100$.

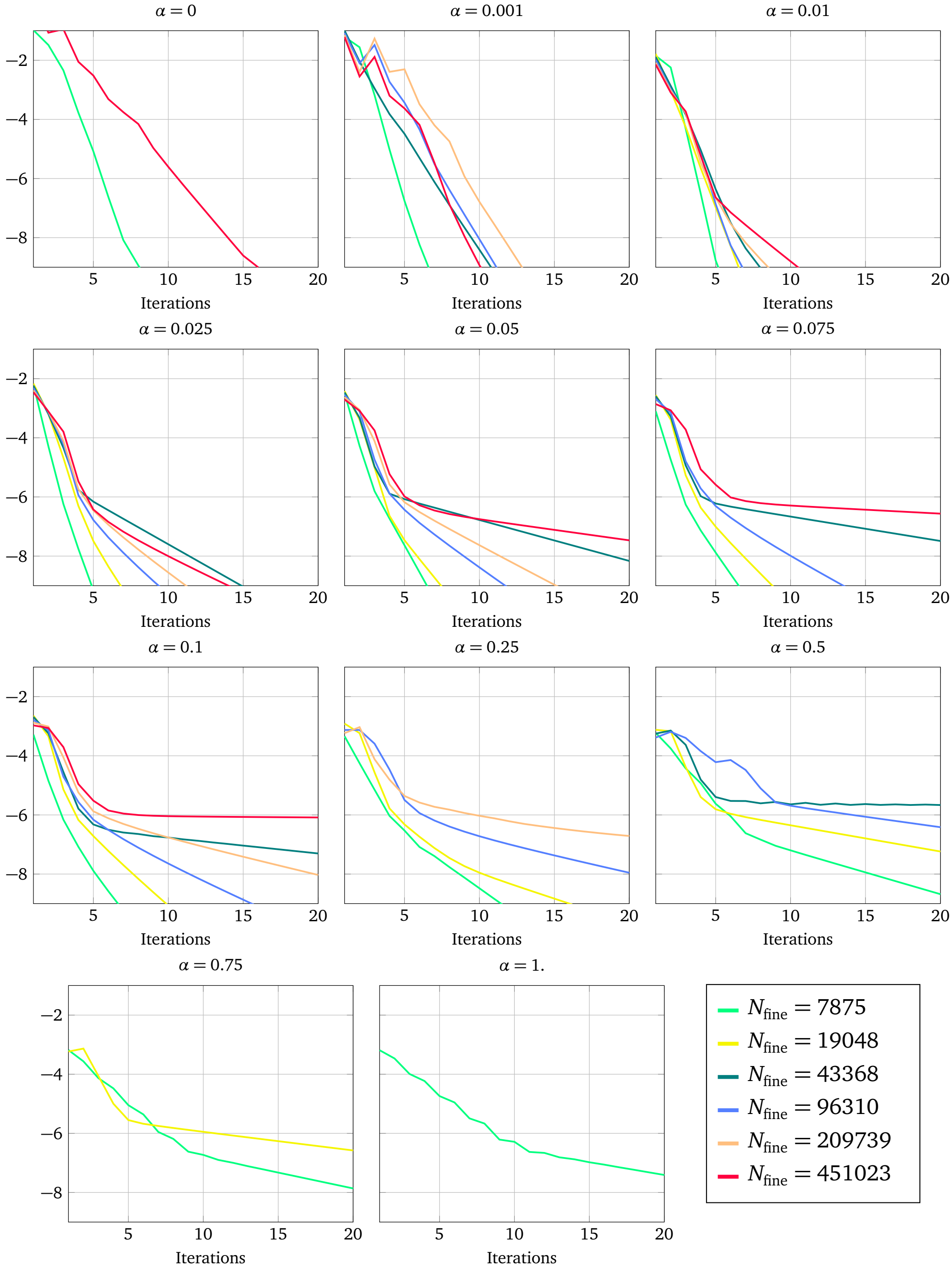


Figure 8.29. $\log_{10}$ of the Euclidean norm of the residual for the MMG method applied to the dual formulation for the Signorini problem of Figure 8.22 with a non uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 3$. Coarse level dimension $N_{\text{coarse}} = 3070$.

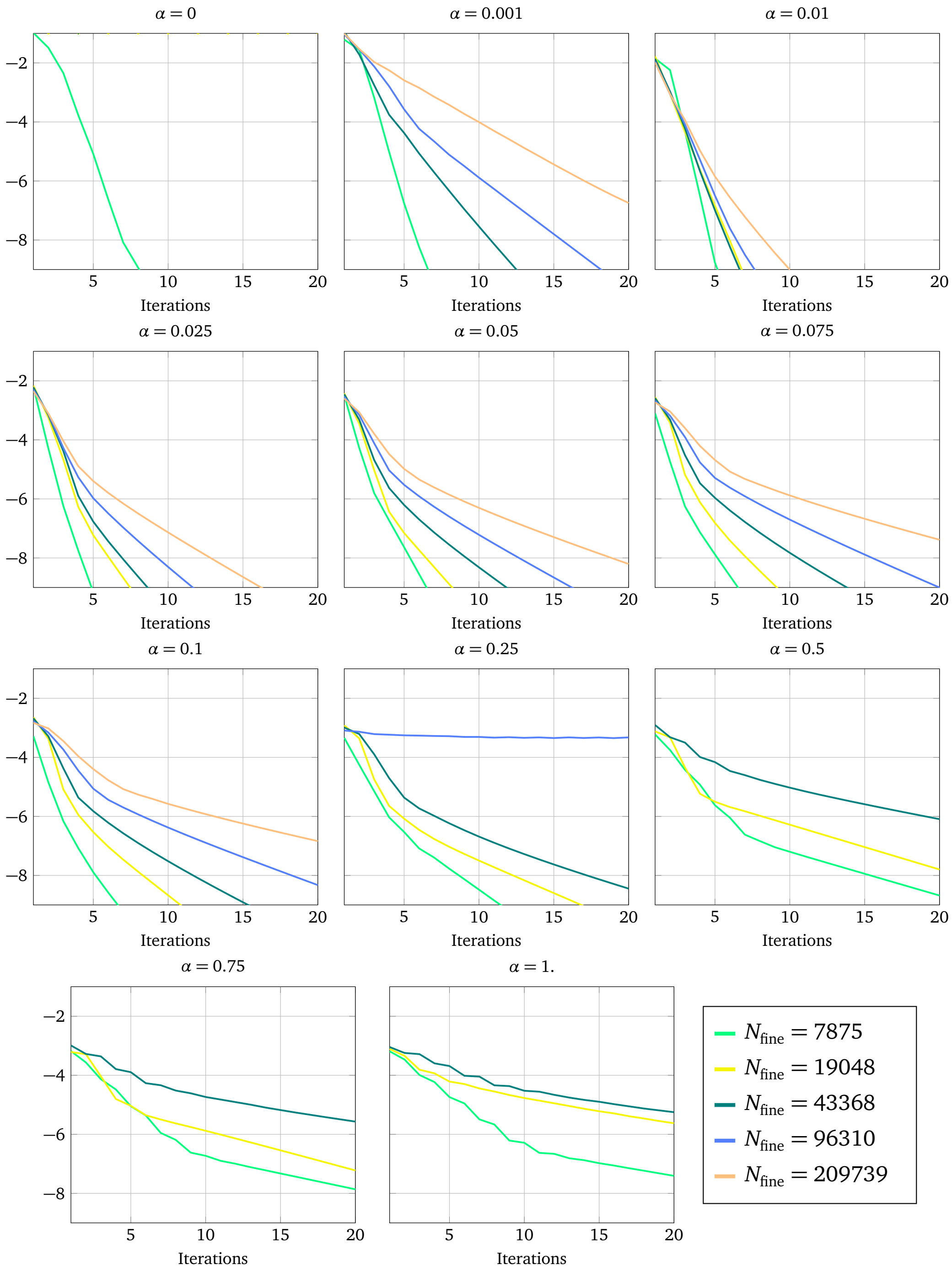
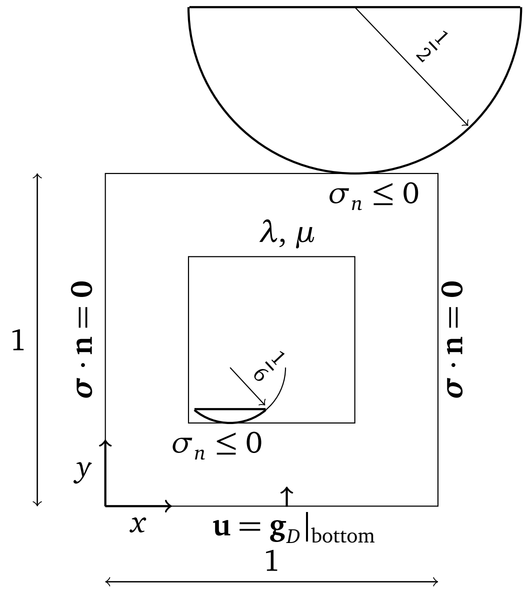


Figure 8.30. $\log_{10}$ of the Euclidean norm of the residual for the 2-levels MMG method applied to the dual formulation for the Signorini problem of Figure 8.22 with a non uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 3$. Coarse level dimension $N_{\text{coarse}} = 3070$.

8.10.3 MMG for one body with hole in contact with two rigid bodies

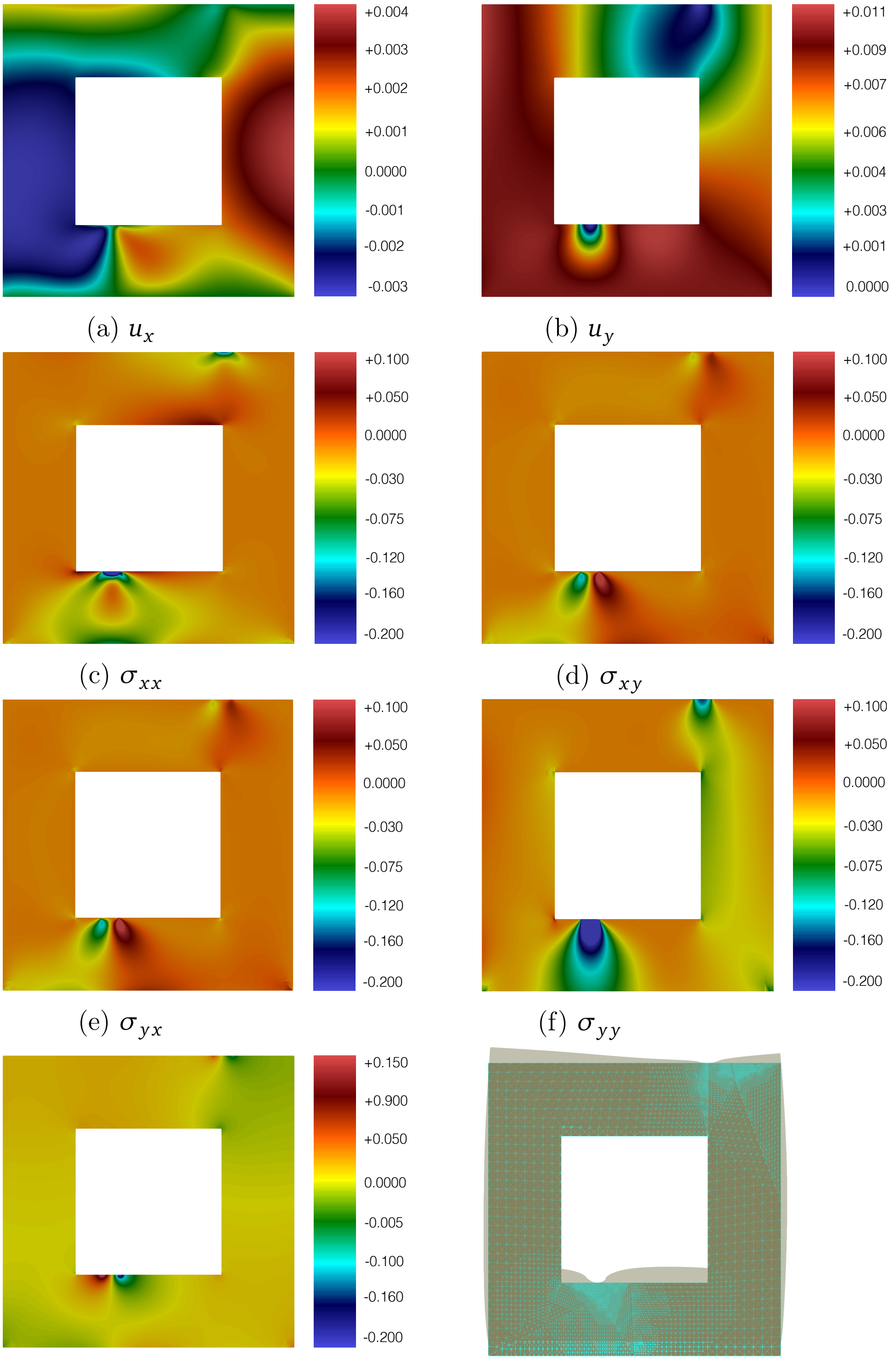
We consider a square-shaped body $\Omega = [0,1] \times [0,1]$, with $\mu = 1$ and $\lambda = \infty$, with a hole defined by $[0.25, 0.75] \times [0.25, 0.75]$. On the bottom we enforce a uniform vertical displacement $\mathbf{g}_D|_{\text{bottom}} = [0, 0.01]^T$. The contact boundary is on the top external edge and on the bottom internal edge, where the obstacles are described by half circles respectively of radius $r = 0.5$ and center $\mathbf{c}_1 = [0.75, 1.5]^T$ and radius $r = 1/6.0$ and center $\mathbf{c}_2 = [3/8, 5/12]^T$. Everywhere else we set free Neumann boundary conditions.

In Figure 8.33, the results for the smoother method are represented. The choice of α does not influence much the rate of convergence of the smoother as a stand-alone solver. However, from Figure 8.34, we can state that $\alpha_{\text{opt}} = 0.01$, while the more α is far away from this value, the worse the convergence becomes. In contrast to the previous examples, the two-levels MMG does not have the same value α_{opt} of the multilevel case. From Figure 8.35, we see that for $\alpha = 0.01$, the convergence is not obtained for fine enough meshes and neither for the coarsest case. Among the values tested, we can say that $\alpha_{\text{opt}} = 0.05$, at least for $N_{\text{fine}} = 269847$. So the more the problem becomes complex and the more α has to be chosen carefully.



$\frac{1}{2}$
$\sigma_n \leq 0$
λ, μ
$\frac{1}{6}$
$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0}$
$\boldsymbol{\sigma} \cdot \mathbf{n} = \mathbf{0}$
1
$\sigma_n \leq 0$
y
x
$\mathbf{u} = \mathbf{g}_D|_{\text{bottom}}$
1

Figure 8.31. Non-convex body with a hole in contact with two rigid body obstacles. Contact boundaries are fixed to be the top external and the bottom internal edges. There both $\sigma_n \leq 0$ and $\boldsymbol{\sigma} \cdot \boldsymbol{n} - \sigma_n \boldsymbol{n} = \mathbf{0}$ hold. On the bottom external edge a uniform vertical displacement is enforced, i.e., $\boldsymbol{u} = [0, 0.01]^T$. Everywhere else free Neumann boundary conditions $\boldsymbol{\sigma} \cdot \boldsymbol{n} = \mathbf{0}$ are imposed.



(a) u_x

(b) u_y

(c) σ_{xx}

(d) σ_{xy}

(e) σ_{yx}

(f) σ_{yy}

(g) ρ

(h) Deformed mesh

Figure 8.32. Results for the problem of Figure 8.31. Parameters: $\mu = 1$, $\lambda = \infty$.

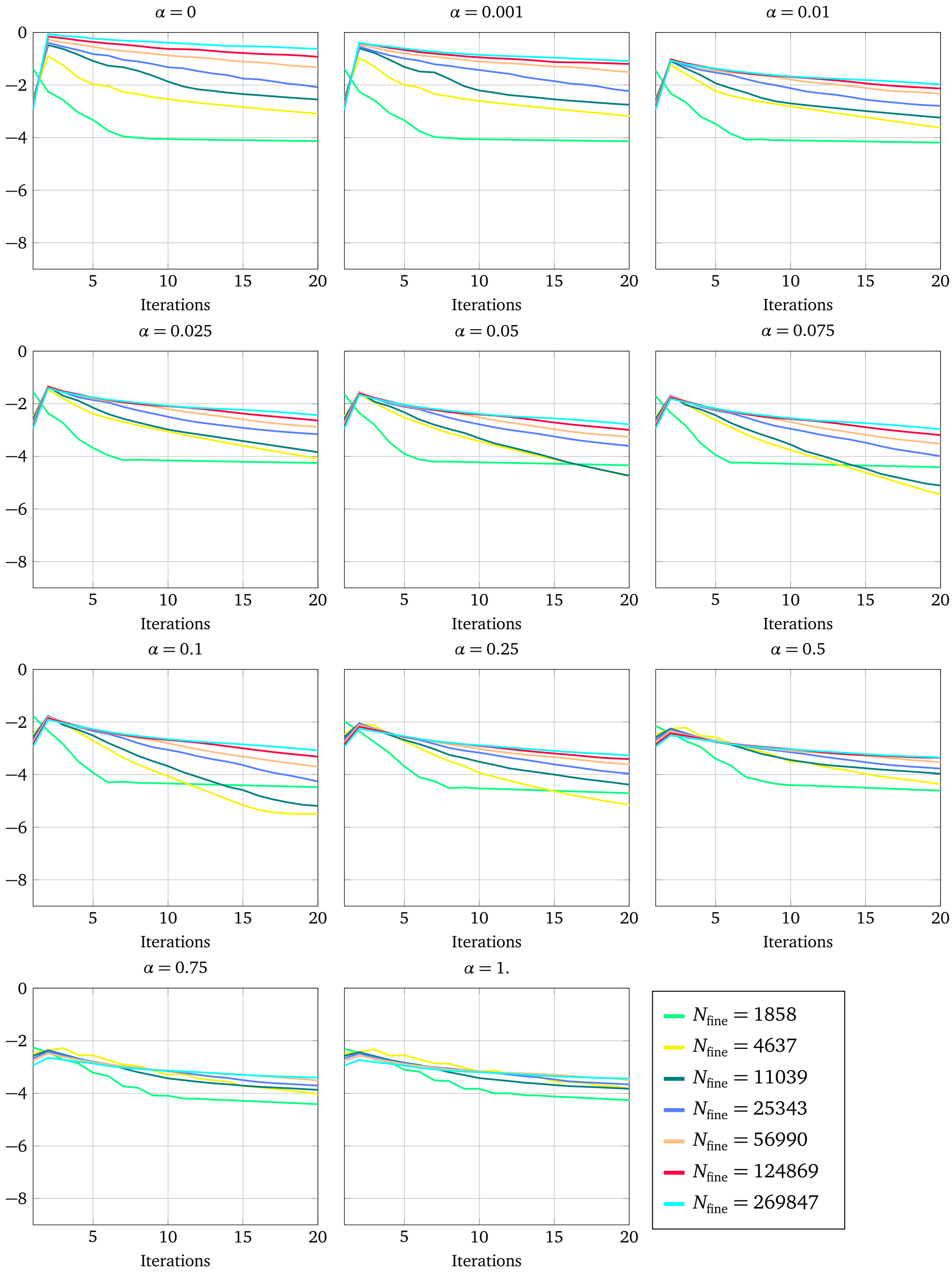


Figure 8.33. $\log_{10}$ of the Euclidean norm of the residual for the smoother applied to the dual formulation for the Signorini problem of Figure 8.31. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 100$.

