A posteriori error estimates and adaptivity in numerical approximation of PDEs : regularization, linearization, discretization, and floating point precision

Ari Rappaport

THÈSE

PRÉSENTÉE À

SORBONNE UNIVERSITÉ

ÉCOLE DOCTORALE: Sciences Mathématiques de Paris Centre (ED 386)

Par **Ari RAPPAPORT**

POUR OBTENIR LE GRADE DE DOCTEUR

SPÉCIALITÉ: Mathématiques Appliquées

Estimations d'erreurs a posteriori et adaptivité en approximation numérique des EDPs : régularisation, linéarisation, discrétisation et précision en virgule flottante

---

Directeur de thèse : Martin Vohralík (Inria Paris)

Co-directeur de thèse : François Févotte (Triscale Innov)

Soutenue le: 22 mars 2024

Devant la commission d'examen formée de:

Sören BARTELS	Albert Ludwigs University of Freiburg	Rapporteur
Roland BECKER	Université de Pau et des Pays de l'Adour	Rapporteur
Clément CANCÈS	Inria Université de Lille	Examinateur
François FÉVOTTE	Triscale Innov	Co-encadrant
Yvon MADAY	Sorbonne Université	Président
Roberta TITARELLI	SUPMICROTECH, CNRS	Examinatrice
Martin VOHRALÍK	Inria Paris	Directeur de thèse
Thomas WIHLER	University of Bern	Examinateur



THESIS

PRESENTED AT THE

SORBONNE UNIVERSITY

DOCTORAL SCHOOL: Mathematical Sciences of Central Paris (ED 386)

By **Ari RAPPAPORT**

TO OBTAIN THE DEGREE OF DOCTOR OF PHILOSOPHY
SPECIALITY: Applied Mathematics

Adaptivity in modeling, nonlinear, and linear solvers and local conservation by post-processing. Applications to problems in underground transfer and storage.

Thesis advisor: Martin Vohralík (Inria Paris)
Thesis co-advisor: Françcois Févotte (Triscale Innov)

Defended on: March 22nd, 2024

In front of the examination committee consisting of:

Sören BARTELS	Albert Ludwigs University of Freiburg	Reviewer
Roland BECKER	Université de Pau et des Pays de l'Adour	Reviewer
Clément CANCÈS	Inria Université de Lille	Examiner
François FÉVOTTE	Triscale Innov	Thesis co-advisor
Yvon MADAY	Sorbonne Université	President
Roberta TITARELLI	SUPMICROTECH, CNRS	Examiner
Martin VOHRALÍK	Inria Paris	Thesis advisor
Thomas WIHLER	University of Bern	Examiner

To Carolyn, for always being there when I needed you most.
To my family for believing in and supporting me.
To my friends for all the laughs we've shared.
To my mentors for helping me shape my destiny.

SERENA

This thesis has been carried out within the research team SERENA, a joint project-team between **Inria Paris** and **Ecole des Ponts ParisTech**.

Inria
2 rue Simone Iff
75589 Paris, France

CERMICS
Ecole des Ponts
77455 Marne-la-Vallée, France

Résumé

Cette thèse se concentre principalement sur l'analyse d'erreur a posteriori et les algorithmes adaptatifs qui en découlent pour résoudre itérativement des équations aux dérivées partielles (EDP) non linéaires. Nous considérons des EDP de type elliptique et parabolique dégénéré. Nous étudions également l'adaptivité dans la précision en virgule flottante d'un solveur multigrille.

Dans les deux premiers chapitres, nous considérons des EDP elliptiques découlant d'un problème de minimisation d'énergie. L'analyse a posteriori est directement basée sur la différence d'énergie entre la solution vraie et approchée. Les applications non linéaires des EDP elliptiques que nous considérons sont en particulier fortement montones et Lipschitziennes. Dans ce contexte, une quantité importante est la « force de la non-linéarité » donnée par le rapport L/α où L est la constante de Lipschitz et α est la constante de (forte) montonicité.

Dans le Chapitre 1, nous étudions un algorithme adaptatif comprenant la régularisation, la discrétisation et la linéarisation adaptative. L'algorithme est appliqué à une EDP elliptique avec une non-linéarité non régulière. Nous établission une borne supérieure garantie, fondée sur un estimateur basé sur l'écart primal-dual. De plus, nous isolons les composantes de l'erreur correspondant à la régularisation, la discrétisation et la linéarisation qui conduisent à des critères d'arrêt adaptatifs. Nous prouvons que les estimateurs des composantes convergent vers zéro dans les limites respectives de régularisation, discrétisation et linéarisation de l'algorithme. Nous présentons des résultats numériques démontrant l'efficacité de l'algorithme. Nous présentons également des preuves numériques de robustesse par rapport au ratio mentionné L/α qui motive le travail dans le deuxième chapitre.

Dans le Chapitre 2, nous examinons la question de l'efficacité et de la robustesse de l'estimateur d'erreur de l'écart primal-dual. Nous considérons en particulier une différence d'énergie augmentée, pour laquelle nous établissons une indépendance vis-à-vis de ce ratio pour la linéarisation de Zarantonello et seulement une dépendance locale et calculable par patch pour d'autres méthodes de linéarisation, y compris la linéarisation de Newton. Des résultats numériques sont présentés pour étayer les développements théoriques.

Dans le Chapitre 3, nous nous tournons vers le problème de régularisation adaptative pour l'équation de Richards. L'équation de Richards apparaît dans le contexte de la modélisation des milieux poreux. Elle contient des non-linéarités non-régulières, et se prêtant à la même approche que celle adoptée dans le Chapitre 1. Nous développons des estimateurs, ici inspirés des estimateurs en norme duale du résidu, ainsi qu'un algorithme adaptatif basé sur ces estimateurs.