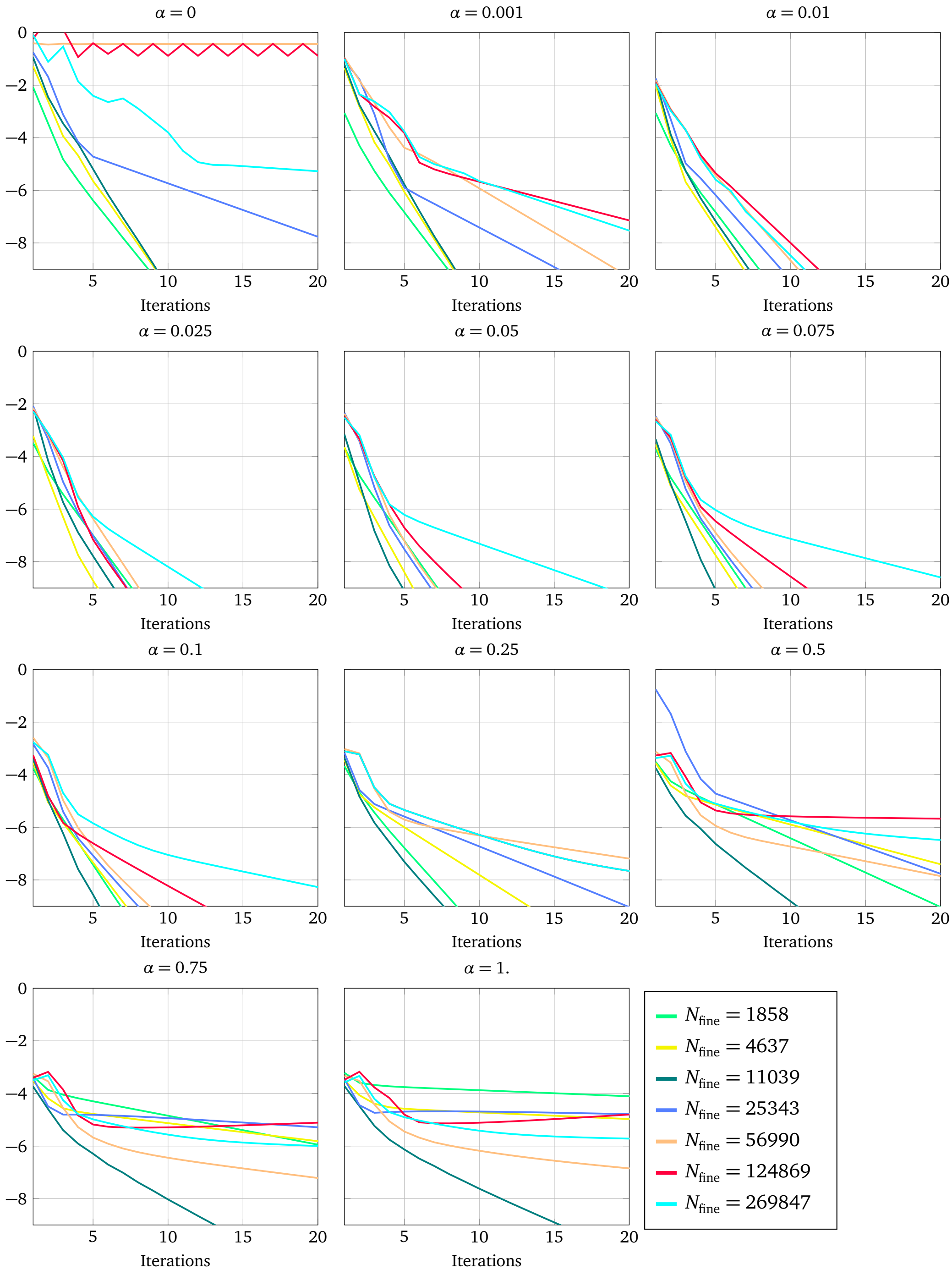


Figure 8.34. $\log_{10}$ of the Euclidean norm of the residual for the MMG method applied to the dual formulation for the Signorini problem of Figure 8.31. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 3$. Coarse level dimension $N_{\text{coarse}} = 579$.

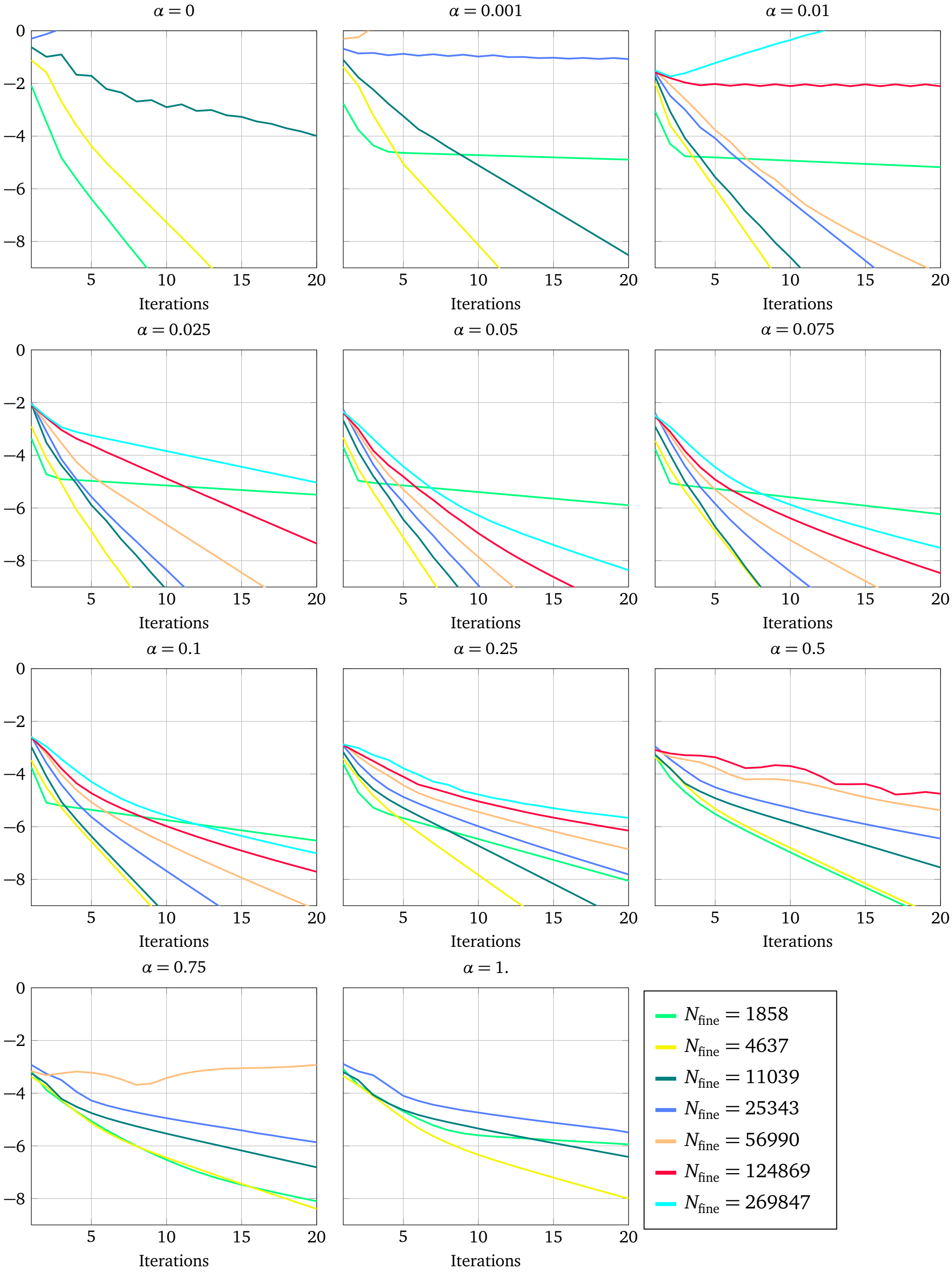
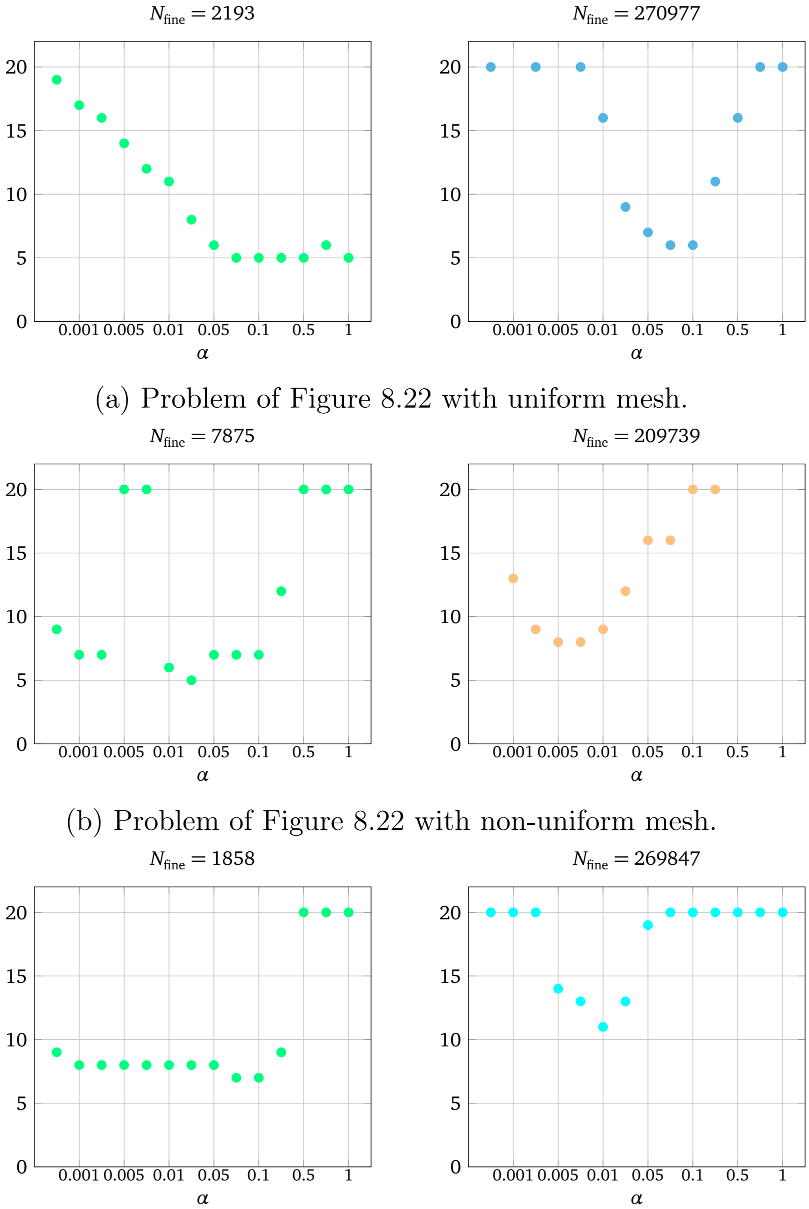


Figure 8.35. $\log_{10}$ of the Euclidean norm of the residual for the 2-levels MMG method applied to the dual formulation for the Signorini problem of Figure 8.31. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps $= 3$. Coarse level dimension $N_{\text{coarse}} = 579$.

8.11 Estimate of the Robin parameter

As we have seen in the previous sections, the choice of the parameter α can have a very important influence on the convergence of the MMG method. It is always possible to choose α to be constant, but then the optimal convergence property of the MMG method is not necessarily ensured. Depending on the problem, a different α should be chosen. In section 8.10.1, for a square domain with a uniform mesh, $\alpha_{\text{opt}} = 0.1$; in section 8.10.2, for a square domain with a non uniform mesh, $\alpha_{\text{opt}} = 0.01$; in section 8.10.3, for a square domain with a hole, $\alpha_{\text{opt}} = 0.01$. It is important to point out that, if a uniform refinement is used and all levels are considered, then α_{opt} is unchanged. However, for different coarse meshes or different problems, the value of α has to be chosen accordingly. Except for the fact we should take $\alpha \in [0,1]$, we have no many other useful clues. For sure, it is useful to notice the plots in Figure 8.36. Here the number of iterations of the MMG method, with a maximum of 20 iterations, is represented against the parameter α for the different problems we have examined so far. If a simulation is not convergent, then no point is depicted. For simplicity, we have decided to show the behavior for the smallest and the largest fine problems. By increasingly refine the problem, we can notice that the almost convex curves in those figures become sharper and sharper, with a minimum in α_{opt}. This is especially true for the finest cases, where a sort of parabola in the neighborhood of α_{opt} is recovered. So for example, in Figure 8.36c, for $N_{\text{fine}} = 1858$, the curve is almost flat, while for $N_{\text{fine}} = 269847$ we get a parabolic shape. Qualitatively this argument is true, but as we see in Figure 8.36b, the convexity can be broken. However, we can still imagine taking advantage of this information to dynamically determine α inside the MMG method.

In this section, we will examine different strategies for a dynamic choice of α. Since the corrections are computed locally on patches, the first idea is to optimize α on a given patch. However, as it will be clear in section 8.11.1, it is not possible to compute α on each patch independently from the other patches. Communication must be ensured and this can be guaranteed for a constant value of α. Thus, at least for a single smoothing step, α has to be chosen constant. The strategy in section 8.11.2 is to compare the results of two smoothing steps with two different values of α, i.e., α_{left} and α_{right}, and to choose the best one. In sections 8.11.3 and 8.11.4, a similar argument will be used to compare the residual decrease of V-cycles computed with different values of α.



(a) Problem of Figure 8.22 with uniform mesh.

(b) Problem of Figure 8.22 with non-uniform mesh.

(c) Problem of Figure 8.31.

Figure 8.36. Number of MMG iterations, with a maximum of 20, against α.

8.11.1 Random computation of α on each patch

Algorithm 9: RandomAlphaPatch

```
Result: $\alpha_{\text{rand}}$
Input: $\alpha_{\text{left}}, \alpha_{\text{right}}$
if Rand(0, 1) < 0.5 then
    $\alpha_{\text{rand}} \leftarrow \alpha_{\text{left}}$
end
else
    $\alpha_{\text{rand}} \leftarrow \alpha_{\text{right}}$
end
```

For the MMG method applied to the problem of Figure 8.22 with a uniform mesh, we can observe from Figure 8.25 that the optimal value of α is $\alpha_{\text{opt}} = 0.1$. This parameter is constant for the whole simulation, but it is interesting to study how its modification on each patch can influence the overall convergence. By choosing α far away from α_{opt}, the convergence should consequently be slowed down. Different strategies can be taken into consideration for the dynamic computation of α. If α is unknown, even in the linear elastic case, the local system of equation (8.17) becomes non-linear. Of course, now that the local solution vector is $[\mathbf{y}_{\text{ext}}, \mathbf{y}_{\text{int}}, \mathbf{z}_{\text{loc}}, \alpha]^T$, an additional equation for closing the problem is necessary. For example, we could minimize the global residual. To this aim, it would not be necessary to compute the whole residual. Indeed a local correction influences the residual only in the dofs of the patch and in the dofs of the elements which surround the patch. We can call this last set of elements the halo of the patch. Then the residual can be computed locally, taking into consideration the dofs on the patch and on its halo. However this approach, due to the same argument used for Figure 8.15, fails. Since the residual can only decrease, we can expect the method to converge, but such convergence is very slow. Indeed it is not important that locally the residual decreases. What really matters is that, on average, the residual decreases as much as possible. From this point of view, the value of α on a patch cannot be chosen independently from the values on the other patches.

Therefore, we will examine the case where we randomly choose α on each patch and show how this affects the convergence. For simplicity, we can make α assume two values α_{left} and α_{right} such that $\alpha_{\text{left}} < \alpha_{\text{opt}} < \alpha_{\text{right}}$. If the parameter α can be chosen independently from patch to patch, then we can expect a global convergence rate in between the ones used in Figure 8.25 for the simulations with constant $\alpha = \alpha_{\text{left}}, \alpha_{\text{right}}$. In particular, we will investigate the convergence behaviour if, on each patch of each level, we choose randomly between two values $\alpha_{\text{left}} = \alpha_{\text{opt}} - \epsilon$ and $\alpha_{\text{right}} = \alpha_{\text{opt}} + \epsilon$. See Algorithm 9, where Rand(0,1) is a function returning a random number in $[0,1]$. We use the values $\epsilon = 0.01, 0.02, \ldots$ and so on. Until $\epsilon = 0.07$, we get the same convergence and this is why in Figure 8.37 the only case $\epsilon = 0.01$ is represented. However for $\epsilon = 0.08$, the method does not converge well or does not convergence at all anymore. Nevertheless in Figure 8.25 , we see that for $\alpha = 0.01 < 0.02 = \alpha_{\text{opt}} - \epsilon$ and for $\alpha = 0.2 > 1.08 = \alpha_{\text{opt}} + \epsilon$, we still recover a convergent method. This fact suggests that the parameter α cannot be chosen on each patch independently from what happens on the other patches. Indeed the communication among different patches is fundamental in the transfer of the information. A local correction is good only if it is built so that the final combination of all local corrections makes the residual decrease. Then, at least on a single smoothing step, the value of α should be taken as constant.

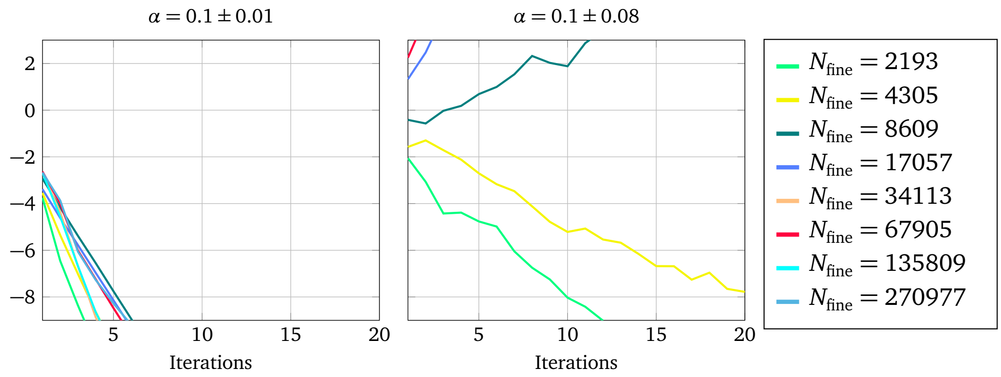


Figure 8.37. $\log_{10}$ of the Euclidean norm of the residual for the MMG method applied to the dual formulation for the Cook's problem in Figure 8.4 with a uniform mesh. Parameters: $\mu = 1$, $\lambda = \infty$, number of smoothing steps = 3. Coarse level dimension $N_{\text{coarse}} = 77$. We choose a random $\alpha \in \{\alpha_{\min}, \alpha_{\max}\}$ from patch to patch. For constant $\alpha = \alpha_{\min}, \alpha_{\max}$, we get convergence in maximum 15 iterations. If we randomly choose α, we do not get convergence.

8.11.2 Computation of α comparing two different smoothing steps

In section 8.11.1, it is shown that it is not a good idea to compute α on each patch, but it is instead better to take it constant at least for a whole smoothing step. In such a case, if α is constant, the subdomains on the given level can properly communicate. Then the issue is how to update α between different smoothing steps. First of all, it is necessary to choose which values α can assume. Instead of choosing $\alpha \in [0,1]$, we have decided to make α belong to a set of discrete values A defined as follows:

$$A := \{25 \cdot 10^{-k}, 50 \cdot 10^{-k}, 75 \cdot 10^{-k}, 100 \cdot 10^{-k}, \quad k \in \mathbb{N}\}\,. \tag{8.36}$$

This decision makes it easier the comparison with the cases of the MMG method with a constant α. Furthermore, since the values are sufficiently spaced, it avoids the incurrence of a local minimum, which was a major problem for strategies involving $\alpha \in [0,1]$. Of course, the set A is arbitrary and can be defined by increasing or decreasing the number of values for a certain order of magnitude. Such a number can also depend on the order of magnitude itself. For example for $k = 2$ and $k = 3$, it is possible to decide how many values we can have. Finally, we want to stress out that $k \in \mathbb{N}$ and that no constraint $k \geq 2$, to get $\alpha \in [0,1]$, is enforced. In this way, if the method that we use makes α increase over the unity, we know it is not doing its job and something has to be changed.

The initial value for α is $\alpha_0 \in A$. Then, at the iteration $k \geq 0$, given $\alpha_k \in A$, we need to find $\alpha_{\mathrm{k}+1} \in A$. We define $r_k = \log_{10} \|\mathbf{r}_k\|$ and $r_{k-1} = \log_{10} \|\mathbf{r}_{k-1}\|$. At the beginning of each V-cycle, we do a preprocessing. If the previous residual decrease was large enough, the same value for α is kept and thus $\alpha_{k+1} = \alpha_k$. This happens if $r_k - r_{k-1} < -\eta$, where η is a positive number that we choose to be $\eta = 0.6$. Otherwise we do two preprocessing smoothings steps with the immediately left and right values of α_{k} in A, i.e., α_{left} and α_{right}. Then we opt for the value of α which most minimizes the residual and the whole V-cycle is carried out with this value of α. See the pseudocode in Algorithm 10, where the function pos is used to determine the position of α in A and the function NormOfTheResidual($\tilde{\mathbf{y}}$) returns the norm of the residual corresponding to the current solution $\tilde{\mathbf{y}}$. The results are represented in Figure 8.1 for starting values $\alpha_0 = \{0.01,\ 0.025,\ 0.05,\ 0.075,\ 0.1,\ 0.25,\ 0.5,\ 0.75\}$. If the mesh is sufficiently refined, the method is not able to properly detect the correct value for α. For the fine meshes, α always tends to increase up to the value one or even more, meaning that also the smoothing step is not sufficient to determine the optimal α for the entire V-cycle. Similar results have been obtained by expanding this preprocessing strategy to the post-smoothing steps or to the coarser levels. Unfortunately no robustness of the algorithm with respect to the initial value α_0 has been recovered.

Algorithm 10: SingleSmoothingStepBasedComputation

```
Result: $\alpha_{\mathrm{k+1}}$, $\mathbf{y}_{k+1}$
Input: $\eta$, $r_{k-1}$, $r_k$, $\mathbf{y}_{\mathrm{k}}$, $\alpha_{\mathrm{k}}$, $A$
if $(r_k - r_{k-1}) < -\eta$ then
    $\alpha_{\mathrm{k+1}} \leftarrow \alpha_{\mathrm{k}}$
end
else
    $\mathbf{y}_{\mathrm{left}} \leftarrow \mathbf{y}_{\mathrm{k}}$
    $\mathbf{y}_{\mathrm{right}} \leftarrow \mathbf{y}_{\mathrm{k}}$
    $\alpha_{\mathrm{left}} \leftarrow A(\mathrm{pos}(\alpha_{\mathrm{k}}) - 1)$
    $\alpha_{\mathrm{right}} \leftarrow A(\mathrm{pos}(\alpha_{\mathrm{k}}) + 1)$
    $\mathbf{y}_{\mathrm{left}} \leftarrow$ Smoothing($\mathbf{y}_{\mathrm{left}}$, $\alpha_{\mathrm{left}}$ )
    $\mathbf{y}_{\mathrm{right}} \leftarrow$ Smoothing($\mathbf{y}_{\mathrm{right}}$, $\alpha_{\mathrm{right}}$)
    $r_{\mathrm{left}} \leftarrow \log_{10}$ NormOfTheResidual($\mathbf{y}_{\mathrm{left}}$)
    $r_{\mathrm{right}} \leftarrow \log_{10}$ NormOfTheResidual($\mathbf{y}_{\mathrm{right}}$)
    if $r_{left} < r_{right}$ then
        $\alpha_{\mathrm{k+1}} \leftarrow \alpha_{\mathrm{left}}$
        $\mathbf{y}_{\mathrm{k+1}} \leftarrow \mathbf{y}_{\mathrm{left}}$
    end
    else
        $\alpha_{\mathrm{k+1}} \leftarrow \alpha_{\mathrm{right}}$
        $\mathbf{y}_{\mathrm{k+1}} \leftarrow \mathbf{y}_{\mathrm{right}}$
    end
end
```

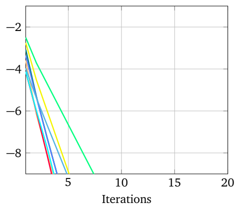
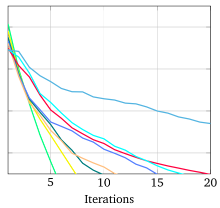
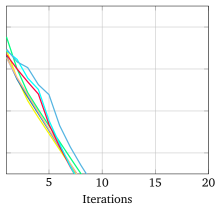
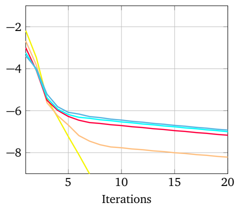
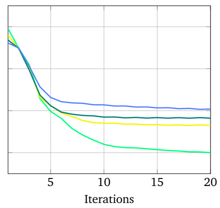
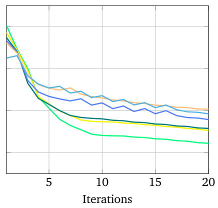
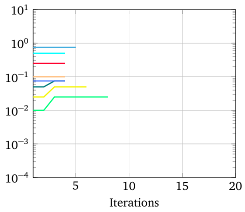
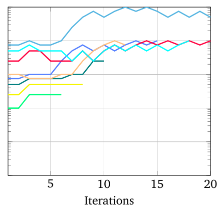
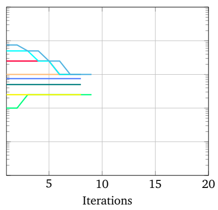
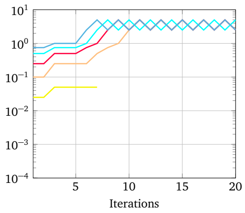
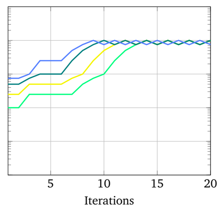
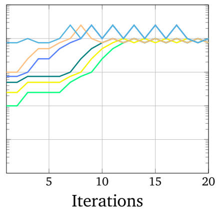
Legend		**Square domain** **uniform mesh**	**Square domain** **non-uniform mesh**	**Square domain** **with hole**
— $\alpha_0 = 0.01$ — $\alpha_0 = 0.025$ — $\alpha_0 = 0.05$ — $\alpha_0 = 0.075$ — $\alpha_0 = 0.1$ — $\alpha_0 = 0.25$ — $\alpha_0 = 0.5$ — $\alpha_0 = 0.75$	$\log_{10} \lVert \mathbf{r} \rVert$	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 
	α	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 

Table 8.1. Results related to Algorithm 10.

8.11.3 Increase or decrease α based on previous V-cycles

In this section and in section 8.11.4, α is modified based on the decrease of the residual norm after an entire V-cycle. Two main ideas of update of α are examined and, for each of them, α is fixed constant for at least one or two iterations.

The strategy of this section is identified by Algorithm 11 and Algorithm 12, where α is fixed respectively for at least one or two iterations. In this case, if the residual decrease is enough, we keep α_k. Otherwise, , depending on a parameter $s \in \{-1, 1\}$, we need to move to the left ($s = -1$) or to the right ($s = 1$) of α_k. Let us assume $\alpha_{k+1} = \alpha_{\text{left}}$. We keep this value until a sufficient decrease of the residual is attained. Assuming at iteration $k + 2$ the decrease is not large enough, we consider $\alpha_{k+2} = \alpha_{\text{right}}$. Then, if the residual decrease is better for α_{left}, we will update α as its left value and put $s = -1$. Otherwise we will update α as the right value of α_{right} and put $s = 1$.

In Algorithm 11, $\eta > 0$ represents the "sufficient decrease" condition; $s \in \{-1, 1\}$ is a parameter which states if it is necessary to move first to the left or to the right of α; $d \in \{0, 1, 2\}$ is a parameter such that: for $d = 0$ we use α_k and, depending on s, for $d = 1$ we use $\alpha_{\text{left}}/\alpha_{\text{right}}$ and for $d = 2$ we use $\alpha_{\text{right}}/\alpha_{\text{left}}$; r_k and α_k are respectively $\log_{10} \|\mathbf{r}\|$ and α at iteration k; the function NormOfTheResidual(α) returns the last norm of the residual related to the solution computed with the given α. The results for $\eta = 0.6$ are presented in Figure 8.2. Even though they are better than the ones in 8.1, we see that the convergence can be slow for the mesh with an internal hole. Furthermore, no convergence is attained for the non-uniform square mesh if α_0 is too far away from α_{opt}. Better results are obtained if α is fixed for at least two iterations, as it is seen in Figure 8.3. This fact suggests that modifying α at each iteration does not permit the method to stabilize. Maintaining α fixed helps in ensuring convergence, even though this can be slower because we require many more iterations for reaching or moving towards α_{opt}.

Algorithm 11: LeftRightOneStop

```
Result: $s$, $d$, $\alpha_{k+1}$, $\alpha_{\text{left}}$, $\alpha_{\text{right}}$
Input: $\eta$, $s$, $d$, $r_{k-1}$, $r_k$, $\alpha_k$, $\alpha_{\text{left}}$, $\alpha_{\text{right}}$, $A$
if $r_k - r_{k-1} < -\eta$ then
    $\alpha_{k+1} \leftarrow \alpha_k$
end
else
    if $d = 0$ then
        $\alpha_{\text{left}} \leftarrow A(\text{pos}(\alpha_k) - 1)$
        $\alpha_{\text{right}} \leftarrow A(\text{pos}(\alpha_k) + 1)$
        if $s = -1$ then
            $\alpha_{k+1} \leftarrow \alpha_{\text{left}}$
        end
        else
            $\alpha_{k+1} \leftarrow \alpha_{\text{right}}$
        end
        $d \leftarrow 1$
    end
    else if $d = 1$ then
        if $s = -1$ then
            $\alpha_{k+1} \leftarrow \alpha_{\text{right}}$
        end
        else
            $\alpha_{k+1} \leftarrow \alpha_{\text{left}}$
        end
        $d \leftarrow 2$
    end
    else if $d = 2$ then
        if $\log_{10} \mathit{NormOfTheResidual}(\alpha_{left}) < \log_{10} \mathit{NormOfTheResidual}(\alpha_{right})$
         then
            $s \leftarrow -1$
            $\alpha_{k+1} \leftarrow A(\text{pos}(\alpha_{\text{left}}) - 1)$
        end
        else
            $s \leftarrow 1$
            $\alpha_{k+1} \leftarrow A(\text{pos}(\alpha_{\text{right}}) + 1)$
        end
        $d \leftarrow 0$
    end
end
```

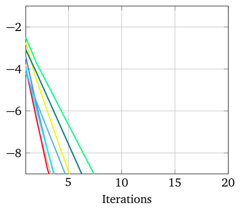
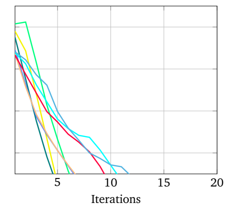
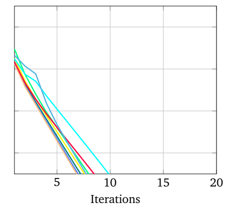
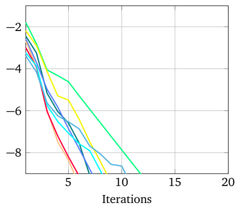
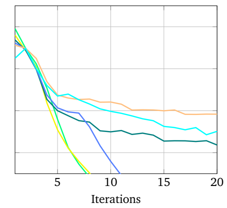
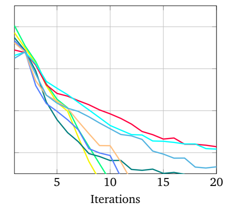
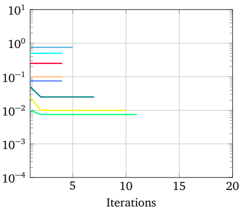
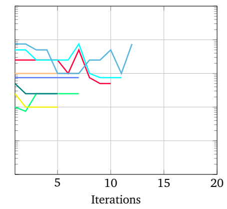
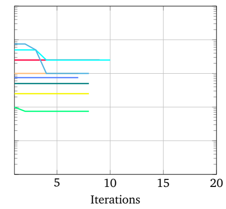
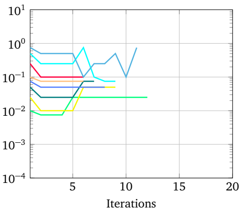
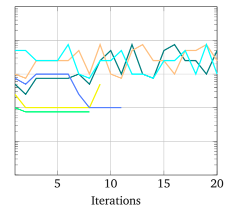
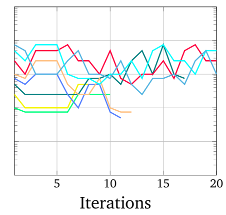
Legend		**Square domain** **uniform mesh**	**Square domain** **non-uniform mesh**	**Square domain** **with hole**
🟩 $\alpha_0 = 0.01$ 🟨 $\alpha_0 = 0.025$ 🟩 $\alpha_0 = 0.05$ 🟦 $\alpha_0 = 0.075$ 🟧 $\alpha_0 = 0.1$ 🟥 $\alpha_0 = 0.25$ 🟦 $\alpha_0 = 0.5$ 🟦 $\alpha_0 = 0.75$	$\log_{10}\|\mathbf{r}\|$	$N_{\text{fine}} = 2193$  	$N_{\text{fine}} = 7875$  	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$  	$N_{\text{fine}} = 209739$  	$N_{\text{fine}} = 269847$ 
	α	$N_{\text{fine}} = 2193$  	$N_{\text{fine}} = 7875$  	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$  	$N_{\text{fine}} = 209739$  	$N_{\text{fine}} = 269847$ 

Table 8.2. Results related to Algorithm 11.

Algorithm 12: LeftRightTwoStops

```
Result: $c$, $s$, $d$, $\alpha_{k+1}$, $\alpha_{\text{left}}$, $\alpha_{\text{right}}$
Input: $c$, $\eta$, $s$, $d$, $r_k$, $\alpha_k$ , $\alpha_{\text{left}}$, $\alpha_{\text{right}}$, $A$
if $r_k < -\eta$ or $c = 0$ then
    $\alpha_{k+1} \leftarrow \alpha_k$
    $c \leftarrow c + 1$
end
else if $c > 0$ then
    $c \leftarrow 0$
    if $d = 0$ then
        $\alpha_{\text{left}} \leftarrow A(\text{pos}(\alpha_k) - 1)$
        $\alpha_{\text{right}} \leftarrow A(\text{pos}(\alpha_k) + 1)$
        if $s = -1$ then
            $\alpha_{k+1} \leftarrow \alpha_{\text{left}}$
        end
        else
            $\alpha_{k+1} \leftarrow \alpha_{\text{right}}$
        end
        $d \leftarrow 1$
    end
    else if $d = 1$ then
        if $s = -1$ then
            $\alpha_{k+1} \leftarrow \alpha_{\text{right}}$
        end
        else
            $\alpha_{k+1} \leftarrow \alpha_{\text{left}}$
        end
        $d \leftarrow 2$
    end
    else if $d = 2$ then
        if $\log_{10} NormOfTheResidual(\alpha_{left}) < \log_{10} NormOfTheResidual(\alpha_{right})$
         then
            $s \leftarrow -1$
            $\alpha_{k+1} \leftarrow A(\text{pos}(\alpha_{\text{left}}) - 1)$
        end
        else
            $s \leftarrow 1$
            $\alpha_{k+1} \leftarrow A(\text{pos}(\alpha_{\text{right}}) + 1)$
        end
        $d \leftarrow 0$
    end
end
```

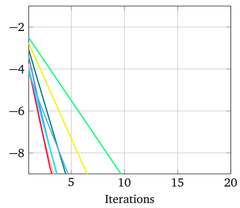
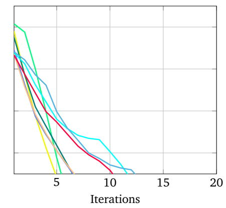
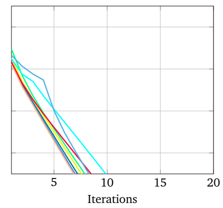
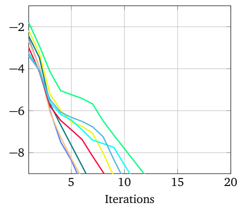
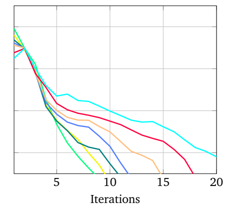
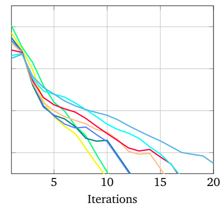
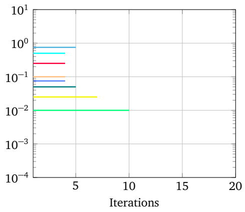
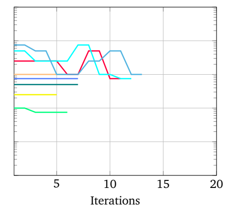
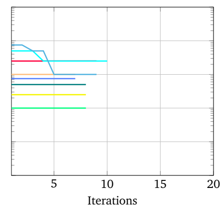
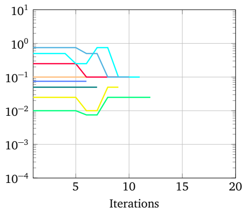
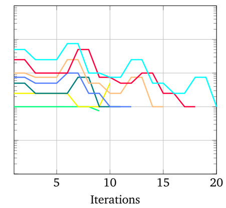
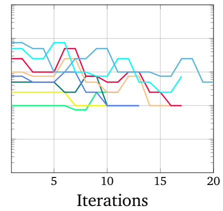
Legend		Square domain uniform mesh	Square domain non-uniform mesh	Square domain with hole
— $\alpha_0 = 0.01$ — $\alpha_0 = 0.025$ — $\alpha_0 = 0.05$ — $\alpha_0 = 0.075$ — $\alpha_0 = 0.1$ — $\alpha_0 = 0.25$ — $\alpha_0 = 0.5$ — $\alpha_0 = 0.75$	$\log_{10} \|\mathbf{r}\|$	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 
	α	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 

Table 8.3. Results related to Algorithm 12.