Dans le Chapitre 4, nous fournissons des détails sur la mise en œuvre efficace du flux équilibré : un ingrédient crucial dans le calcul des estimateurs d'erreur. La mise en œuvre s'appuie sur le paradigme du multi-threading dans le langage de programmation Julia. Une boucle supplémentaire est introduite pour éviter les allocations de mémoire, essentielle pour obtenir un passage à l'échelle parallèle.

Dans le Chapitre 5, nous considérons un algorithme de précision mixte avec une méthode de multigrille géométrique comme solveur interne. Le solveur multigrille fournit intrinsèquement un estimateur d'erreur que nous utilisons dans le critère d'arrêt pour le raffinement itératif. Nous présentons un benchmark pour démontrer l'accélération obtenue en utilisant des représentations en simple précision des matrices creuses impliquées. Nous concevons également un algorithme adaptatif qui utilise l'estimateur mentionné pour identifier quand le raffinement itératif échoue pour des problèmes trop mal conditionnés, l'algorithme est alors capable de récupérer et de résoudre le problème entièrement en double précision.

Mots-clés : Équations aux dérivées partielles, non linearité fortement monotone et Lipschitzienne, méthode des éléments finis, linéarisation itérative non linéaire, méthode de Newton, estimation d'erreur a posteriori, adaptivité, régularisation, multigrille, équation de Richards, milieux poreux, précision mixte, raffinement itératif

Abstract

This thesis concerns a posteriori error analysis and adaptive algorithms to approximately solve nonlinear partial differential equations (PDEs). We consider PDEs of both elliptic and degenerate parabolic type. We also study adaptivity in floating point precision of a multigrid solver of systems of linear algebraic equations.

In the first two chapters, we consider elliptic PDEs arising from an energy minimization problem. The a posteriori analysis therein is based directly on the difference of the energy in the true and approximate solution. The nonlinear operators of the elliptic PDEs we consider are strongly monotone and Lipschitz continuous. In this context, an important quantity is the "strength of the nonlinearity" given by the ratio L/α where L is the Lipschitz continuity constant and α is the (strong) monotonicity constant.

In Chapter 1 we study an adaptive algorithm comprising adaptive regularization, discretization, and linearization. The algorithm is applied to an elliptic PDE with a nonsmooth nonlinearity. We derive a guaranteed upper bound based on primal-dual gap based estimator. Moreover, we isolate components of the error corresponding to regularization, discretization, and linearization that lead to adaptive stopping criteria. We prove that the component estimators converge to zero in the respective limits of regularization, discretization, and linearization steps of the algorithm. We present numerical results demonstrating the effectiveness of the algorithm. We also present numerical evidence of robustness with respect to the aforementioned ratio L/α which motivates the work in the second chapter.

In Chapter 2, we consider the question of efficiency and robustness of the primal-dual gap error estimator. We in particular consider an augmented energy difference, for which we establish independence of the ratio L/α (robustness) for the Zarantonello linearization and only patch-local and computable dependence for other linearization methods including the Newton linearization. Numerical results are presented to substantiate the theoretical developments.

In Chapter 3 we turn our attention to the problem of adaptive regularization for the Richards equation. The Richards equation appears in the context of porous media modeling. It contains nonsmooth nonlinearities, which are amenable to the same approach we adopt in Chapter 1. We develop estimators and an adaptive algorithm where the estimators are inspired by estimators based on the dual norm of the residual. We test our algorithm on a series of numerical examples coming from the literature.

In Chapter 4 we provide details for an efficient implementation of the equilibrated flux, a crucial ingredient in computing the error estimators so far discussed. The implementation relies on the multi-threading paradigm in the Julia programming language. An additional loop is introduced to avoid memory allocations, which is crucial to obtain parallel scaling.

In Chapter 5 we consider a mixed precision iterative refinement algorithm with a geometric multigrid method as the inner solver. The multigrid solver inherently provides an error estimator of the algebraic error which we use in the stopping criterion for the iterative refinement. We present a benchmark to demonstrate the speedup obtained by using single precision representations of the sparse matrices involved. We also design an adaptive algorithm that uses the aforementioned estimator to identify when iterative refinement in single precision fails and is able to recover and solve the problem fully in double precision.

Keywords: Partial differential equations, Strongly monotone and Lipschitz continuous nonlinear operators, finite element method, iterative linearization, Newton method, a posteriori error estimate,

adaptivity, regularization, multigrid, Richards equation, porous media, mixed precision, iterative refinement

"Quality is not an act, it is a habit."
- Aristotle

Acknowledgements

The contents of this thesis correspond to three years of work as a PhD student in the SERENA team at Inria Paris. I can hardly believe that period has come to an end and that it's time for the next step in my journey.

First and foremost, I would like to express my deepest gratitude to Martin Vohralík, my thesis director. From our very first meeting, I could see your passion, and you have never ceased to be a source of inspiration for me. Your knowledge and expertise of numerical analysis is really exceptional and I feel honored to have been your student. Your dedication to excellence have inspired me to be a better version of myself. You are also incredibly kind and patient, which was very much appreciated.

I am also immensely grateful to my co-supervisor François Févotte. Your logical approach and optimistic mindset helped me overcome some of the most difficult obstacles during my thesis. Thanks to you, I have learned to look for solutions and always do my best. I feel honored to have authored my first Julia package with you, and I hope it won't be the last.

I was honored to have Sören Bartels and Roland Becker as my thesis reviewers. They are both amazing scientists and to have them read my work was very special to me. Their reports were both insightful and very kind. I would also like to thank the other members of my committee, Clément Cancès, Roberta Titarelli, Yvon Maday, and Thomas Wihler. They all shared their unique point of view and asked very deep questions that lead to fruitful discussions.

Next, I would also like to give an enormous thanks to Zhaonan Dong, a permanent member of the SERENA team. You are so talented and your expertise spans from the most theoretical analysis to the tiniest implementation detail. I felt that whatever problem I had, your door was open and you would enthusiastically help me out. I also enjoyed our (non-Bercy) lunches together as well as our various outings to restaurants in Paris. This thesis would not have been possible without you.

On a personal level, I would like to thank all the other present and former students, postdocs and permanent members at Inria. With Zuodong, Clément and Gregor: I will never forget our climbing/kebab sessions. André and Koondi, I enjoyed our mathematical discussions and I learned a lot from both of you. To the other members of the SERENA team Alexandre, Jean-Luc, Géraldine, Michel, François, Alessandra, Romain, Morgane, Akrahm, Simon, Raphaël, thank you for the moments shared together. You all really made me feel welcome and I hope we can stay in touch. I would also like to thank my friends from the DYOGENE team including Alessia and Roman. It was getting to know you and I think we were able to create a cross-team community which was awesome (along with Coco and Élise from MATHRISK).

I would also like to thank my family (Josh, Kathy and Ella) for all their support and kindness. I looked forward to our video calls, as well as my visits back. When I always miss the beautiful New Mexico sky and the taste of fresh green chile.

Last but certainly not least, I would like to thank my dear girlfriend, Caro. It has been a wild couple of years with a lot of challenges and we have both persevered. You are always so sweet and thoughtful and strong. I am really proud of both of us and I think we both grew a lot as people during this period. I look forward to our next chapter together!

Contents

"*C'est avec la logique que nous prouvons*
et avec l'intuition que nous trouvons."
- Henri Poincaré

Introduction

In the modern age, many fields of science and engineering rely increasingly on numerical simulation to gain insight into complex phenomena. Numerical experiments have several advantages over conventional laboratory experiments.

- Cost: Numerical experiments are almost always more cost effective than their physical counterparts.
- Reproducibility: Numerical experiments provide a higher degree of reproducibility due to the deterministic nature of computers.
- Generality and extensibility: The same numerical methods can be applied to a large class of physical models as well as taxonomies of models with increasing complexity.

Furthermore, in certain cases, physical experiments may be dangerous or even impossible. This is indeed the case for the French agency for nuclear waste management (ANDRA) that funded this thesis.

i Context and motivation

Originally founded in 1979 as a branch of the atomic energy center (CEA), ANDRA was established as an independent organization in 1991. Immediately following its independence, the 1991 Waste Act tasked ANDRA with determining the feasibility of a deep geological repository for high-level and long-lived intermediate-level radioactive waste. One of the major culminations of ANDRA's work in this direction is the Cigéo (Centre industriel de stockage géologique) project. The goal of this ambitious project is to provide a solution for nuclear waste disposal that is viable for at least 100 years. The plan is to bury steel casks containing the waste around 500 meters below the ground. The main facility (see Figure 1) located in the Meuse/Haute-Marne region was chosen carefully based on its geological characteristics. In particular, this region is home to vast clay deposits that can potentially insulate the surface in the case of radioactive seepage.

Indeed, understanding the consequences of potential leaks of radioactive material is of paramount importance to ANDRA. In a worst-case scenario, the groundwater could corrode the steel casks and transport the radioactive material away from the containment zone. To understand and mitigate such risks, ANDRA conducts large-scale simulations over very long timescales (thousands of years) in addition to physical testing at the facility. The underlying mathematical models involve modeling *flow in porous media* such as clay or rock. These porous media models are often highly complex, with parameters varying over large space and timescales, as well as containing highly nonlinear and/or nonsmooth closure laws.

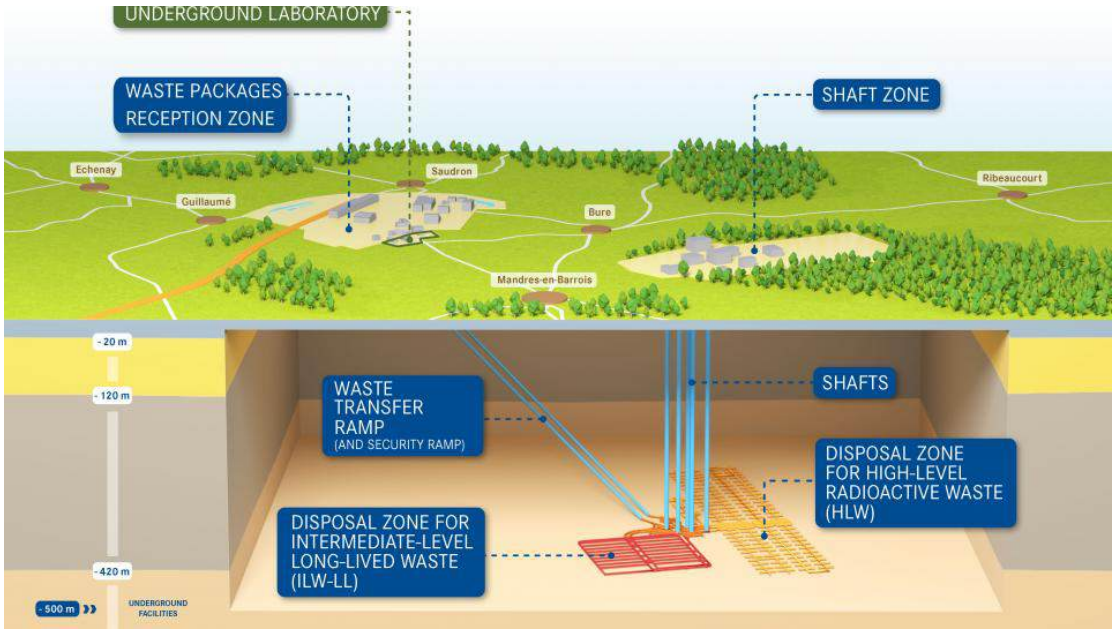


Figure 1: Cutaway view of the main Cigéo site (courtesy of andra.fr).