8.11.4 Dynamic reduction of α

In this section we present Algorithm 13 and Algorithm 14, where α is fixed respectively for at least one or two iterations. First of all, a sufficiently large α_0 needs to be used. Then, if the residual decrease is not enough, we move to its α_{left} value. And we continue in this way towards the minimum of Figure 8.36. Since we aim for a sufficient decrease of the residual, we avoid local minima. However, we need to not ask for a too-large decrease of the residual η, which maybe is impossible to reach, since this could produce an always decreasing α.

Given $\eta = 0.6$, results for Algorithm 13 can be found in Figure 8.4. We see that fast convergence is obtained in all cases if the value α_0 is such that $\alpha_0 > \alpha_{\text{opt}}$ and if it is not too large. Good convergence results, but with a slower rate are obtained for Algorithm 14. See Figure 8.5. Since in section 8.11.3, α could be increased or decreased, it was more difficult to reach its stabilization and for this reason, two steps were helpful. Now, since we only decrease α, the method is more stable and, to get good results, we can fix α for a single iteration. We can state that, if α_0 is not too small, Algorithm 13 is quite robust for the problems examined. In conclusion, the combination of Algorithm 13 and the MMG method gives rise to an optimal solver for the Signorini problem in the nearly-incompressible and incompressible cases.

Algorithm 13: ReductionOneStop

```
Result: $\alpha_{k+1}$
Input: $\eta$, $r_k$, $r_{k-1}$, $\alpha_k$, $A$
if $r_k - r_{k-1} < -\eta$ then
    $\alpha_{k+1} \leftarrow \alpha_k$
end
else
    $\alpha_{k+1} \leftarrow A(\text{pos}(\alpha_k) - 1)$
end
```

Algorithm 14: ReductionTwoStops

```
Result: $\alpha_{k+1}$, $c$
Input: $c$, $\eta$, $r_k$, $r_{k-1}$, $\alpha_k$, $A$
if $r_k - r_{k-1} < -\eta$ or $c = 0$ then
    $\alpha_{k+1} \leftarrow \alpha_k$
    $c \leftarrow c + 1$
end
else if $k > 1$ and $r_k - r_{k-1} < \eta$ then
    $\alpha_{k+1} \leftarrow \alpha_k$
    $c \leftarrow c + 1$
end
else if $c > 0$ then
    $c \leftarrow 0$
    $\alpha_{k+1} \leftarrow A(\mathrm{pos}(\alpha_k) - 1)$
end
else
    $c \leftarrow c + 1$
end
```

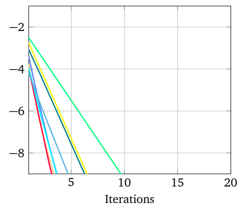
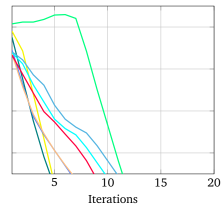
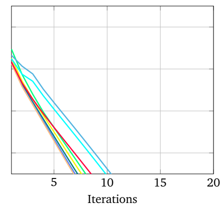
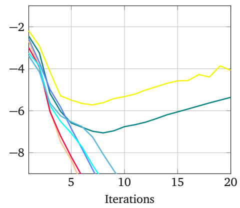
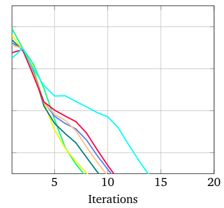
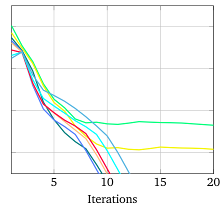
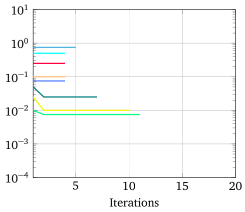
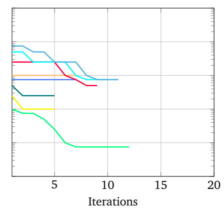
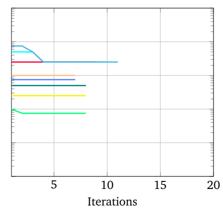
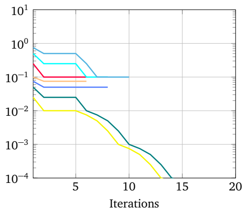
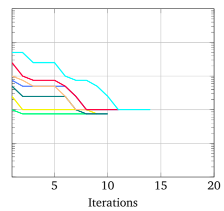
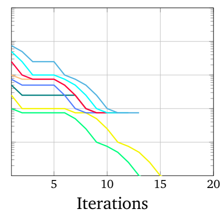
Legend		Square domain uniform mesh	Square domain non-uniform mesh	Square domain with hole
▬ $\alpha_0 = 0.01$ ▬ $\alpha_0 = 0.025$ ▬ $\alpha_0 = 0.05$ ▬ $\alpha_0 = 0.075$ ▬ $\alpha_0 = 0.1$ ▬ $\alpha_0 = 0.25$ ▬ $\alpha_0 = 0.5$ ▬ $\alpha_0 = 0.75$	$\log_{10} \lVert \mathbf{r} \rVert$	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 
	α	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 

Table 8.4. Results related to Algorithm 13.

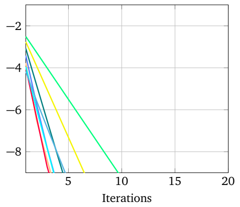
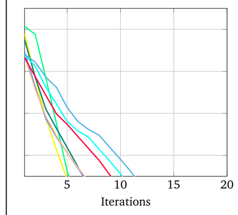
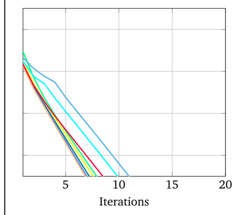
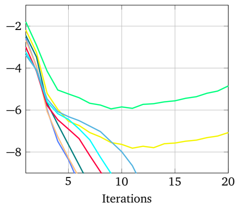
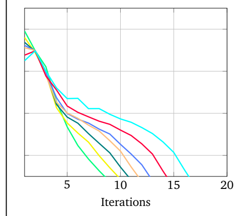
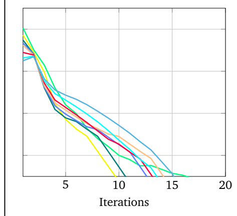
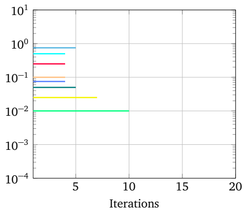
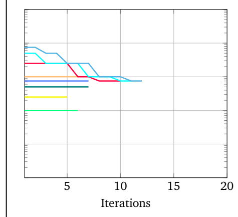
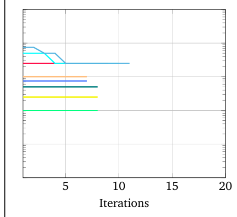
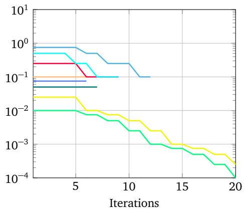
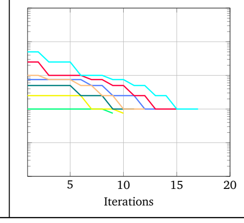
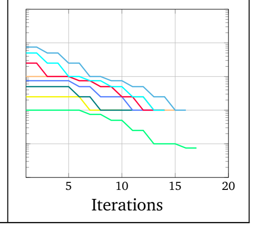
Legend		Square domain uniform mesh	Square domain non-uniform mesh	Square domain with hole
— $\alpha_0 = 0.01$ — $\alpha_0 = 0.025$ — $\alpha_0 = 0.05$ — $\alpha_0 = 0.075$ — $\alpha_0 = 0.1$ — $\alpha_0 = 0.25$ — $\alpha_0 = 0.5$ — $\alpha_0 = 0.75$	$\log_{10} \lVert \mathbf{r} \rVert$	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 
	α	$N_{\text{fine}} = 2193$ 	$N_{\text{fine}} = 7875$ 	$N_{\text{fine}} = 1858$ 
		$N_{\text{fine}} = 270977$ 	$N_{\text{fine}} = 209739$ 	$N_{\text{fine}} = 269847$ 

Table 8.5. Results related to Algorithm 14.

Chapter 9

MARS

9.1 Introduction to MARS

MARS is an open-source mesh management library designed to handle N-dimensional simplicial elements ($N \leq 4$). A more detailed description of the library can be found in Zulian, Patrick and Ganellari, Daniel and Rovi, Gabriele [2018]. The programming language is C++11. See Meyers, Scott [2005], Abrahams and Gurtovoy [2004], Vandevoorde, David and Josuttis, Nicolai M. [2002], Yang, Daoqi [2000] for further reading about C++. The "rt-elements" branch, written in C++17, is the one developed by the author. In this chapter, we present the main components of the framework. Since it can be subject to changes, some parts might differ. Hence we aim to explain the main ideas and we adopt shorter names for classes and functions. In addition, only the central part of the classes will be discussed, leaving the understanding of the minor details to the reader.

9.2 Introduction to rt-elements branch

The branch "rt-elements" in MARS allows the user to write and assemble easily its own linear and bilinear forms to be solved with the FE method. A general linear problem is the following. Find the trial function $u \in W$ such that, for any test function $v \in W$ it holds:

$$a(u,v) = f(v)\,. \tag{9.1}$$

Examples of the forms we are interested in are the following:

$$\begin{aligned} a(u,v) &= (\nabla u, \nabla v)_{L^2(\Omega)}\,, \\ f(v) &= (v, f)_{L^2(\Omega)}\,, \end{aligned} \tag{9.2}$$

where $W = H^1(\Omega)$ and the auxiliary function $f \in L^2(\Omega)$, or:

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) &= (\nabla \mathbf{u}, \nabla \mathbf{v})_{L^2(\Omega)}\,, \\ f(\mathbf{v}) &= (\mathbf{v}, \mathbf{f})_{L^2(\Omega)}\,, \end{aligned} \tag{9.3}$$

where $W = \left[H^1(\Omega)\right]^m$, with $0 < m \in \mathbb{N}$, and the auxiliary function $\mathbf{f} \in \left[L^2(\Omega)\right]^m$, or:

$$\begin{aligned} a([\boldsymbol{\sigma},\mathbf{u}],[\boldsymbol{\tau},\mathbf{v}]) = &+\gamma\,(\mathcal{A}\boldsymbol{\sigma},\mathcal{A}\boldsymbol{\tau})_{L^2(\Omega)} - \gamma\,(\boldsymbol{\varepsilon}(\mathbf{u}),\mathcal{A}\boldsymbol{\tau})_{L^2(\Omega)} + \delta\,(\mathrm{div}\boldsymbol{\sigma},\mathrm{div}\boldsymbol{\tau})_{L^2(\Omega)} \\ &-\gamma\,(\mathcal{A}\boldsymbol{\sigma},\boldsymbol{\varepsilon}(\mathbf{v}))_{L^2(\Omega)} + \gamma\,(\boldsymbol{\varepsilon}(\mathbf{u}),\boldsymbol{\varepsilon}(\mathbf{v}))_{L^2(\Omega)} \\ &+((\boldsymbol{\tau}\mathbf{n})_n,\mathbf{u}_n)_{L^2(\Gamma_C)} + ((\boldsymbol{\sigma}\mathbf{n})_n,\mathbf{v}_n)_{L^2(\Gamma_C)} \\ f([\boldsymbol{\tau},\mathbf{v}]) = &-\delta\,(\mathrm{div}\boldsymbol{\tau},\mathbf{f})_{L^2(\Omega)} + ((\boldsymbol{\tau}\mathbf{n})_n, g)_{L^2(\Gamma_C)} \end{aligned} \tag{9.4}$$

where $\mathbf{W} = \mathbf{H}_{\mathrm{div}}(\Omega) \times \mathbf{H}^1(\Omega)$ and the auxiliary functions $\mathbf{f} \in \mathbf{L}^2(\Omega)$ and $g \in L^2(\Gamma_C)$.

First the user defines the type of finite element FE, e.g. a simplex in d dimension, and then the FE spaces $V \subset W$, for the trial and the test functions, and V_{aux}, for other auxiliary functions. For example, for the equation (9.2), the Lagrangian spaces $V = P_m$ and $V_{\mathrm{aux}} = P_n$ could be chosen, where for varying $0 < m, n \in \mathbb{N}$ different discretization will arise. In general, V and V_{aux} can be direct sums of other FE spaces. Each of these spaces can be a single component FE space, like P_1 or RT_0, or an m-times components FE space, like $[P_1]^m$ or $[RT_0]^m$. If the space is written in bold, it means $m = d$. For example, for the FOSLS formulation in (9.4) $\mathbf{V} = \mathbf{RT}_0 \times \mathbf{P}_1$. A similar argument can be extended for the auxiliary space.

Once the FE and the FE spaces are chosen, the user defines the bilinear and linear forms. As we have seen, a form is a sum of integral terms involving trial, test, and auxiliary functions. The way the linear and bilinear forms are written in math is very compact and we would like to maintain this philosophy in MARS. For example, in MARS the user can write the bilinear form (9.4) as follows:

$$\begin{aligned} a([\boldsymbol{\sigma},\mathbf{u}],[\boldsymbol{\tau},\mathbf{v}]) = &+\gamma\,(\mathcal{A}\boldsymbol{\sigma}-\boldsymbol{\varepsilon}(\mathbf{u}),\mathcal{A}\boldsymbol{\tau}-\boldsymbol{\varepsilon}(\mathbf{v}),)_{L^2(\Omega)} + \delta\,(\mathrm{div}\boldsymbol{\sigma},\mathrm{div}\boldsymbol{\tau})_{L^2(\Omega)} \\ &+((\boldsymbol{\tau}\mathbf{n})_n,\mathbf{u}_n)_{L^2(\Gamma_C)} + ((\boldsymbol{\sigma}\mathbf{n})_n,\mathbf{v}_n)_{L^2(\Gamma_C)} \\ f([\boldsymbol{\tau},\mathbf{v}]) = &-\delta\,(\mathrm{div}\boldsymbol{\tau},\mathbf{f})_{L^2(\Omega)} + ((\boldsymbol{\tau}\mathbf{n})_n, g)_{L^2(\Gamma_C)}\,, \end{aligned} \tag{9.5}$$

and then MARS automatically builds the corresponding split version (9.4), used for run-time computations. In this way, math and code are very close to each other, but we still exploit the linearity of the integrals for reasons that will be discussed later in section 9.14. Furthermore, MARS automatically checks whether the user has written correctly the forms. We can have forms of order 0, 1, and 2, which are respectively sums of integrals fulfilling the following conditions:

- Each integral of the zero-form has no test nor trial functions;
- Each integral of the linear form has only one test function, in the left or in the right arguments;
- Each integral of the bilinear form has exactly one test function in the left-/right argument and one trial function in the right/left argument.

If none of these conditions is guaranteed, then a compile-time error occurs. In this way, MARS let the user know that, for example, the forms $(\mathbf{u},\mathbf{u})_{L^2(\Omega)}$, $(\mathbf{u}^T\mathbf{u},\mathbf{u})_{L^2(\Omega)}$ or $(\mathbf{u},\mathrm{div}\boldsymbol{\sigma})_{L^2(\Omega)}$ are wrong. But the user is still free to put test or trial functions where he or she desires. So $(\mathbf{u},\mathbf{v})_{L^2(\Omega)}$ and $(\mathbf{v},\mathbf{u})_{L^2(\Omega)}$ are both compatible.

In order to get the corresponding discrete version of the problem (9.1) on the mesh $\mathcal{T}_h$ approximating the domain Ω:

$$\mathbf{A}_h\mathbf{y}_h = \mathbf{f}_h\,, \tag{9.6}$$

it is necessary to assemble the matrix $\mathbf{A}_h$ and the right-hand side $\mathbf{f}_h$. To this aim, we exploit the linearity of the integrals over Ω as sum of integrals over the elements $K \in \mathcal{T}_h$. In this way, the global matrix $\mathbf{A}_h$ and vector $\mathbf{f}_h$ can be interpreted as the composition of local matrices and vectors. We definine $\mathbf{A}_{h,\mathrm{loc}} = \mathbf{A}_h|_K$, $\mathbf{f}_{h,\mathrm{loc}} = \mathbf{f}_h|_K$, $\mathbf{y}_{h,\mathrm{loc}} = \mathbf{y}_h|_K$. We approximate the local integrals over the element K by means of the quadrature rule. A quadrature rule approximates the integral of a function $i \in L^2(\Omega)$ as follows:

$$\int_K i(\mathbf{x})d\mathbf{x} \approx \sum_{i=1}^{N_{QR(o)}} i(\mathbf{p}_i)w_i\,, \tag{9.7}$$

where $\{\mathbf{p}_i\}_{i=1}^{N_{QR(o)}}$ are the $N_{QR(o)}$ quadrature points and $\{w_i\}_{i=1}^{N_{QR(o)}}$ are the corresponding weights for a quadrature rule of order o. However, if the integrand function i is a polynomial, then it is possible to compute the integral exactly. In particular, if the integrand function is a Euclidean product and its arguments are polynomials, then the integral can be computed exactly by choosing the right quadrature order. Since all the trial, test, and auxiliary functions are approximated by means of FE functions, this is the case. Indeed FE functions, if restricted to the element K, reduce to polynomials, and thus their combinations are again a polynomial.

In a form, each integral has its minimum order of quadrature that is necessary for its exact computation. Two choices are now possible: different integrals can have different required quadrature orders or all of them have the same quadrature order, equal to the maximum of all quadrature orders. Both approaches have pros and cons, but in MARS we opted for the former possibility. To understand how they would influence the implementation, let us focus on the assembly. In particular, we can examine:

$$a(u,v) = (u,v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)}\,, \tag{9.8a}$$

$$f(v) = (f,v)_{L^2(\Omega)}\,. \tag{9.8b}$$

If we discretize the problem (9.8) by means of continuous linear Lagrange FE, i.e., $V = V_{\text{aux}} = P_1$, we get (9.9):

$$(\mathbf{A}_{h,\text{loc}})_{ij} = (\phi_i, \phi_j)_{L^2(\Omega)} + (\nabla\phi_i, \nabla\phi_j)_{L^2(\Omega)} \qquad i,j = 1,\dots,d+1\,, \tag{9.9a}$$

$$(\mathbf{f}_{h,\text{loc}})_i = (\phi_i, \sum_{k=1}^{d+1}(\phi_k f_k))_{L^2(\Omega)} \qquad i = 1,\dots,d+1\,, \tag{9.9b}$$

where, for simplicity of notation, we consider the local numbering for the indices i,j. The shape functions ϕ_i are associated to the corresponding operators that, in this case, are the identity and the gradient. We obtain ϕ_i and $\nabla\phi_i$. In general, other operators can be used, such as the divergence, the trace, composite operators and so on.