This thesis is primarily concerned introducing new methods for treating difficulties arising from the form of nonlinear nonsmooth closure laws, and the subsequent study of the resulting adaptive algorithms. The central tools will be **a posteriori error analysis** and **adaptivity**, which we detail in the forthcoming sections. In particular, in Chapters 1 and 2 we study an academic model problem related to energy minimization while in Chapter 3 we study the Richards equation which is a reduced model of two-phase flow in porous media with highly nonlinear and nonsmooth closure laws. Implementation aspects are considered in Chapter 4 and we explore mixed precision floating point adaptivity in Chapter 5.

ii Goals of this thesis

Accurate simulation of nonlinear nonsmooth partial differential equations (PDEs) such as those appearing in porous media flows can be prohibitively expensive in terms of computational resources, whether it be in terms of mesh resolution, temporal resolution, or number of solver iterations for the resulting linear or nonlinear systems of algebraic equations. These steps will be described in greater details in the following but for now let us simply observe how each approximation comes with a set of parameters allowing to increase the accuracy, typically at the detriment of the computational cost:

- A spatial discretization method–here the finite element method (FEM)–allows approximating the continuous PDE by a discrete problem, see §iv.2. The approximation involved is governed by the chosen mesh, which can be refined in order to make the solution more accurate.

- A linearization procedure allows handling the nonlinear nature of the problem at hand as in §iv.3, transforming it into a sequence of linear problems. In this case the approximate solution can be made more accurate by hardening the stopping criterion so that

the number of iterations increases. A regularization procedure will be introduced to handle non-smooth closure laws (see §iv.1). Its accuracy is governed by a parameter, which can be made smaller to let the regularized closure law converge to the exact non-smooth law.

- Even when, after all these steps, only linear systems of algebraic equations remain to be solved (see §v.1), some choices can still be made to control the accuracy and speed of the linear solver. One such choice is the stopping criterion used for an iterative solver. Another choice is the precision of the arithmetic used for the underlying floating point operations (see §v.2).

For hard problems such as the Richards equation, it becomes of paramount importance to try and spend the computational effort to increase the accuracy of the weakest link in the chain, and to try and avoid wasting computational resources on approximations that do not actually influence the overall solution accuracy.

In such a context, the natural question then becomes: how can we steer the overall algorithm that chains all these approximations? In particular, how to decide, at a certain step of the algorithm, whether to, e.g., refine the mesh or reduce the regularization? At a given linearization iteration, is it enough to use a single-precision linear solver, or is double precision needed? Or can a mixed-precision approach provide the required accuracy?

The main unifying feature of our work here is to propose methods that free the user from the burden of choosing appropriate parameters to control the accuracy of each step in the algorithm. Instead we propose methods that **automatically steer the overall algorithm** to spend computational resources where it matters most. Our main strategy will be to

1. estimate the error with a **computable a posteriori error estimator** (see §vi);
2. decompose the estimator into **components** corresponding to the **different parts** of the **solution procedure**, and
3. **balance these components**.

This therefore leads us to

- **Adaptivity** in **discretization** by local mesh refinement,
- **Adaptivity** in **regularization** by updating a regularization parameter,
- **Adaptivity** in **linearization** by adaptive stopping criteria,
- **Adaptivty** in **floating point precision** in the linear solver using iterative refinement.

iii Mathematical models

The overarching mathematical theme of this thesis are strongly nonlinear and nonsmooth partial differential equations (PDEs).

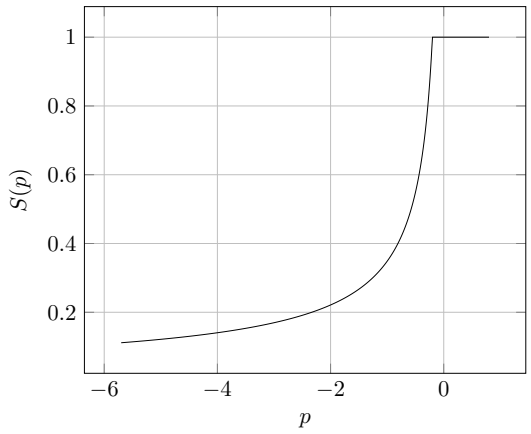


Figure 2: The nonlinear saturation function for the Brooks–Corey model [25]. The function is nonsmooth as it is not differentiable at the point $p = -0.2$.

iii.1 The Richards equation

An important model in the context of porous media modeling is the Richards equation [33, 39]. The Richards equation provides a reduced model of two-phase flow. The equation reads, for unknown pressure p and saturation s,

$$\partial_t s - \nabla\cdot[\mathbf{K}\kappa(s)(\nabla p + \boldsymbol{g})] = f \quad \text{in } \Omega \times [0,T], \tag{1}$$

where Ω is a spatial domain, T is the final time, $\mathbf{K}$ is the absolute permeability tensor, κ is the relative permeability function, $-\boldsymbol{g}$ represents the constant force of gravity, and f is a body source term. The equation is closed by the so-called capilary pressure relation

$$s = S(p). \tag{2}$$

We note that the functions $\kappa : [0,1] \to \mathbb{R}^+$ and $S : \mathbb{R} \to [0,1]$ are nonlinear in practical models. We study the Richards equation in more detail in Chapter 3. In the case where $S(p)$ is constant so that $\partial_t s = 0$, and if $\boldsymbol{g} = \mathbf{0}$ and $\mathbf{K} = \mathbf{I}$, the Richards equation is a stationary equation for the pressure p, namely,

$$-\nabla\cdot(\kappa(S(p))\nabla p) = f \quad \text{in } \Omega. \tag{3}$$

In Figure 2, we show a nonlinear and nonsmooth example of the saturation function $S(p)$. Degenerate functions κ are also considered in Chapter 3.

iii.2 Energy minimization

In Chapters 1–2 we study equations similar to (3), namely PDEs of the form

$$-\nabla\cdot(a(|\nabla u|)\nabla u) = f \quad \text{in } \Omega, \tag{4}$$

where the function $a : \mathbb{R}^+ \to \mathbb{R}^+$ is nonlinear and possibly nonsmooth. We assume additional structure in that there exists a convex function $\phi \in C^1(\mathbb{R})$ such that

$$a(s) = \frac{\phi'(s)}{s}. \tag{5}$$

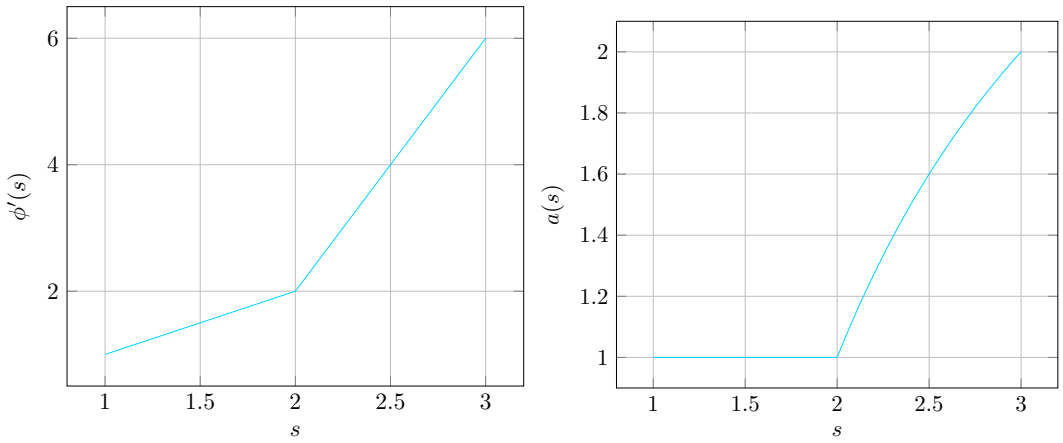


Figure 3: The nonlinear and nonsmooth function a and the corresponding function ϕ'. They are both non-differentiable at $s = 2$.

For some appropriate Sobolev space V, our problem can then be stated as

$$u = \arg\min_{v \in V} \int_\Omega \phi(|\nabla v|) - fv \, \mathrm{d}\boldsymbol{x}. \tag{6}$$

This is further equivalent to the so-called Euler–Lagrange equations: find $u \in V$ such that

$$(a(|\nabla u|)\nabla u, \nabla v) = (f, v) \qquad \forall v \in V, \tag{7}$$

whereby (4) is the strong form of (7).

iv Solution strategy

iv.1 Regularization of nonsmooth nonlinearities

As mentioned above, we are interested in solving problems where nonsmooth nonlinear functions appear cf. Figure 2 and Figure 3. We in particular seek to avoid poor iterative linearization solver performance due to low regularity. Our recipe is to regularize the nonsmooth nonlinearities. Regularization is a well known approach in many circumstances for handling low regularity. For example, for inverse problems [73, 98, 48], regularization is a common tool to deal with issues pertaining to ill-posedness. Closer to the current setting, regularization has been used as a theoretical tool for establishing existence and uniqueness in the context of degenerate parabolic PDEs [100, 43, 96], a class that the Richards equation (1) belongs to. Degenerate parabolic equations are characterized by "degeneracies", i.e., points in either space or time where the PDE changes type (from parabolic to either elliptic or hyperbolic in the present setting). In [100] and [96], sequences of regularized problems are introduced that avoid these degenerate regions. These regularized problems are always parabolic, and thus amenable to standard methods to prove well-posedness. Next, it is shown that the limit of the regularized solutions converges in some sense to the solution of the original problem.

Here, we are rather interested in how to improve nonlinear solvers performance, and in particular Newton's method. For Newton-type methods, regularization (smoothing) Newton methods replace non-differentiable nonlinearities with smooth counterparts, see [122, 103,