In order to reduce the computational cost, it is convenient to compute the reference shape functions $\hat{\phi}_i \in \mathbb{R}$, $\hat{\nabla}\hat{\phi}_i \in \mathbb{R}^d$ for $i = 1,\dots,d+1$, before iterating over all the elements. Given the element K, the mappings $M_{CG^q,I} = 1$ and $M_{CG^q,\nabla} = \mathbf{J}^{-T}$ can be computed only once in order to get $\phi_i = \hat{\phi}_i$ and $\nabla\phi_i = \mathbf{J}^{-T}\hat{\nabla}\hat{\phi}_i$ for $i = 1,\dots,d+1$. The functions ϕ_i can be used for both the left and right arguments respectively of the first integral in (9.9a) and in (9.9b), while $\nabla\phi_i$ can be used in the left and right arguments of the second integral in (9.9a). In this way, computational time is saved with respect to the full computation of ϕ_i and $\nabla\phi_i$ every time they appear in the forms.

Let us assume the quadrature rule to be Gaussian. Since i in (9.7) is a dot product involving shape functions, the same shape functions have to be evaluated in the quadrature points as well. Therefore we need an array which contains the evaluation of the shape functions in $\{\mathbf{p}_i\}_{i=1}^{N_{QR(o)}}$. In particular, we only need the reference shape functions in the reference quadrature points $\{\hat{\mathbf{p}}_i\}_{i=1}^{N_{QR(o)}}$. Indeed the array of reference shape functions in $\{\hat{\mathbf{p}}_i\}_{i=1}^{N_{QR(o)}}$ can be pre-computed and then it is sufficient to apply the proper mapping to each of its components. Of course, for different quadrature orders, different are the points in which the integrand i has to be evaluated and so different collections of evaluations of shape functions are required. See section 9.3. However, the FE, the FE spaces, and the forms are known at compile-time. Hence it is possible to identify the order of quadrature of each integral at compile-time and the arrays of evaluations of the reference shape functions can be computed at compile-time as well.

By inspecting (9.9), we see that the first integral in (9.9a) and the one in (9.9b) are of the third order. So we can precompute the collection of arrays $\hat{\Phi}_3 = \{(\hat{\phi}_1(\hat{\mathbf{p}}_j), \dots, \hat{\phi}_{d+1}(\hat{\mathbf{p}}_j))\}_{j=1}^{N_{QR}(3)}$. On the other hand, the second integral in (9.9a) has order one, so we can precompute the collection of arrays $\hat{\nabla}\hat{\Phi}_1 = \{(\hat{\nabla}\hat{\phi}_1(\hat{\mathbf{p}}_j), \dots, \hat{\nabla}\hat{\phi}_{d+1}(\hat{\mathbf{p}}_j))\}_{j=1}^{N_{QR}(1)}$. See Table 9.1 for rules explaining the computation of the quadrature order of an integral. Since the shape functions ϕ and $\nabla\phi$ are involved in different integrals and do not influence each other, we can reduce to the minimum the evaluation in the quadrature points for both.

However, given a constant vector $\mathbf{b} \in \mathbb{R}^d$, we can consider the problem (9.10):

$$a(u,v) = (u,v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)} + (u, \mathbf{b}^T \nabla v)_{L^2(\Omega)}\,, \tag{9.10a}$$

$$f(v) = (f,v)_{L^2(\Omega)}\,, \tag{9.10b}$$

which differs from (9.8) for the addition of the third integral in (9.10a). This integral has order two. In MARS, for problem (9.10) we also precompute $\hat{\Phi}_2 = \{(\hat{\phi}_1(\hat{\mathbf{p}}_j), \dots, \hat{\phi}_{d+1}(\hat{\mathbf{p}}_j))\}_{j=1}^{N_{QR}(2)}$ and $\hat{\nabla}\hat{\Phi}_2 = \{(\hat{\nabla}\hat{\phi}_1(\hat{\mathbf{p}}_j), \dots, \hat{\nabla}\hat{\phi}_{d+1}(\hat{\mathbf{p}}_j))\}_{j=1}^{N_{QR}(2)}$. Altogether $\hat{\Phi}_2$, $\hat{\Phi}_3$, $\hat{\nabla}\hat{\Phi}_1$, $\hat{\nabla}\hat{\Phi}_2$ are precomputed. Thus at run-time for each element K, it would be necessary to map them in order to get the corresponding arrays of evaluation of shape functions on K, i.e., Φ_2, Φ_3, $\nabla\Phi_1$, $\nabla\Phi_2$. To avoid the mappings of four different collections, it could be argued that it is better to take just the maximum order of quadrature among all the integrals and compute and map only $\hat{\Phi}_3$ and $\hat{\nabla}\hat{\Phi}_3$. However, in this way, it would be necessary to compute a third order collection $\hat{\nabla}\hat{\Phi}_3$ instead of two lower order collections, $\hat{\nabla}\hat{\Phi}_1$ and $\hat{\nabla}\hat{\Phi}_2$. With the increasing of the dimension d, this could not be beneficial. However it is clear that the best choice depends on the contest and MARS is programmed so that the forms of the type of (9.9) can be optimized.

Until now, we have considered V to coincide with a single FE space. However, the advantage of computing arrays of reference shape functions in the reference quadrature points can be extended to more FE spaces. For example, if $V = P_1 \times P_1$, for a bilinear form of the type:

$$a([u,p],[v,q]) = (\nabla u, \nabla v)_{L^2(\Omega)} + (\nabla p, \nabla q)_{L^2(\Omega)} + (p,v)_{L^2(\Omega)}\,, \tag{9.11}$$

we can precompute $\hat{\nabla}\hat{\Phi}_1$ only once and use it for both the first two integrals which involve different trial and test functions in the left and right arguments of the dot product. This means that, to reduce the computational cost, we need to detect the minimum required shape functions to evaluate for all the FE spaces of V. The same argument would not be true anymore for $V = P_1 \times P_2$. We would need different collections of gradient of shape functions, for both P_1 and P_2. However the mapping $M_{CG^q,\nabla} = \mathbf{J}^{-T}$ could be used for both. Indeed the mapping does depend on the FE family and on the operator of the shape function, but it is independent of the order and the continuity type of the FE space. Since two are the operators of the forms, i.e., the gradient and the identity, a collection of mappings is required as well.

From what has been explained so far, it is clear some processes can be carried out and optimized at compile-time, others at run-time. Roughly speaking, whatever strictly depends on the mesh is at run-time, everything else is at compile-time. Therefore the FE spaces, the forms, the order of quadrature of the integrals, the dimension of the shape functions, and of the collections of their reference evaluations for all necessary order of quadratures and operators, are known at compile time. In particular, we can say the following.

- The trial space V is defined as $V = V_1 \times \ldots \times V_v$, where each V_i, for $i = 1, \ldots, v$, is a FE space identified by its FE family, order, continuity type and number of components. We discuss the local and global FE spaces respectively in sections 9.4 and 9.5.

- To compute the collections of mappings from the reference element $\hat{K}$, we need to remove the duplicates of the FE spaces in V. Then, for a given FE space obtained in this way, all the mappings related to the operators appearing in the forms are built. See section 9.15.

- To compute all the collections of evaluations of reference shape functions, we need to remove all the duplicate FE spaces from V. Then, for each space obtained in this way, we need to consider all the shape functions, with their associated operators (like the identity, the gradient, the divergence, the trace or custom operators as explained in section 9.9), appearing in the forms and, for each quadrature order, we build the corresponding arrays of evaluations. See section 9.16.

- It is necessary to distinguish between volumetric and surface shape functions and mappings. For this reason, we need to define the objects: `volumetric_map`, `volumetric_shape_function`, `surface_map` and `surface_shape_function`. In Algorithm 15 we describe the run-time assembly procedure, where it is also stressed out the difference between the volumetric and the surface quantities. The former have to be updated for each element, but only once. The latter have to be updated for each face of the element, but only if it lies on an external boundary.

Algorithm 15: Assembly algorithm

```
for elem = 1,..,$N_{elems}$ do
    FE.init(elem);
    volumetric_map.init(FE);
    volumetric_shape_function.init(FE, volumetric_map);
    bilinear_form.eval(FE, volumetric_shape_function, $\mathbf{A}_h$);
    linear_form.eval(FE, volumetric_shape_function, $\mathbf{f}_h$);
    for face = 1,..,$N_{faces}$ do
        FE.init_boundary(face);
        if FE.is_side_on_boundary() then
            surface_map.init(FE);
            surface_shape_function.init(FE,surface_map);
            bilinear_form.eval(FE, surface_shape_function, $\mathbf{A}_h$);
            linear_form.eval(FE, surface_shape_function, $\mathbf{f}_h$);
        end
    end
end
```

9.3 Tuples and constexpr expressions

In section 9.2 the importance of collections of objects of different types has been highlighted. Indeed we need to collect different FE spaces, different arrays of evaluations of shape functions, and also different mappings. A tuple is a container object, with size declared at compile-time, which can collect objects of different types. Its variadic definition is the following:

```
template< class ... Args >
class tuple;
```

The first tuple to define is the one for the FE spaces. Then the tuples for the mappings and the shape functions can be consequently declared. To easily manage tuples, MARS defines operations like the removal of duplicates, the removal of old types or the addition or change of new types in certain positions, the return of the position of the first occurrence of a given type in the tuple, and so on. Furthermore, MARS massively uses the `constexpr` specifier. If a variable is declared as `constexpr`, then its evaluation, when possible, can be done at compile-time. Thanks to *C*++17 features, the `constexpr` specifier can be used, for example, to evaluate reference shape functions in some reference quadrature points.

Many of the operations on tuples described above require to associate to each component of the tuple a number. For this reason, it is important to handle collections of numbers. However, at compile-time, it is still easier to handle tuples than arrays. At least, this is true for *C*++11, which was initially used for the purposes of this thesis. Nevertheless, the tuple type can collect different types, but, of course, not integer objects. In order to have a type representation of integers, the class Number, which is in a bijective relation with the integers, has been defined:

```
 template<Integer N> class Number
                    {public: static constexpr Integer value=N;};
```

Tuples of `Numbers` related to different integers can be used like arrays of integers. The only caution is to access the `value` variable of each `Number`.

9.4 Local FunctionSpace

We define a general template class called `BaseFunctionSpace` for the FE spaces, based on their family, order, continuity and number of components. With this definition, we are able to make the considered FE space element-independent.

```
template <Integer F,Integer O,Integer C=Continuous, Integer NC=1>
 class BaseFunctionSpace
 {
 public:
    static const Integer FEFamily=F;
    static const Integer Order=0;
    static const Integer Continuity=C;
    static const Integer NComponents=NC;
};
}
```

The BaseFunctionSpace class considers just the following parameters:

1. FEFamily is the FE family. This parameter is a negative integer identifying the family itself.

   ```
   constexpr Integer LagrangeFE = -10;
   constexpr Integer RaviartThomasFE = -11;
   ```

2. Order is the order of approximation, which is an integer number. Considering simplicial meshes, for Lagrangian functions the order refers to the local order of the polynomia. So Order$= 0, 1, 2...$ stands for constant, linear, quadratic functions and so on. In the case of Raviart-Thomas functions, always on simpicial meshes, Order$= 0, 1, ...$ stands for linear, quadratic functions and so on.

3. Continuity is an integer parameter. Continuity $=$ Continuous $= 1$ means that the continuity of the degrees of freedom between the elements is enforced. On the other hand, Continuity $=$ Discontinuous $= 0$, if no continuity among dofs on different elements is enforced. See (5.11) and (5.12).

   ```
   constexpr Integer Discontinuous = 0;
   constexpr Integer Continuous = 1;
   ```

4. NComponents refers to the number of components of the space, which differs from the number of components of the shape function itself. For example, the RT_0 space is a single component space, but its shape functions are vectorial function of dimension d.

The fact that all these parameters -especially the Continuity one- are integers, is a good strategy for future generalizations which can consider different scenarios. In the library, new definitions have been created based on this class to make it easier to call standard and implemented spaces.

```
template<Integer Order,Integer NComponents>
using Lagrange=BaseFunctionSpace<LagrangeFE,Order,Continuous,NComponents>;

template<Integer Order,Integer NComponents>
```

```
using LagrangeDG=BaseFunctionSpace<LagrangeFE,Order,Discontinuous,NComponents>;

template<Integer Order,Integer NComponents>
using RT=BaseFunctionSpace<RaviartThomasFE,Order,Continuous,NComponents>;

template<Integer Order,Integer NComponents>
using RTDG=BaseFunctionSpace<RaviartThomasFE,Order,Discontinuous,NComponents>;
```

A further specialization is obtained by considering the type of element of the mesh, which gives rise to the class `BaseElementFunctionSpace`. This class, in addition to the same parameters of the `BaseFunctionSpace` class, also considers, as a first parameter, the type of element. The general `Elem` class is characterized by two parameters, the dimension of the problem `Dim` and the one of the manifold `MDim`. For example, given a triangle in 3D, $\texttt{Dim} = 3$ and $\texttt{MDim} = 2$.

```
 template <typename E, Integer FEF, Integer O,Integer C, Integer NC>
 class BaseElementFunctionSpace
 {
    using  Elem=E;
    static constexpr Integer Dim=Elem::Dim;
    static constexpr Integer ManifoldDim=Elem::ManifoldDim;
    static constexpr Integer FEFamily=FEF;
    static constexpr Integer Order=O;
    static constexpr Integer NComponents=NC;
    static constexpr Integer Continuity=C;
 };
```

The `ElementFunctionSpace` class inherits from `BaseElementFunctionSpace` and different specialisation have been written for Lagrange and Raviart-Thomas spaces.

```
template <typename E,Integer FEF,Integer O,Integer C, Integer NC>
class ElementFunctionSpace<Elem,FEF,O,C,NC>:
      public BaseElementFunctionSpace<E,FEF,O,C,NC>
 {
  public:
     static constexpr const std::array<Integer,M> entity;
     static constexpr const std::array<Integer,N> dofs_per_entity;
     static constexpr const Integer ShapeFunctionDim;
 };
```

To explain its usage, we first must introduce the concept of entity in MARS. We now consider the case of simplicial meshes of dimension `MDim` immersed in a space of dimension `Dim`, with $\texttt{MDim} \leq \texttt{Dim}$. Then an entity of dimension $\texttt{EDim} \leq \texttt{MDim}$, for $\texttt{EDim} = 0, 1, 2, 3, \ldots$, is a point, a segment, a triangle, a tetrahedron and so on. In the array `entity` we set, in an increasing order of dimension, the dimension of the entities of interest for the definitions of the degrees of freedom (dofs). Similarly, in `dofs_per_entity`, we explicit how many dofs per entity are required. Finally `ShapeFunctionDim` defines the dimension of the shape function; this number is independent of the number of components.

For example, for linear Lagrangian elements on a simplex, the dofs belong to the vertices of $\texttt{EDim} = 0$ and there is only one dof per vertex. For first-order Raviart-Thomas functions, there are `MDim` dofs per face (of dimension $\texttt{EDim} = \texttt{MDim} - 1$) and `MDim` internal dofs (referred to the simplex of dimension $\texttt{EDim} = \texttt{MDim}$). Therefore:

- for P^1 elements:

```
static constexpr const std::array<Integer,1> entity={0};
static constexpr const std::array<Integer,2> dofs_per_entity={1};
static constexpr const Integer ShapeFunctionDim=1;
```

- for RT_1 elements:

```
static constexpr const std::array<Integer,2> entity={MDim-1,MDim};
static constexpr const std::array<Integer,2> dofs_per_entity={MDim,MDim};
static constexpr const Integer ShapeFunctionDim=MDim;
```

We notice that the definition of the `ElementFunctionSpace` is local and is therefore independent of the continuity or discontinuity of the global FE space. To create a new local function space, it is necessary to introduce its FE family integer and define the corresponding variables in the `ElementFunctionSpace`.

9.5 Global FunctionSpaces

Having defined the class `ElementFunctionSpace` for local function spaces, we now want to define the class `FunctionSpace` for global function spaces. To this aim, we need first to introduce three new classes: the `Mesh` class, which depends on the type of element considered; the `Bisection` and the `Node2ElemMap` classes, which depend on the `Mesh` class. The `Bisection` class is used for bisection algorithms, while `Node2ElemMap` is used for building a map from the nodes of the mesh to the elements, very useful for the assembly of the operators.

```
MeshT mesh;
Bisection<MeshT> bisection(mesh);
Node2ElemMap<MeshT> n2em(mesh,bisection);
```

The mesh consists of a list of elements, read from an input file or built with a mesh generation function. In case of refinement, the new elements are just added to the previous list. In this way, more levels of a mesh can be built and used, for example, for multigrid methods. For adaptively or uniformly refining the mesh, we can exploit the Bisection class. Let n_levels be the number of levels and refinement_jump be the number of times we use the bisection algorithm between two levels of the mesh. For a uniform refinement, we can do:

```
for(int i=0;i<n_levels;i++)
{
bisection.tracking_begin();
bisection.uniform_refine(refinement_jump);
bisection.tracking_end();
}
```

Finally it is possible to define the class FunctionSpace for a single function space, dependent on the Mesh and on a particular BaseFunctionSpace. In the following we show the different specializations for continuous linear Lagrangian functions with one component and N components, for discontinuous linear Lagrangian functions with MDim components, and for first-order *RT* functions with MDim components:

```
template<typename MeshT,typename BaseFunctionSpace>
class FunctionSpace {...};

using P_1_1=FunctionSpace<MeshT,Lagrange<1,1>>;
using P_1  =FunctionSpace<MeshT,Lagrange<1,N>>;
using DGP_1=FunctionSpace<MeshT,LagrangeDG<1,MDim>>;
using RT_1 =FunctionSpace<MeshT,RT<1,MDim>>;
```

Once the type has been defined, the corresponding object space can be built:

```
P_1_1  p11(mesh,bisection,n2em);
P_1     p1(mesh,bisection,n2em);
DGP_1 dgp1(mesh,bisection,n2em);
RT_1   rt1(mesh,bisection,n2em);
```

Each space contains its dofmap, i.e., a map which associates to each element the corresponding dofs of the space for every level of the mesh. However, in the definition of the bilinear and linear forms, more than one global space can occur. In particular, we must distinguish between two collection types of global spaces: the mixed space, which is a container for all the spaces where the test and trial variables belong; and the auxiliary space, which is a container for all the spaces needed for all other functions that are neither tests nor variables. By using variadic templates, we can define:

```
template<typename Arg,typename...Args>
class MixedSpace<Arg,Args...>{...};

template<typename Arg,typename...Args>
```

```
class AuxMixedSpace<Arg,Args...>{...};

template<typename...Args,typename...AuxArgs>
class FullSpace<MixedSpace<Args...>,AuxMixedSpace<AuxArgs...>> {...};
```

and then build respectively the mixed, the auxiliary, the full spaces and the corresponding smart pointer:

```
auto  Wtrial = MixedFunctionSpace(rt1,dgp1,p1_1);
auto    Waux = AuxFunctionSpacesBuild(p1,p1_1);
auto       W = FullSpaceBuild(Wtrial,Waux);
using W_type = decltype(W);
auto   W_ptr = std::make_shared<W_type>(W);
```

Inside the FullSpace class, all the spaces inserted are collected in a proper tuple tuple_of_spaces of type TupleOfSpaces. See section 9.3 for explanations of the tuples in *C*++. In addition, the tuple object unique_tuple_of_spaces of type UniqueTupleOfSpaces is the tuple tuple_of_spaces without duplicates. This means, for example, that if the MixedFunctionSpace is built from tuple_of_spaces = (p1, p1, p1, rt1, p1), the corresponding tuple of unique spaces is unique_tuple_of_space = (p1, rt1). The object unique_tuple_of_spaces is used for the assembly of shape functions. Indeed we reduce the collection of spaces and the corresponding shape functions to compute, to its minimum. This avoid the computation of the same shape function, associated to a given operator, multiple times. Of course, for a correct assembly, it is necessary to relate the TupleOfSpaces to the UniqueTupleOfSpaces. SpacesToUniqueNumbersTuple is a tuple of numbers which maps a type in TupleOfSpaces to the corresponding position in UniqueTupleOfSpaces. In summary, we get:

```
using TupleOfSpaces=std::tuple< Space_1,...,Space_n >;
using UniqueTupleOfSpaces=Unique<TupleOfSpaces>;
using SpacesToUniqueNumbersTuple=FromTo<TupleOfSpaces,UniqueTupleOfSpaces>;
```

For the dual contact problem, the tuples look like as follows:

```
using TupleOfSpaces=
     tuple<ElementFunctionSpace<Elem,RaviartThomas,1,Continuous,   2>,
           ElementFunctionSpace<Elem,Lagrange,     1,Discontinuous,2>,
           ElementFunctionSpace<Elem,Lagrange,     1,Continuous,   1>,
           ElementFunctionSpace<Elem,Lagrange,     1,Continuous,   2>,
           ElementFunctionSpace<Elem,Lagrange,     1,Continuous,   1>>

using UniqueTupleOfSpaces=
     tuple<ElementFunctionSpace<Elem,RaviartThomas,1,Continuous,   2>,
           ElementFunctionSpace<Elem,Lagrange,     1,Discontinuous,2>,
           ElementFunctionSpace<Elem,Lagrange,     1,Continuous,   1>,
           ElementFunctionSpace<Elem,Lagrange,     1,Continuous,   2>>

using  SpacesToUniqueNumbersTuple=tuple<Number<0>,
                                        Number<1>,
                                        Number<2>,
                                        Number<3>,
                                        Number<2> >
```

There is no difference between TupleOfSpaces and UniqueTupleOfSpaces regarding the trial and test spaces, because they are all different. However P_1 in $\mathrm{W}_{\mathrm{aux}}$, which is already present in $\mathrm{W}_{\mathrm{mixed}}$, does not have to be repeated. In SpacesToUniqueNumbersTuple we see the collection of numbers mapping TupleOfSpaces into UniqueTupleOfSpaces.