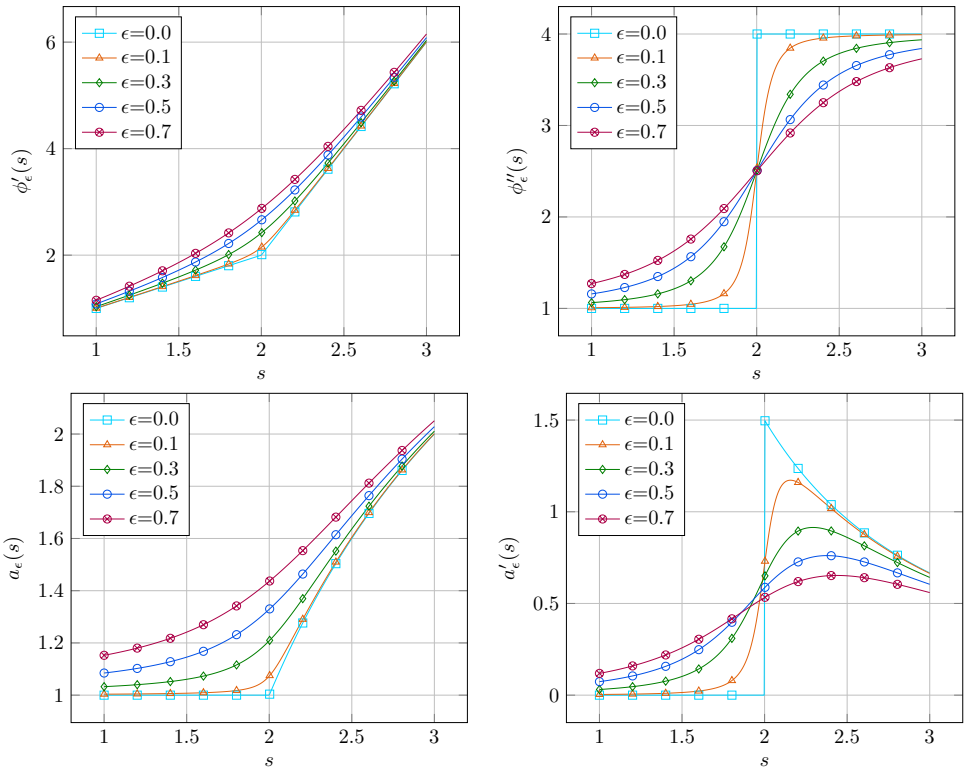


Figure 4: Regularized approximatations of the nonlinear and nonsmooth functions a and ϕ in the energy minimization problem of (12).

102] and the references therein. In this case, the amount of added regularization is typically proportional to a parameter that is driven to zero as the Newton iterations progress, thereby approaching the original problem.

In our context, for the given nonsmooth nonlinear function a and the associated function ϕ, we introduce a sequence of positive regularization parameters $\{\epsilon^j\}_{j\geq 1}$ that gives rise to a sequence of regularized functions a_{ϵ^j} and ϕ_{ϵ^j}, see Figure 4 for an example in the energy minimization setting where $a \in C^0(\mathbb{R}) \setminus C^1(\mathbb{R})$ and correspondingly the convex function $\phi \in C^1(\mathbb{R}) \setminus C^2(\mathbb{R})$. In this case, we obtain a sequence of modified weak problems for $j \geq 1$: find $u^j \in V$ such that

$$(a_{\epsilon^j}(|\nabla u^j|)\nabla u^j, \nabla v) = (f, v) \quad \forall v \in V. \tag{8}$$

Our salient point is that we will be **choosing ϵ^j adaptively using a posteriori error estimators**. This approach is also relevant when we study the Richards equation in more detail in Chapter 3, in particular for the Brooks–Corey model for which the saturation function (2) contains a kink point cf. Figure 3. Figure 5 gives an example of our proposed regularization in this context. We will again be choosing ϵ^j adaptively.

iv.2 Finite element discretization

Problem (8) is still infinite-dimensional. To discretize it, we consider the continuous Galerkin (Lagrange) finite element method of arbitrary order, see e.g., [50]. More precisely, given a conforming, shape-regular mesh $\mathcal{T}_\ell$ of the domain Ω, our discrete approximate space, for polynomial order $p \geq 1$, will be $V_\ell^p := \mathbb{P}_p(\mathcal{T}_\ell) \cap H_0^1(\Omega)$. In particular, the choice of notation

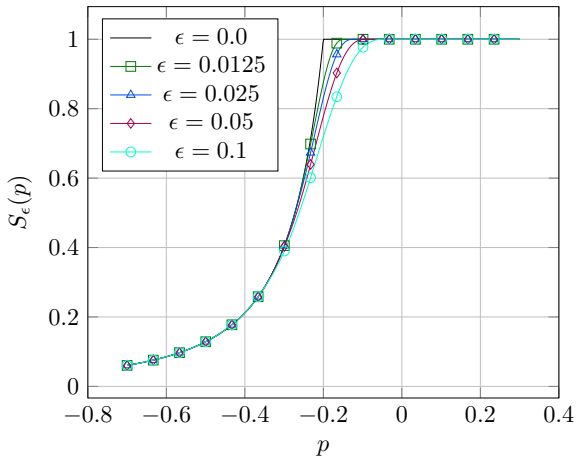


Figure 5: Regularized approximations for the saturation function (2) in the Brooks–Corey model [25] cf. (3.7).

$\ell \in \mathbb{N}$ rather than $h \in \mathbb{R}$ reflects the fact that we will work with sequences of meshes that are typically graded and are generated from an adaptive mesh refinement (AMR) procedure. We thus consider a regularized and discretized problem associated to (7): for a mesh index ℓ and a regularization index j, find $u_\ell^j \in V_\ell^p$ such that

$$(a_{\epsilon^j}(|\nabla u_\ell^j|)\nabla u_\ell^j, \nabla v_\ell) = (f, v_\ell) \quad \forall v_\ell \in V_\ell^p. \tag{9}$$

iv.3 Iterative linearization

Once the problem has been regularized and discretized, we consider a sequence of linear problems to solve, whose solutions approximate the solution u_ℓ^j to the nonlinear problem (9). We refer to this procedure as iterative linearization. For general nonlinear PDE, there exist many well known linearization procedures, e.g. the Picard/fixed point iteration, the Zarantonello iteration, and the Newton iteration, see [20, 42, 50, 67, 80, 129] and the references therein. However, Newton's method stands apart in providing potentially quadratic convergence under certain conditions. Thus, as stated in §iv.1, we seek to satisfy the necessary conditions for Newton's method (in particular the regularity requirements) by introducing an adaptive regularization. For a mesh index $\ell \geq 1$ and a regularization index $j \geq 0$, given an initial guess $u_\ell^{j,0}$, we can consider a regularized, discretized, and linearized problem associated to (7): for a linearization index $k \geq 1$, find $u_\ell^{j,k} \in V_\ell^p$ such that

$$(\boldsymbol{A}_\ell^{j,k-1}\nabla u_\ell^{j,k}, \nabla v_\ell) = (f, v_\ell) + (\boldsymbol{b}_\ell^{j,k-1}, \nabla v_\ell) \quad \forall v_\ell \in V_\ell^p, \tag{10}$$

where $\boldsymbol{A}_\ell^{j,k-1} : \Omega \to \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_\ell^{j,k-1} : \Omega \to \mathbb{R}^d$ depend on the previous iterate $u_\ell^{j,k-1}$ and on the chosen linearization.

We now present three typical choices of the linearization (10):

- The Picard (fixed point) iteration, see, e.g. [42], is defined as

$$\boldsymbol{A}_\ell^{j,k-1} = a(|\nabla u_\ell^{j,k-1}|)\boldsymbol{I}_d \quad \text{and} \quad \boldsymbol{b}_\ell^{j,k-1} = \boldsymbol{0} \quad \text{in } \Omega. \tag{a}$$

- The Zarantonello iteration, introduced in [127], is defined as

$$\boldsymbol{A}_\ell^{j,k-1} = \gamma \boldsymbol{I}_d \quad \text{and} \quad \boldsymbol{b}_\ell^{j,k-1} = \left(\gamma - a(|\nabla u_\ell^{j,k-1}|)\right)\nabla u_\ell^{j,k-1} \quad \text{in } \Omega, \tag{b}$$

where $\gamma \in (0, \infty)$ is a constant parameter. The Zarantonello iteration converges linearly, but the convergence is slow as γ takes large values.

- The Newton iteration, see, e.g. [42], is defined as

$$\begin{aligned} \boldsymbol{A}_\ell^{k-1} &= a(|\nabla u_\ell^{j,k-1}|)\boldsymbol{I}_d + \frac{a'(|\nabla u_\ell^{j,k-1}|)}{|\nabla u_\ell^{j,k-1}|}\nabla u_\ell^{j,k-1} \otimes \nabla u_\ell^{j,k-1} \\ \text{with } \boldsymbol{b}_\ell^{k-1} &= a'(|\nabla u_\ell^{j,k-1}|)|\nabla u_\ell^{j,k-1}|\nabla u_\ell^{j,k-1} \text{ in } \Omega. \end{aligned} \tag{c}$$

Specifically for the Richards equation, the design of effective linearization procedures is an ongoing effort. A sufficient condition for the convergence of Newton's method in the context of the Richards equation was derived in [75]. Other linearization schemes specific to the Richards equation include the modified Picard method [31], L-schemes [111, 87, 99, 92], and the Jäger–Kačur method [77, 78]. These methods are generally more robust than Newton's method at the cost of slower convergence. In particular, the L-scheme was shown to be unconditionally convergent in [105], though it only converges linearly. In our philosophy we want to keep the Newton method but rather apply it to a suitably (adaptively) regularized nonlinearity.

v Linear solvers, iterative refinement, and mixed precision

v.1 Linear solvers

Once we have reduced the original problem to a sequence of finite dimensional linear problems of the form (10), we can choose a basis of the finite-dimensional space V_ℓ^p and then the discrete linear problem (10) is equivalent to a linear system of the form $Ax = b$. One possible choice to solve these types of equations are *direct* methods, i.e., procedures to produce the exact solution x of the linear system (up to machine precision) in a predetermined number of steps, see, e.g., [62, 126, 3] and references therein. We will however, be primarily interested in *iterative* solvers, where a sequence of approximate algebraic vectors are computed that converge to the true solution x. In the model context (10), this would give us an approximate solution $u_\ell^{j,k,i} \in V_\ell^p$ on each iteration i of the linear solver.

Classical iterative methods include fixed point-type methods such as Jacobi, Gauss–Seidel, and successive over-relaxation. An important class of solution methods are Krylov subspace methods such as the conjugate gradient method (CG) and generalized minimum residual methods (GMRES), see, e.g., [110, 80]. Another branch of linear solvers are so-called multi-level methods. These methods rely on information from the underlying discretized system (namely, the mesh the problem is discretized on) to consider auxiliary versions of the original problem with fewer unknowns. More precisely, we are interested in geometric multigrid methods, see, e.g., [65, 24]. A typical geometric multigrid algorithm consists of a so-called V-cycle, see Figure 6. A sequence of grids is established, and discrete functions are transferred between them. The operations performed on the individual levels (grids) are referred to as *smoothing*, usually consisting of an iterative fixed point method, and a *coarse solve*: a direct method on the lowest level (coarsest grid). The smoothing steps are called pre-smoothing (resp. post) because they occur before (resp. after) the coarse solve, see Figure 6.