9.6 Trial, test and auxiliary functions

Trial and test functions can be declared after W_ptr, a shared pointer to the full space, has been initialized. For creating the trial and test functions belonging to the N-th space of the mixed space, we can do:

```
 auto TrialN = MakeTrial<N>(W_ptr);
 auto TestN  = MakeTest <N>(W_ptr);
```

where the Trial and Test classes are Expressions (see section 9.7) that depend on the global collection of spaces W_ptr, on the integer N and on a particular operator which is set by default to be the IdentityOperator. However other different differential operators such as the gradient, the divergence or, as we will see, composite operators, can be used. The trial and tests with different operators can be created directly in the forms by using, for example, Grad(TrialN), Div(TrialN), Trace(TrialN) and so on.

```
template<typename MixedSpace, Integer N, typename Operator=IdentityOperator>
class Test: public Expression<Test<MixedSpace,N,Operator>>{...};

template<typename MixedSpace, Integer N, typename Operator=IdentityOperator>
class Trial: public Expression<Trial<MixedSpace,N,Operator>>{...};
```

The definitions for the Trial and the Test classes are identical. They also have an integer variable value, which refers to the position of the N-th space of the trial/test function in UniqueTupleOfSpaces.

```
template<typename MixedSpace, Integer N, typename  Operator_=IdentityOperator>
class Test: public Expression<Test<MixedSpace,N,Operator_>>
```

```
{public:
using Operator=Operator_;
using UniqueTupleOfSpaces=typename MixedSpace::UniqueTupleOfSpaces;
using SpacesToUniqueNumbersTuple=typename MixedSpace::SpacesToUniqueNumbers;
static constexpr Integer Nmax=TupleTypeSize<UniqueTupleOfSpaces>::value;
static constexpr Integer number=N;
static constexpr Integer value=GetType<SpacesToUniqueNumbersTuple,N>::value;

Test(const std::shared_ptr<MixedSpace>& W_ptr): spaces_ptr_(W_ptr) {}
Test(const MixedSpace& W): spaces_ptr_(std::make_shared<MixedSpace>(W) {}

inline auto spaces_ptr()const {return spaces_ptr_;};

private:
std::shared_ptr<MixedSpace> spaces_ptr_;
};
```

Declaring tests and trials for the previous example, we do obtain:

```
auto sigma = MakeTrial<0>(W_ptr);
auto     u = MakeTrial<1>(W_ptr);
auto theta = MakeTrial<2>(W_ptr);
auto   tau = MakeTest <0>(W_ptr);
auto     v = MakeTest <1>(W_ptr);
auto   rho = MakeTest <2>(W_ptr);
```

The auxiliary function referred to the N-th auxiliary space is instantiated similarly. Separately it is defined the class `FunctionType` which defines the function to be discretized. It requires a `static` `eval` method for its evaluation which takes as input the point of evaluation and the FE object, which is unused in this case. So, for example, if we want to define the class `FunctionZero`, that has the dimension as a template parameter, we need the following code:

```
template<Integer Dim> class FunctionZero
{public: using type=Matrix<Real,Dim,1>;

   template<typename Point,typename Elem>
   static type eval(const Point& p, FiniteElem<Elem>& FE)
   {
    Matrix<Real,Dim,1> func;
    for(Integer i=0;i<Dim;i++)
     func(i,0)=0;
    return func;
   }
};
```

where the definition of the `Matrix` class is trivial. Then two are the ways for defining volumetric or boundary functions:

```
auto FuncN      = MakeFunction<N,FunctionType>(W_ptr);
auto TraceFuncN = MakeTraceFunction<N,FunctionType>(W_ptr);
```

that have, as operators, `IdentityOperator` and `TraceOperator`.

9.7 Expression templates

Linear and bilinear forms are substantially sums of integrals. Their complexity grows with the complexity of each integral term and with the increase of their numerosity. In principle, we would like to write compile-time expressions for the forms which can be used for the assembly on a mesh, given as input at run-time. The basic operations which can appear in the forms are unary or binary. For example, between integrals, we do have addition or subtraction, whilst inside the integrals, we can have addition, subtraction, multiplication, dot product, and so on. Each operation with one or more arguments is a new type and the combination of different operations, in turn, gives rise to new different types, which can be hard to handle.

For the sake of clarity, let us focus on the sum of integrals that appear in the linear and bilinear forms. Since each integral is different, a templated class for integrals will be required. We identify the integral object with the lowercase letter `i` and its type with the capital letter `I`. Whenever we do sum up two integrals $\mathtt{i}_1$ of type $\mathtt{I}_1$ and $\mathtt{i}_2$ of type $\mathtt{I}_2$, the result $\mathtt{i}_1 + \mathtt{i}_2$ is an object of a new templated addition class of type $\mathtt{A} < \mathtt{I}_1, \mathtt{I}_2 >$. Such class takes, as template arguments, two integral classes. However, this class is not an integral class as well, because it does not evaluate a single integral, but the sum of two. If we go even further, it is clear that for the sum of three integrals $\mathtt{i}_1 + (\mathtt{i}_2 + \mathtt{i}_3)$, we would first need to define the type $\mathtt{A} < \mathtt{I}_2, \mathtt{I}_3 >$ and then $\mathtt{A} < \mathtt{I}_1, \mathtt{A} < \mathtt{I}_1, \mathtt{I}_2 >>$. For $(\mathtt{i}_1 + \mathtt{i}_2) + \mathtt{i}_3$, on the other hand, we should define the class $\mathtt{A} < \mathtt{A} < \mathtt{I}_1, \mathtt{I}_2 >, \mathtt{I}_3 >$. By adding more and more terms, the more combinations we can have and must define. So it is evident that this way of proceeding is far from being optimal. This happens because the finite sum of integrals is, mathematically, again an integral. But in *C++*, the type of the sum of two integral classes is a totally different class which is not an integral anymore. We need to homogenize the resulting type of unary and binary operations to their arguments so that both, the arguments and the results of the operations, are simply `Expressions`. To this aim, let us consider the following example:

```
class Base: public Expression<Base>
{};
```

The Base class inherits from the class Expression that has, as template parameter, the Base class itself. Typically, the Base class could inherit from classes that have already been defined. Since Expression depends on Base itself, which is not been declared yet, a recursion shows up. However, *C++* enables this possibility. This suggests the name of *curiously recursive template pattern* (CRTP). Such a pattern is very useful because it enables all classes to be, in addition to their type, expressions as well. In this way, the arguments of the operations can simply be expressions and the operations will result in expressions as well. The Expression class is very simple. It is a template class that depends on the Derived type. Its methods permit to access the Derived type object which can be therefore used both as an Expression $<$ Derived $>$ type or a Derived type.

```
template<typename Derived> class Expression
{public: inline constexpr Derived &derived()
                 {return static_cast<Derived &>(*this);}
          inline constexpr const Derived &derived() const
                 {return static_cast<const  Derived &>(*this);}
          inline constexpr operator Derived &()
                 {return derived();}
          inline constexpr  operator const Derived &() const
                 {return derived();}
};
```

The previous sum of three integrals can be seen now as sum of expressions. So we do have:

```
using Addition_2_3=Expression<Addition<Expression<I_2>,Expression<I_3>>>;
using Addition_1_2_3=Expression<Addition<Expression<I_1>,Addition_2_3>>
```

This type is different and its evaluation follows a different order, even thought the final result is the same, of the tree of expressions below:

```
using Addition_1_2=Expression<Addition<Expression<I_1>,Expression<I_2>>>;
using Addition_1_2_3=Expression<Addition<Addition_1_2,Expression<I_3>>>
```

It is clear that any combination of expressions results in a tree of expressions. This tree has to be run-time evaluated later in the assembly. Its order of evaluation follows the same structure of the expressions tree.

9.8 Predefined Operators

9.8.1 Algebraic and differential operators

In MARS, there are predefined operations that can be used for creating new expressions. Some standard algebraic operations, are:

- unary plus;
- unary minus;
- binary plus (addition);
- binary minus (subtraction);
- binary multiplication;
- binary division;
- binary inner product;
- unary transposition;

and other operators for the only trial and test functions, that are expressions as well, are:

- unary differential operators (gradient, divergence);
- unary trace;

The application of a differential operator or the trace operator to a test/trial function creates a new test/trial. The corresponding operator is not the identity anymore. Instead, it is the applied differential or trace operator.

9.8.2 Examples of a unary and a binary operator classes

Here we present a snippet of code for the definition of the UnaryPlus and the Addition classes. We first introduce the operator+. It can be defined to take one or two inputs expressions. In the first case, the object returned is of the type UnaryPlus, whilst in the second case is of the type Addition.

```
template<typename Derived>
class UnaryPlus<Expression<Derived>>
operator+(const Expression<Derived>& expr)
{return UnaryPlus<Expression<Derived>>(expr);}}

template<typename L,typename R>
class Addition<Expression<L>, Expression<R>>
operator+(const Expression<L>& expr_left, const Expression<R>& expr_right)
{return Addition<Expression<L>,Expression<R>>(expr_left,expr_right);}
```

Here the definition of the UnaryPlus class:

```
template<typename Derived>
class UnaryPlus<Expression<Derived>>:
public Expression<UnaryPlus<Expression<Derived>>>
{
public:
UnaryPlus(const Expression<Derived>& expr_value):
value_(expr_value.derived()){};

Derived& operator()(){return value_;};
const constexpr Derived& operator()()const{return value_;};
Derived& derived(){return value_;};
const constexpr Derived& derived()const{return value_;};

private:
Derived value_;
};
```

and here the one for the `Addition` class:

```
template< typename L,typename R>
class Addition<Expression<L>,Expression<R>>:
public Expression<Addition<Expression<L>,Expression<R>>>
{
public:
using Left=L;
using Right=R;
Addition(const Expression<L>& expr_left, const Expression<R>& expr_right):
left_(expr_left.derived()),
right_(expr_right.derived())
{};

const constexpr L& left()
const{return left_;};
const constexpr R& right()const
{return right_;};
auto operator()()
{return Addition<Expression<L>,Expression<R>>(left_,right_);};
auto operator()()const
{return Addition<Expression<L>,Expression<R>>(left_,right_);};

private:
Left left_;
Right right_;
};
```

It is interesting to notice that `UnaryPlus` has one private member, `value_` of type `Derived` and not `Expression`. For this reason, in the constructor we do not initialize `value_` simply using the `expr_value` object. Instead, we exploit the method derived() of the Expression class, which returns the object with its templated type, i.e., Derived. We do the same in the Addition class for both the left and right expressions, expr_left and expr_right, which are derived in order to obtain a Left and a Right objects.

9.9 Composite Operators

In MARS it is also possible to have user-defined operators which can be compositions of the predefined operators. A new operator can be built by means of the NewOperator function and can be used later in the linear and bilinear forms. For example, in order to define the inverse of the elasticity operator C_inverse or the asymmetry term of a 2D tensor Asym, we can do as follows:

```
constexpr Real mu=1.0;
constexpr Real lambda=9999999999.0;
constexpr auto C_1=Constant<Scalar>(1/(2 mu));
constexpr auto C_2=Constant<Scalar>(-lambda/((2 mu)(MDim lambda+ 2 mu)));
constexpr auto id_mat=Constant<Identity<MDim>>();
constexpr auto a_mat=Constant<Mat<2,2>>(0.0,1.0, 0.0,0.0);
auto C_inverse=NewOperator(C_1 Identity()+ C_2 id_mat MatTrace(Identity()) );
auto Asym=NewOperator(Inner(a_mat,(Identity()-Transpose(Identity()))));
```

where we have omitted the casting from Integer to Real variables.

9.10 Integrals

In MARS, the integral type for both volumetric and surface integrals is L2DotProductIntegral. Volumetric integrals can be constructed by means of the L2Inner() function which takes the two inputs of the L^2 scalar product. On the other hand, boundary integrals are constructed using the surface_integral() function. Its first input is the boundary label of interest, while the second and the third inputs are again the L^2 scalar product arguments. These functions are templated and, in addition to the left and right expressions, have two other default parameters. A bool value which states if the integral is volumetric or not and the type of quadrature rule, which is set by default to be Gaussian.

```
template<typename L,typename R,bool Volumetric=true,Integer QR=Gaussian>
auto L2Inner
(const Expression<Left>& left,const Expression<Right>& right)
{return L2DotProductIntegral<Left,Right,VolumeIntegral,QR>(left,right);}
```

```
template<typename L,typename R,Integer QR=Gaussian>
auto surface_integral
(const Integer label,const Expression<L>& left,const Expression<R>& right)
{return L2DotProductIntegral<L,R,false,QR>(left,right,label);}
```

The declaration of the `L2DotProductIntegral` class is:

```
template<typename L,typename R,bool Volumetric=true,Integer QR=Gaussian>
class L2DotProductIntegral:
public Expression<L2DotProductIntegral<L,R,Volumetric,QR>>
```

The most important information of this class are:

- the order of quadrature;
- the type of form (zero, one or two-form);
- in case of a one-form, the position of the FE space, corresponding to the test function, in `TupleOfSpaces`; this is used for the dofmap in the one-form;
- in case of a two-form, the positions of the FE spaces, corresponding to the test and trial functions, in `TupleOfSpaces`; these are used for the dofmaps in the two-form;

The order of quadrature Q of an L^2-inner product with arguments a and b, is defined as $Q((a,b)) = Q(a)Q(b)+1$. The quadrature order of the single terms $Q(a)$ and $Q(b)$ can be computed by using recursively the rules in Table 9.1.

$+u$	$-u$	$u+v$	$u-v$	$u*v$	u/v	(u,v)
$Q(u)$	$Q(u)$	$\max(Q(u),Q(v))$	$\max(Q(u),Q(v))$	$Q(u)+Q(v)$	$Q(u)-Q(v)$	$Q(u)+Q(v)$

Q	$I(u)$	$\nabla(u)$	$\mathrm{div}(u)$	$\mathrm{trace}(u)$
Lagrange	$Q(u)$	$Q(u)-1$	$-$	$Q(u)$
Raviart-Thomas	$Q(u)$	-	$Q(u)-1$	$Q(u)-1$

Table 9.1. Quadrature order rules for trees of expressions.

9.11 Linear and bilinear forms

Here we do propose the bilinear and the linear forms of the Signorini problem for the dual formulation of linear elasticity:

```
auto bilinearform=L2Inner(C_inverse(sigma),tau)+
                  L2Inner(Div(sigma),v)+
                  L2Inner(u,Div(tau))+
                  L2Inner(Asym(tau),theta)+
                  L2Inner(Asym(sigma),rho);
auto linearform=L2Inner(v,-f_ext)+
                surface_integral(dirichlet_label,Trace(tau),u_dirichlet)+
                surface_integral(contact_label,Inner(Trace(tau),normal),g);
```

See section 4.4 for the continuous formulation.

9.12 Essential boundary conditions

To enforce the N-th trial function to be equal to FuncType on the boundary Gamma, an essential boundary condition can be defined in this way:

```
auto bcs=DirichletBC<N,FuncType>(W_ptr,Gamma);
```

If it is necessary to prescribe only the M-th component of the trial function to be equal to FuncTypeSingleComponent, we write the following:

```
auto bcs=DirichletBC<N,FuncTypeSingleComponent,M>(W_ptr,Gamma);
```

A list of boundary conditions can then be used:

```
auto bcs1=DirichletBC<0,FunctionZero<ManifoldDim>>(W_ptr,neumann_boundary);
auto bcs2=DirichletBC<0,FunctionZero<1>,0>(W_ptr,contact_boundary);
```

9.13 FE context and assembly

Once the forms and boundary conditions have been defined, the corresponding context object can be created. The create_context() function takes the bilinear form and then, eventually, the linear form and/or the boundary conditions desired. Then the matrix $\mathbf{A}_h$ and vector $\mathbf{f}_h$ related to the bilinear and linear forms can be assembled by means of the assembly method.

```
auto context=create_context(bilinearform,linearform,bcs1,bcs2);
context.assembly(A_h,f_h);
```

9.14 Evaluation

Template expressions enable to define compile-time structures (the forms in this case) which can be later evaluated. The evaluation of the tree of expressions is carried by means of Evaluation objects. An Evaluation object can be built in general for unary and binary operators. In addition to the Unary or Binary classes, also other template arguments can be given as input. For example, the quadrature rule is a piece of additional and necessary information for the evaluation of the shape functions. Indeed shape functions as expressions are objects independent from the integrals in which they will be inserted. In fact, the same expression of a shape function can appear in different integrals terms. See for example (9.10a). Nevertheless, its evaluation strictly depends on the integral itself, which determines the quadrature rule. It is then obvious that we first must declare the tree of expressions and, only later, we can go for its evaluation, by considering the specific quadrature rule.

The Evaluation object of the forms cannot be built with the same structure of the tree of expressions. Indeed the tree of the forms involves integrals with different test and trial functions, which do not influence each other. We first need to decompose the overall form into sub-forms involving the same test or trial functions. Starting from the objects in (9.11), we need to create a matrix and a vector of sub-forms to be evaluated. For the dual formulation, we can decompose the evaluation of bilinear and linear forms into a matrix **E** and a vector **e** of evaluations as in (9.12). Even though it is a different example, we can notice the importance of using linearity in the FOSLS formulation (9.5). Indeed, for a decomposed evaluation like in (9.12), we need each $L^2(\Omega)$ dot product of a bilinear form to have only one test function and one trial function in its arguments. This is especially true if the trial or the tests belong to different spaces.

$$\mathbf{E} = \begin{bmatrix} (\mathcal{A}\boldsymbol{\sigma}, \boldsymbol{\tau})_\Omega & (\mathbf{u}, \mathrm{div}\ \boldsymbol{\tau})_\Omega & (\boldsymbol{\theta}, \mathbf{as}\ \boldsymbol{\tau})_\Omega \\ (\mathrm{div}\ \boldsymbol{\sigma}, \mathbf{v})_\Omega & 0 & 0 \\ (\mathbf{as}\ \boldsymbol{\sigma}, \boldsymbol{\gamma})_\Omega & 0 & 0 \end{bmatrix} \quad \mathbf{e} = \begin{bmatrix} (\boldsymbol{\tau}\mathbf{n}, \mathbf{g}_D)_{\Gamma_D} + ((\boldsymbol{\tau}\mathbf{n})_n, g)_{\Gamma_C} \\ (-\mathbf{f}, \mathrm{div}\ \boldsymbol{\tau})_\Omega \\ 0 \end{bmatrix} \tag{9.12}$$

The difficult part here is that, at compile-time, we need to define from the global types/object of (9.11), the local types/objects of the components in (9.12). We will not discuss how this is done, but want to stress out the fact that, even though is not trivial at all, it has been coded.

Once the Evaluation object is built, it can be evaluated by means of the apply method. The output to be returned is also the first input, passed as a reference, while all other variadic input arguments are useful for its computation.

```
template<template<class>class Unary,typename Derived,typename...Others>
class Evaluation<Expression<Unary<Expression<Derived> > >,Others...>
{
 public:
 using type=Unary<Expression<Derived>>;
 using Eval=Evaluation<Expression<Derived>,Others...>;
 using subtype= OperatorType<Derived,Others...>;
 using outputsubtype= OperatorType<type,Others...>;

 Evaluation(){};

 Evaluation(const Expression<type>& expr):
 expr_(expr.derived()),
 eval_(Eval(expr_()))
 {};

 template<typename...Inputs>
 void apply(outputsubtype& output, Inputs&...inputs)
 {

  eval_.apply(derived_value_,inputs...);
  OperatorApply<Unary<subtype>>::apply(output,derived_value_);
 }
       auto  expression()     {return expr_;}
 const auto& expression()const{return expr_;}

private:

 type expr_;
 Eval eval_;
 subtype derived_value_;
};
```

9.15 Collection of maps

In the assembly process, it is of fundamental importance the mapping between the reference and the actual elements. Depending on the element, on the FE family, and on the operator of interest, a different mapping is necessary. Therefore the arguments of the ReferenceMap class are the operator (identity, gradient, divergence, trace, and so on), the element, and the FE family. Such class has a private member map_ that is initialized by means of the init() method, which takes the finite element FE as input, and can be returned thanks to the function operator().

```
 template<typename Operator,typename Elem,Integer FEFamily>
 class ReferenceMap
  {
```

```
 public:
 inline constexpr void  init(const FiniteElem<Elem>& FE) {...}
 inline constexpr const auto&  operator() ()const{return map_;}

 private:
 MapType map_;
 };
```

In Table 9.2, we propose the maps for the Lagrangian and Raviart-Thomas families for simplicial elements and for different operators:

	Identity	Trace	Gradient	Divergence
Lagrangian	1	1	$\mathbf{A}^{-T}$	-
Raviart-Thomas	$\frac{\mathbf{A}}{\det(\mathbf{A})}$	$\frac{1}{\det(\mathbf{A})}$	-	$\frac{1}{\det(\mathbf{A})}$

Table 9.2. Mapping for different FE spaces and operators

The context object builds a MapFromReferenceCollection object for the bilinear and the linear forms. This again constructs two collections of maps, one for the volumetric operators, MapVolumetric, and the other one for the trace operators, MapSurface. To this aim, two tuples of tuples are needed and built as follows. Let us focus, for example, on the tuple of tuples MapVolumetric. It is no longer than the UniqueTupleOfSpaces. Indeed the maps require only the FE family integer labels. Additional information such as continuity or number of components is not necessary. Therefore, if in UniqueTupleOfSpaces can appear two continuous Lagrangian linear spaces which differ for the number of components, in MapVolumetric both spaces are represented by the only Lagrangian family.

For a FE family in the N-th position of MapVolumetric, the corresponding component in MapVolumetric is again a tuple that collects all the reference maps, related to the N-th FE family, necessary for the bilinear and the linear forms. Even though the same operator can show up multiple times in the forms, the tuples are reduced to their minimum, with no duplicate reference map. So for example, we do have:

```
using MapVolumetric=tuple<tuple<ReferenceMap<Divergence,Elem,RT>,
                                ReferenceMap<Identity,  Elem,RT>>,
                          tuple<ReferenceMap<Identity,  Elem,Lagrange>>>

using MapSurface=tuple<tuple<ReferenceMap<Trace,BoundaryElem,RT>>,
                       tuple<ReferenceMap<Trace,BoundaryElem,Lagrange>>>
```

inside the MapFromReferenceCollection class. In the dual formulation, the stress variable $\boldsymbol{\sigma}_h$ is subject to the identity and the divergence operators. Thus the first component of the tuple of type `MapVolumetric` is a tuple with two components associated to the `Divergence` and the `Identity` operators. On the other hand, the displacement $\mathbf{u}_h$ is a discontinuous piecewise linear functions of two components, while the rotation θ_h is a scalar continuous piecewise linear function. They share the same map, associated with the `Identity` operator, in the second component of the tuple of type `MapVolumetric`. The same argument is used for the surface tuple of tuples of type `MapSurface`.

9.16 Collection of shape functions

As for the maps, the classes `ShapeFunctionVolumetric` and `ShapeFunctionSurface` for the collection of the volumetric and surface shape functions can be created. The class `ShapeFunction` is fully determined by the element `E`, the `BaseFunctionSpace`, the operator `Operator` and quadrature rule `QR` types:

```
template<typename E,typename BaseFunctionSpace,typename Operator,typename QR>
class ShapeFunction;
```

Two are the main differences between the map and the shape functions collections. The first resides in the fact that the shape functions have to be evaluated in quadrature points, while the map is fixed for the given element (at least if we assume that the elements deform only linearly). The second one is that for the maps only the FE family is important, while for the shape functions the whole `BaseFunctionSpace` is. Therefore we can create a tuple of tuples, as long as the `UniqueTupleOfSpaces` tuple. The N-th component is a tuple referring to the N-th space of the tuple `UniqueTupleOfSpaces`. It contains all the occurrences of the shape function necessary for the bilinear and linear forms. If two shape functions are defined for the same space and the same operator, but different quadrature rules, then both of them will be considered. Nevertheless, as for the map case, no duplicate is allowed.

If we go back to the 2D dual formulation, the resulting collections are the following:

```
using ShapeFunctionVolumetric=
   tuple<tuple<ShapeFunction<Elem,RT1_2, Identity,  GaussPoints<Elem,5>>,
               ShapeFunction<Elem,RT1_2, Divergence,GaussPoints<Elem,3>>,
               ShapeFunction<Elem,RT1_2, Identity,  GaussPoints<Elem,4>>,
         tuple<ShapeFunction<Elem,DGP1_2 Identity,  GaussPoints<Elem,3>>>,
         tuple<ShapeFunction<Elem,P1_1,  Identity,  GaussPoints<Elem,4>>>,
         tuple<ShapeFunction<Elem,P1_2,  Identity,  GaussPoints<Elem,3>>>>
```

```
using ShapeFunctionSurface=
   tuple<tuple<ShapeFunction<Elem,RT1_2, Trace, GaussPoints<BoundaryElem,3>>,
               ShapeFunction<Elem,RT1_2, Trace, GaussPoints<BoundaryElem,4>>>,
         tuple< >,
         tuple<ShapeFunction<Elem,P1_1,  Trace, GaussPoints<BoundaryElem,4>>>,
         tuple<ShapeFunction<Elem,P1_2,  Trace, GaussPoints<BoundaryElem,3>>,
               ShapeFunction<Elem,P1_2,  Trace, GaussPoints<BoundaryElem,4>>>>
```

The first three components of the main tuples, `ShapeFunctionVolumetric` and `ShapeFunctionSurface`, refer to the `MixedFunctionSpace`, while the other ones refer to the `AuxiliarySpace`. For simplicity of notation, the number of components of the spaces is described with a subscript. For example, `RT1_2` means that we consider the RT_1 space with two components. In this case, two are the order of quadrature rules, 4 and 5, associated with the identity operator. In all other cases, only one quadrature rule is used. Finally, we see that the second component of `ShapeFunctionSurface` is an empty tuple because in the dual formulation there is no boundary trace of the displacement.

The collection of shape functions with their operators and quadrature rules is extended also to composite operators. Indeed, not only standard operators can be reused in the forms, but also the composite operators. Therefore also a tuple that enables their computation is necessary. Since the composite operators depend on the standard operators, first the shape functions related to standard operators are computed and, consequently, the composite operators can be computed as well.

9.17 Assembly of the FOSLS and dual formulations

In this final section, we exploit the information of the previous pages to code the assembly for both the FOSLS and the dual formulation. We consider the problem in Figure 8.22. The domain is a square mesh, with boundaries numbered from one to four, starting from the right edge and moving in a counterclockwise direction. Thus $\Gamma_N = \{1,3\}$, $\Gamma_C = \{2\}$, $\Gamma_D = \{4\}$. The gap function is defined as follows:

```
class GapFunction{
public:
using type=Matrix<Real,1,1>;
template<typename Point,typename FiniteElem>
static type eval(const Point& point,FiniteElem& FE)
{return +1.* 0.5*(1-sin(acos((point[0]-0.5)/0.5)));}
};
```

The code for the FOSLS and dual formulations is described below. Some parts, like the reading of the input mesh, the declaration of the matrix and the vector, or other minor details, are not inserted.

```
/////////////////////////////
///// FOSLS FORMULATION /////
/////////////////////////////
// Some type definitions
using MeshT=Mesh<Dim, ManifoldDim>;
using RT_0= FunctionSpace< MeshT, RT<0,MDim>>;
using P_1= FunctionSpace< MeshT, Lagrange<1,MDim>>;
using AuxP_1_1= FunctionSpace< MeshT,Lagrange<1,1>>;

// Mesh and auxiliary geometric objects
MeshT mesh;
Bisection<MeshT> bisection(mesh);
Node2ElemMap<MeshT> n2em(mesh,bisection);

// Single FE spaces
RT_0     rt0(mesh,bisection,n2em);
P_1      p1(mesh,bisection,n2em);
AuxP_1_1 p1_1(mesh,bisection,n2em);

// Collection of FE spaces
auto Wtrial = MixedFunctionSpace(rt0,p1);
auto Waux   = AuxFunctionSpacesBuild(p1,p1_1);
auto W      = FullSpaceBuild(Wtrial,Waux);

using W_type = decltype(W);
auto W_ptr   = std::make_shared<W_type>(W);

// Trial functions
auto sigma = MakeTrial<0>(W_ptr);
auto u     = MakeTrial<1>(W_ptr);

// Test functions
auto tau   = MakeTest<0>(W_ptr);
auto v     = MakeTest<1>(W_ptr);

// Auxiliary functions
auto f_ext  = MakeFunction<0,FunctionZero<MDim>>(W_ptr);
auto u_D    = MakeTraceFunction<0,FuncNthComp<MDim,1,Func001>>(W_ptr);
auto normal = MakeTraceFunction<0>(W_ptr);
auto gap    = MakeGapFunction<1>(W_ptr);

// Compute the normal values (at nodes or faces depending)
```

```
constexpr Real mu=1;
constexpr Real lambda=inf;
constexpr auto half=Constant<Scalar>(0.5);
constexpr Real alpha=1/(2mu);
constexpr Real beta=alpha (-lambda/(MDim lambda+2mu));
constexpr auto id_mat=Constant<Identity<MDim>>();
constexpr auto asym_mat=Constant<Mat<2,2>>(0.0,1.0, 0.0,0.0);
constexpr auto C1=Constant<Scalar>(1/(2 mu));
constexpr auto C2=Constant<Scalar>((1/(2mu))(-lambda/(MDim lambda+2mu)));

// Definition of asymmetric tensor
auto C_inv=NewOperator(C1 Identity() + C2 id_mat MatTrace(Identity()) );
// Definition of the compliance tensor
auto Asym=NewOperator(Inner(asym_mat,(Identity()-Transpose(Identity()))));
// Definition of the deformation tensor
auto Eps=NewOperator(half ((Gradient()+Transpose(Gradient()))));

// Bilinear form definition
auto b_form=+Int(Div(sigma),Div(tau))
            +Int(C_inv(sigma)-Eps(u),C_inv(tau)-Eps(v))
            +Surf_Int(GammaC,Inner(Trace(sigma),normal),Inner(Trace(v),normal)
            +Surf_Int(GammaC,Inner(Trace(tau),normal),Inner(Trace(u),normal));

// Linear form definition
auto l_form=+Int(-Div(tau),f_ext)
            +Surf_Int(GammaC,Inner(Trace(tau),normal),gap);

// Boundary conditions
auto bcs1=DirichletBC<0,FunctionZero<ManifoldDim>>(W_ptr,GammaN);
auto bcs2=DirichletBC<0,FunctionZero<1>,0>(W_ptr,contact_boundary);
auto bcs3=DirichletBC<0,FunctionZero<ManifoldDim>>(W_ptr,GammaN);
auto bcs4=DirichletBC<1, FuncNthComp<ManifoldDim,1,Func001> >(W_ptr,GammaD);

// Creation of the FE context
auto context=create_context(b_form,l_form,bcs1,bcs2,bcs3,bcs4);

// Matrix and right hand side
context.assembly(A,b);
```

```
////////////////////////////
///// DUAL FORMULATION /////
////////////////////////////
// Some type definitions
using MeshT=Mesh<Dim, ManifoldDim>;
using RT_1= FunctionSpace< MeshT, RT<1,MDim>>;
using DGP_1= FunctionSpace< MeshT, LagrangeDG<1,MDim>>;
using P_1= FunctionSpace< MeshT, Lagrange<1,MDim>>;
using AuxP_1_1= FunctionSpace< MeshT,Lagrange<1,1>>;

// Mesh and auxiliary geometric objects
MeshT mesh;
Bisection<MeshT> bisection(mesh);
Node2ElemMap<MeshT> n2em(mesh,bisection);

// Single FE spaces
RT_1     rt1(mesh,bisection,n2em);
DGP_1    dgp1(mesh,bisection,n2em);
P_1      p1(mesh,bisection,n2em);
AuxP_1_1 p1_1(mesh,bisection,n2em);

// Collection of FE spaces
auto Wtrial = MixedFunctionSpace(rt1,dgp1,p1_1);
auto Waux   = AuxFunctionSpacesBuild(p1,p1_1);
auto W      = FullSpaceBuild(Wtrial,Waux);

using W_type = decltype(W);
auto W_ptr   = std::make_shared<W_type>(W);

// Trial functions
auto sigma = MakeTrial<0>(W_ptr);
auto u     = MakeTrial<1>(W_ptr);
auto theta = MakeTrial<2>(W_ptr);

// Test functions
auto tau   = MakeTest<0>(W_ptr);
auto v     = MakeTest<1>(W_ptr);
auto rho   = MakeTest<2>(W_ptr);

// Auxiliary functions
auto f_ext  = MakeFunction<0,FunctionZero<MDim>>(W_ptr);
auto u_D    = MakeTraceFunction<0,FuncNthComp<MDim,1,Func001>>(W_ptr);
auto normal = MakeTraceFunction<0>(W_ptr);
auto gap    = MakeGapFunction<1>(W_ptr);

// Compute the normal values (at nodes or faces depending)

constexpr Real mu=1;
constexpr Real lambda=inf;
constexpr Real alpha=1/(2mu);
constexpr Real beta=alpha (-lambda/(MDim lambda+2mu));
constexpr auto id_mat=Constant<Identity<MDim>>();
constexpr auto asym_mat=Constant<Mat<2,2>>(0.0,1.0, 0.0,0.0);
constexpr auto C1=Constant<Scalar>(1/(2 mu));
constexpr auto C2=Constant<Scalar>((1/(2mu))(-lambda/(MDim lambda+2mu)));

// Definition of asymmetric tensor
auto C_inv=NewOperator(C1 Identity() + C2 id_mat MatTrace(Identity()) );
// Definition of the compliance tensor
auto Asym=NewOperator(Inner(asym_mat,(Identity()-Transpose(Identity()))));

// Bilinear form definition
auto b_form=+Int(C_inv(sigma),tau)
            +Int(Div(sigma),v)
            +Int(u, Div(tau))
            +Int(Asym(tau), theta)
            +Int(Asym(sigma),rho);

// Linear form definition
auto l_form=+Int(v,-f_ext)
            +Surf_Int(GammaD,Trace(tau),u_D)
            +Surf_Int(GammaC,Inner(Trace(tau),normal),gap);

// Boundary conditions
auto bcs1=DirichletBC<0,FunctionZero<MDim>>(W_ptr,GammaN);
auto bcs2=DirichletBC<0,FunctionZero<1>,0>(W_ptr,GammaC);
auto bcs3=DirichletBC<0,FunctionZero<ManifoldDim>>(W_ptr,GammaN);

// Creation of the FE context
auto context=create_context(b_form,l_form,bcs1,bcs2,bcs3);

// Matrix and right hand side
context.assembly(A,b);
```

Chapter 10

Conclusions

10.1 Summary

In this thesis, a MMG method has been introduced for the FOSLS and the dual stress-based formulations for the Signorini problem. In the FOSLS formulation, the displacement $\mathbf{u}$ and the stress $\boldsymbol{\sigma}$ are discretized respectively by means of linear Lagrangian and lowest-order Raviart-Thomas functions. In the dual formulation, the stress $\boldsymbol{\sigma}$, the displacement $\mathbf{u}$, and the rotation $\boldsymbol{\theta}$ are approximated respectively with first-order Raviart-Thomas, discontinuous linear Lagrangian, and continuous linear Lagrangian functions (in 2D). In contrast to the standard primal formulation, which computes only the displacement $\mathbf{u}$ and that is unbounded in the incompressible limit, the stress-based formulations always treat the stress $\boldsymbol{\sigma}$ as the main variable and can easily deal with incompressible materials. For these reasons, a working MMG method applied to a stress-based formulation of the Signorini problem would be able to solve, with optimal complexity, for linear elastic incompressible materials subject to inequality constraints.

The original idea of MMG is to sequentially minimize an energy functional by means of fine and coarse corrections fulfilling the constraints. This principle can be guaranteed only for the FOSLS formulation, since in the dual formulation the global equality constraints are just projected on the coarser levels. However, it is always possible to satisfy the local inequality constraints on coarser levels by using monotone restrictions operators. Since the constraints on the boundary can be discretized by means of linear functions, the monotone restrictions can be defined easily for both the FOSLS and the dual cases.

The same goes for the truncation of the basis functions related to the active degrees of freedom. This technique is used for accelerating convergence, but it is not beneficial for the FOSLS formulation, where essential boundary conditions are enforced on both the primal and the dual variables, $\mathbf{u}$, and $\boldsymbol{\sigma}$. Since in the dual formulation the boundary conditions concern only the stress variable $\boldsymbol{\sigma}$, truncation is effective as for the primal formulation, that enforces essentially the only boundary conditions on the displacement $\mathbf{u}$. However, for the dual formulation and, in particular, in the case of aggressive coarsening, the monotone restrictions did not perform very well in the numerical experiments. Thus the only truncation has been used.

In addition to monotone restrictions and truncation of the basis functions, a proper smoother must be introduced as well. In fact, $\boldsymbol{\sigma} \in \mathbf{H}_{\text{div}}$ and the smoother must be defined to tackle also the divergence-free components of the error. The Hiptmair smoother of section 6.12 is based on the Helmholtz decomposition and requires the computation of divergence-free corrections subject to global inequality constraints. The transformation from local to global inequality constraints is not desirable and so we opted for the Arnold-Falk-Winther smoother of section 6.13. The idea is to enlarge the local subdomains of the Gauss-Seidel smoother to entire patches so that divergence-free functions are tackled directly. In particular, we use a monolithic approach, meaning that we consider all possible degrees of freedom on the patch, not only the ones related to the stress.

In principle, for the FOSLS case, it could be possible to smooth separately the stress and the displacement, but adding to the stress patch also the degrees of freedom of the internal node is very simple. On the other hand, for the dual formulation, a monolithic approach is a necessary choice in the incompressible limit. Indeed the main bilinear form is only semi-positive definite and cannot be inverted unless the constraints are imposed as well. Therefore, on a patch, we can consider both the stress and the Lagrange multipliers dofs. Nevertheless, depending on how the boundary stress dofs are treated, we can recover a full Neumann, a full Dirichlet, or a full Robin problem. In the full Neumann case, the constraints are satisfied exactly, but rigid body motions are allowed. Thus, we must remove them, explicitly or by additional zero-average constraints on the Lagrange multipliers. Unfortunately, both attempts do not make the smoother satisfactory. On the other hand, the full Dirichlet approach, which incorporates also the stress dofs on the boundary, is more promising. The disadvantage is that some constraints outside the patch cannot be fulfilled. An idea could be to damp the boundary stress dofs of the local correction so that the constraints outside the patch can be violated in a minor way. This idea corresponds to a full Robin approach of parameter $\alpha \geq 0$. For $\alpha = 0$ we recover the full Dirichlet problem, while for $\alpha > 0$ local Robin boundary conditions are enforced.

We have shown in the numerical experiments that the parameter α is fundamental for the convergence of a two-level MMG method for linear elasticity, in case of aggressive coarsening. But it is also crucial for the convergence of the Signorini problem. The main issue of the Robin parameter α is that its optimal value is not known in advance. Therefore a dynamic computation of α should be carried out. In particular, we have proven that α cannot be modified from patch to patch, but it should remain constant at least for a whole smoothing step. Indeed the role of α is to make adequately communicate the various patch subdomains so that the overall error reduces as fast as possible. Optimization of α on a certain patch without considering all the other ones is not possible. Therefore α must be computed based on information obtained during the MMG process. In particular, among the strategies proposed, the best one works as follows. Given an initial value α_0, we keep it for the next V-cycle if the residual decrease is enough, otherwise, we reduce α_0. This strategy makes the algorithm robust with respect to α_0, if α_0 is chosen to be sufficiently large but less than one. Therefore we can state that we have defined an α_0-robust MMG method for the dual formulation applied to linear elastic nearly-incompressible or incompressible materials subject to rigid obstacle constraints.

10.2 Outlook

In this thesis, we have provided a robust MMG algorithm for solving nearly-incompressible or incompressible linear elastic obstacle problems for the dual formulation. All the numerical experiments are in 2D. Thus a natural generalization would be the 3D case. It would be also interesting to generalize the setting also to other local non-linearities, like plasticity or Coulomb friction, and see how the parameter α influences the performance. Furthermore, since the MMG here presented is geometric, future work can concern the treatment of domains with non-trivial shape. To this purpose, we would need to generalize the work done so far to the algebraic or semi-geometric multigrid frameworks and eventually to the extended FE method.

Bibliography

David Abrahams and Aleksey Gurtovoy. *C++ template metaprogramming: concepts, tools, and techniques from Boost and beyond*. Pearson Education, 2004.

Douglas Arnold, Richard Falk, and Ragnar Winther. Preconditioning in H(div) and applications. *Mathematics of Computation of the American Mathematical Society*, 66(219):957–984, 1997.

Douglas Arnold, Gerard Awanou, and Ragnar Winther. Finite elements for symmetric tensors in three dimensions. *Mathematics of Computation*, 77(263): 1229–1251, 2008.

Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Multigrid in H(div) and H(curl). *Numerische Mathematik*, 85(2):197–217, 2000.

Ragnar Winther Arnold Douglas, Richard S. Falk. Multigrid preconditioning in H(div) on non-convex polygons. *Computational and Applied Mathematics*, 17: 303–316, 1998.

I. Babuška, Fabio Nobile, and Raúl Tempone. Worst case scenario analysis for elliptic problems with uncertainty. *Numerische Mathematik*, 101(2):185–219, 2005.

Ivo Babuška. On randomised solutions of Laplace's equation. *Časopis pro pěstování matematiky*, 86(3):269–276, 1961.

Ivo Babuška. Error-bounds for finite element method. *Numerische Mathematik*, 16(4):322–333, 1971.

Ivo Babuška, Fabio Nobile, and Raúl Tempone. Reliability of computational science. *Numerical Methods for Partial Differential Equations: An International Journal*, 23(4):753–784, 2007a.

Ivo Babuška, Fabio Nobile, and Raúl Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007b.

Lori Badea. Convergence rate of a multiplicative Schwarz method for strongly nonlinear variational inequalities. In *IFIP International Information Security Conference*, pages 31–41. Springer, 2002.

Lori Badea. Global convergence rate of a standard multigrid method for variational inequalities. *IMA Journal of Numerical Analysis*, 34(1):197–216, 2014.

Lori Badea and Rolf Krause. One-and two-level Schwarz methods for variational inequalities of the second kind and their application to frictional contact. *Numerische Mathematik*, 120(4):573–599, 2012.

Harald Bader and Ronald HW Hoppe. *Multigrid solution of Signorini type problems in contact elastostatics*. Bibliothek der Fak. Mathematik und Informatik, Technische Univ., 1993.

Bank, Randolph E. and Yserentant, Harry. Multigrid convergence: a brief trip down memory lane. *Computing and visualization in science*, 13(4):147–152, 2010.

Markus Berndt, Thomas A. Manteuffel, and Stephen F. McCormick. Local error estimates and adaptive refinement for first-order system least squares (FOSLS). *Electron. Trans. Numer. Anal*, 6:35–43, 1997.

Markus Berndt, Thomas A. Manteuffel, Stephen F. McCormick, and Gerhard Starke. Analysis of first-order system least squares (FOSLS) for elliptic problems with discontinuous coefficients: Part I. *SIAM Journal on Numerical Analysis*, 43(1):386–408, 2005.

Bochev, Pavel B. and Gunzburger, Max D. *Least-squares finite element methods*, volume 166. Springer Science & Business Media, 2009.

Braess, Dietrich. *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2007.

Jan Brandts, Yanping Chen, and Julie Yang. A note on least-squares mixed finite elements in relation to standard and mixed finite elements. *IMA Journal of Numerical Analysis*, 26(4):779–789, 2006.

Brenner, Susanne and Scott, Ridgway. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2007.

Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*, volume 15. Springer Science & Business Media, 2012.

Cai, Zhiqiang and Lazarov, R. and Manteuffel, Thomas A. and McCormick, Stephen F. First-order system least squares for second-order partial differential equations: Part I. *SIAM Journal on Numerical Analysis*, 31(6):1785–1799, 1994.

Cai, Zhiqiang and Starke, Gerhard. Least-squares methods for linear elasticity. *SIAM Journal on Numerical Analysis*, 42(2):826–842, 2004.

Carstensen, Carsten and Scherf, Oliver and Wriggers, Peter. Adaptive finite elements for elastic bodies in contact. *SIAM Journal on Scientific Computing*, 20 (5):1605–1626, 1999.

Chen, Zhangxin. Multigrid algorithms for mixed methods for second order elliptic problems. 1994.

Ciarlet, Philippe G. The finite element method for elliptic problems. *Classics in applied mathematics*, 40:1–511, 2002.

Dostál, Zdenek. Box constrained quadratic programming with proportioning and projections. *SIAM Journal on Optimization*, 7(3):871–887, 1997.

Eck, Christof and Steinbach, Olaf and Wendland, Wolfgang L. A symmetric boundary element method for contact problems with friction. *Mathematics and Computers in Simulation*, 50(1-4):43–61, 1999.

Efstathiou, Evridiki and Gander, Martin J. Why restricted additive Schwarz converges faster than additive Schwarz. *BIT Numerical Mathematics*, 43(5):945–959, 2003.

Gander, Martin and Halpern, Laurence and Magoulès, Frédéric and Roux, François-Xavier. Analysis of patch substructuring methods. *International Journal of Applied Mathematics and Computer Science*, 17(3):395–402, 2007.

Gander, Martin J. and Halpern, Laurence and Magoules, Frédéric. An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation. *International journal for numerical methods in fluids*, 55 (2):163–175, 2007.

Gander, Martin J. and Vanzan, Tommaso. Heterogeneous optimized Schwarz methods for second order elliptic PDEs. *SIAM Journal on Scientific Computing*, 41(4):A2329–A2354, 2019.

Gander, Martin Jakob. Schwarz methods over the course of time. *Electronic transactions on numerical analysis*, 31:228–255, 2008.

Gelman, E. and Mandel, J. On multilevel iterative methods for optimization problems. *Mathematical Programming*, 48(1-3):1–17, 1990.

Gilbert, Alexander D. and Graham, Ivan G. and Kuo, Frances Y. and Scheichl, Robert and Sloan, Ian H. Analysis of quasi-Monte Carlo methods for elliptic eigenvalue problems with stochastic coefficients. *Numerische Mathematik*, 142 (4):863–915, 2019.

Giles, Michael B. Multilevel monte carlo methods. *Acta Numerica*, 24:259, 2015.

Gittelson, Claude Jeffrey. *Adaptive Galerkin methods for parametric and stochastic operator equations*. PhD thesis, ETH Zurich, 2011a.

Gittelson, Claude Jeffrey. An adaptive stochastic Galerkin method. In *Research report/Seminar für Angewandte Mathematik*, volume 2011. ETH Zurich, 2011b.

Harbrecht, Helmut and Peters, Michael D. and Schmidlin, Marc. Uncertainty quantification for PDEs with anisotropic random diffusion. *SIAM Journal on Numerical Analysis*, 55(2):1002–1023, 2017.

Hintermüller, Michael. Semismooth Newton methods and applications. *Department of Mathematics, Humboldt-University of Berlin*, 2010.

Hintermüller, Michael and Ito, Kazufumi and Kunisch, Karl. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13(3):865–888, 2002.

Hiptmair, Ralf. Multigrid method for H(div) in three dimensions. *Electron. Trans. Numer. Anal*, 6(1):133–152, 1997.

Hiptmair, Ralf and Toselli, Andrea. Overlapping and multilevel Schwarz methods for vector valued elliptic problems in three dimensions. In *Parallel solution of partial differential equations*, pages 181–208. Springer, 2000.

Hiptmair, Ralf and Xu, Jinchao. Nodal auxiliary space preconditioning in H(curl) and H(div) spaces. *SIAM Journal on Numerical Analysis*, 45(6):2483–2509, 2007.

Hoppe, Ronald HW. Multigrid algorithms for variational inequalities. *SIAM journal on numerical analysis*, 24(5):1046–1065, 1987.

Noboru Kikuchi and John Tinsley Oden. *Contact problems in elasticity: a study of variational inequalities and finite element methods*, volume 8. SIAM, 1988.

Axel Klawonn and Gerhard Starke. A preconditioner for the equations of linear elasticity discretized by the PEERS element. *Numerical linear algebra with applications*, 11(5-6):493–510, 2004.

Kober, Bernhard. *Symmetrie und Konformität in spannungsbasrierter FEM für Lineare Elastizität. Master's thesis, Fakult́at für Mathematik - Universität Duisburg-Essen*. PhD thesis, 2017.

Kober, Bernhard. Stress-based Finite Element Methods for Variational Inequalities in Contact Mechanics. 2020.

Tzanio V. Kolev and Panayot S. Vassilevski. Parallel auxiliary space AMG solver for H(div) problems. *SIAM Journal on Scientific Computing*, 34(6):A3079–A3098, 2012.

Kornhuber, Ralf. Monotone multigrid methods for elliptic variational inequalities I . *Numerische Mathematik*, 69(2):167–184, 1994.

Kornhuber, Ralf. Monotone multigrid methods for elliptic variational inequalities II. *Numerische Mathematik*, 72(4):481–499, 1996.

Kornhuber, Ralf and Krause, Rolf. Adaptive multigrid methods for Signorini's problem in linear elasticity. *Computing and Visualization in Science*, 4(1):9–20, 2001.

Kornhuber, Ralf and Krause, Rolf and Sander, O. and Deuflhard, P. and Ertel, S. A monotone multigrid solver for two body contact problems in biomechanics. *Computing and Visualization in Science*, 11(1):3–15, 2008.

Rolf Krause. A nonsmooth multiscale method for solving frictional two-body contact problems in 2d and 3d with multigrid efficiency. *SIAM Journal on Scientific Computing*, 31(2):1399–1423, 2009.

Krause, Rolf. *Monotone multigrid methods for Signorini's problem with friction*. PhD thesis, 2001.

Krause, Rolf and Müller, Benjamin and Starke, Gerhard. An adaptive least-squares mixed finite element method for the Signorini problem. *Numerical Methods for Partial Differential Equations*, 33(1):276–289, 2017.

Krause, Rolf and Rigazzi, Alessandro and Steiner, Johannes. A parallel multigrid method for constrained minimization problems and its application to friction, contact, and obstacle problems. *Computing and Visualization in Science*, 18(1): 1–15, 2016.

Kunisch, Karl. Semi-smooth Newton Methods for Non-Differentiable Optimization Problems. lipschitz lectures, 2008.

Mandel, Jan. A multilevel iterative method for symmetric, positive definite linear complementarity problems. *Applied Mathematics and Optimization*, 11(1):77–95, 1984.

Meyers, Scott. *Effective C++: 55 specific ways to improve your programs and designs*. Pearson Education, 2005.

Müller, Benjamin. *Mixed least squares finite element methods based on inverse stress-strain relations in hyperelasticity*. PhD thesis, Universitätsbibliothek Duisburg-Essen, 2015.

Multerer, MD. A note on the domain mapping method with rough diffusion coefficients. *Applied Numerical Mathematics*, 145:283–296, 2019.

Nocedal, Jorge and Wright, Stephen. *Numerical optimization*. Springer Science & Business Media, 2006.

Pasciak, Joseph E. and Wang, Yanqiu. A multigrid preconditioner for the mixed formulation of linear plane elasticity. *SIAM journal on numerical analysis*, 44 (2):478–493, 2006.

Pezzuto, Simone and Quaglino, Alessio and Potse, Mark. *On Sampling Spatially-Correlated Random Fields for Complex Geometries*, pages 103–111. 05 2019. ISBN 978-3-030-21948-2. doi: 10.1007/978-3-030-21949-9_12.

Quaglino, A. and Pezzuto, S. and Krause, R. Generalized Multifidelity Monte Carlo Estimators. *Preprint*, 2018.

Quarteroni, Alfio and Quarteroni, Silvia. *Numerical models for differential problems*, volume 2. Springer, 2009.

Rognes, Marie E. and Kirby, Robert C. and Logg, Anders. Efficient assembly of H(div) and H(curl) conforming finite elements. *SIAM Journal on Scientific Computing*, 31(6):4130–4151, 2009.

Schwab, Ch and Stuart, AM. Sparse deterministic approximation of Bayesian inverse problems, submitted. *arXiv preprint arXiv:1103.4522*, 2011.

Schwab, Ch and Todor, Radu Alexandru. Sparse finite elements for stochastic elliptic problems–higher order moments. *Computing*, 71(1):43–63, 2003.

Schwab, Christoph. High dimensional finite elements for elliptic problems with multiple scales and stochastic data. *arXiv preprint math/0305007*, 2003.

St-Cyr, Amik and Gander, Martin J. and Thomas, Stephen J. Optimized multiplicative, additive, and restricted additive Schwarz preconditioning. *SIAM Journal on Scientific Computing*, 29(6):2402–2425, 2007.

Gerhard Starke. Multilevel boundary functionals for least-squares mixed finite element methods. *SIAM journal on numerical analysis*, 36(4):1065–1077, 1999.

Gerhard Starke. An adaptive least-squares mixed finite element method for elasto-plasticity. *SIAM Journal on Numerical Analysis*, 45(1):371–388, 2007.

Gerhard Starke. Adaptive least squares finite element methods in elasto-plasticity. In *International Conference on Large-Scale Scientific Computing*, pages 671–678. Springer, 2009.

Gerhard Starke, Alexander Schwarz, and Jörg Schröder. Analysis of a modified first-order system least squares method for linear elasticity with improved momentum balance. *SIAM Journal on Numerical Analysis*, 49(3):1006–1022, 2011.

Starke, Gerhard. Gauss-Newton multilevel methods for least-squares finite element computations of variably saturated subsurface flow. *Computing*, 64(4): 323–338, 2000.

Trefethen, Lloyd N. Cubature, approximation, and isotropy in the hypercube. *SIAM Review*, 59(3):469–491, 2017.

Vandevoorde, David and Josuttis, Nicolai M. *C++ Templates: The Complete Guide, Portable Documents*. Addison-Wesley Professional, 2002.

Xu, Jinchao. Iterative methods by space decomposition and subspace correction. *SIAM review*, 34(4):581–613, 1992.

Xu, Jinchao. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56(3):215–235, 1996.

Xu, Jinchao and Chen, Long and Nochetto, Ricardo H. Optimal multilevel methods for H(grad), H(curl), and H(div) systems on graded and unstructured grids. In *Multiscale, non-linear and adaptive approximation*, pages 599–659. Springer, 2009.

Yang, Daoqi. *C++ and object-oriented numeric computing for scientists and engineers*. Springer Science & Business Media, 2000.

Yang, Suh-Yuh and Liu, Jinn-Liang. Least-squares finite element methods for the elasticity problem. *Journal of computational and applied mathematics*, 87(1): 39–60, 1997.

Zulian, Patrick and Ganellari, Daniel and Rovi, Gabriele. MARS - Mesh Adaptive Refinement for Supercomputing. Git repository, 2018. URL `https://bitbucket.org/zulianp/mars`.