In particular, we will use the geometric multigrid solver introduced in [91, 89] and in the PhD thesis [90]. This version of geometric multigrid has several advantages: 1) there are no

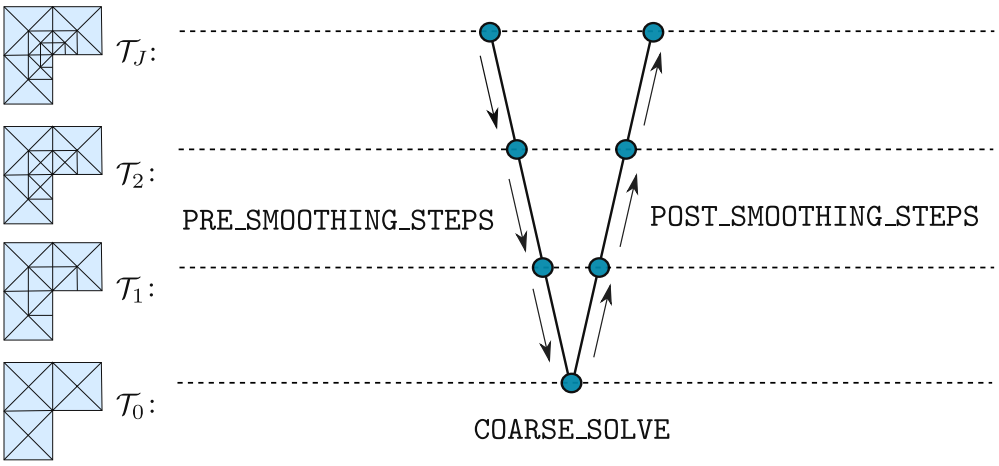


Figure 6: A typical multigrid V-cycle with $J = 4$ levels (Figure 3 of [90]).

pre-smoothing steps and only one post-smoothing step; 2) the smoother is a simple Jacobi smoother in the lowest order case and block Jacobi in general; 3) the solver is p-robust, that is, the contraction property of the solver is independent of the polynomial order; 4) the solver has a built-in a posteriori estimator of the algebraic error. This last property will be the most crucial for us, as we will design an adaptive algorithm based on this estimator.

v.2 Iterative refinement and mixed precision

The vast majority of numerical algorithms are carried out in finite precision, more specifically using the floating point arithmetic. Mixed precision methods involve using different precision levels (such as single, double, or extended precision) at various stages of a computation, see, e.g., [71, 124] and references therein. This approach allows for the balancing of computational speed and memory usage, as lower precision arithmetic is typically faster and less memory-intensive. One Mixed precision can in particular be applied in iterative refinement, originally introduced in [124].

Iterative refinement is a correction method consisting of an inner solver and a residual correction. In particular, an equation for the error is solved in lower precision and the residual calculation is computed in higher precision, see Algorithm 1. The basic tenant of iterative refinement is that solving for the correction generally requires less precision since the entries of the residual are on a smaller order compared to the right hand side b. Since the residual is small, and if A is not too ill-conditioned, the correction y should also be small. Furthermore, if y is small, then the rounding errors in computing y should be negligible when compared to the rounding errors in x. Iterative refinement has been studied in the context of direct solvers [124, 38], Krylov subspace methods [27, 26], and more recently in [88, 115] for multigrid methods.

Algorithm 1: Iterative refinement in mixed precision

Input: initial guess $x^{(0)}$, A, b
$i := 0$
while $\|r^{(i)}\| > \text{tol}$ **do**
  $r^{(i)} := b - Ax^{(i)}$ ∗ Compute residual in high precision
  $y^{(i)} := \text{InnerSolve}(A, r^{(i)})$ ∗ Compute correction in low precision
  $x^{(i+1)} := x^{(i)} + y^{(i)}$ ∗ Update in high precision
  $i := i + 1$

vi A posteriori error estimation and adaptive mesh refinement

The final but crucial ingredient for designing adaptive algorithms are a posteriori error estimates. A posteriori error estimation for PDEs is a well established subject, see for example the books of Verfürth [120], Ainsworth and Oden [1], Repin [107], and the references therein. A posteriori errors estimators can be utilized to 1) certify the error; 2) drive adaptive mesh refinement strategies; and 3) drive adaptivity in a larger sense, such as providing stopping criteria for iterative solvers. In general, important properties of such estimators are their *reliability* (upper bound for the error) and *efficiency* (lower bound for the error), where the constants in the upper bounds are ideally explicit, independent of the PDE data and finite-dimensional approximation parameters. More specifically in the context of adaptive stopping criteria, it is especially attractive to have a *constant-free* upper bound. As for the error lower bound, the goal is to obtain a generic constant ideally independent of the model and/or discretization parameters, which we refer to as *robustness*.

vi.1 Primal–dual gap error estimators

Thanks to the minimization structure in (12), we can consider a special type of a posteriori error estimators known as primal–dual gap estimators [109, 108, 107, 13, 129]. In particular, these estimators do indeed provide a constant-free upper bound on the difference of energies. The following uses standard duality arguments for convex functions, following [17, 107, 129]. Note that in this section we specifically consider Sobolev space $V = H^1_0(\Omega)$. We begin by defining the energy functional $\mathcal{J} : H^1_0(\Omega) \to \mathbb{R}$ as the right hand side of (12), i.e.

$$\mathcal{J}(v) := \int_\Omega \phi(|\nabla v|)\,\mathrm{d}\boldsymbol{x} - \int_\Omega f v\,\mathrm{d}\boldsymbol{x}. \tag{11}$$

We consider the associated minimization problem:

$$u = \operatorname*{arg\,min}_{v \in H^1_0(\Omega)} \mathcal{J}(v). \tag{12}$$

Then, we define the associated dual functional $\mathcal{J}^* : \boldsymbol{H}(\mathrm{div}, \Omega) :\to \mathbb{R}$ by

$$\mathcal{J}^*(\boldsymbol{\varsigma}) := -\int_\Omega \phi^*(|\boldsymbol{\varsigma}|)\,\mathrm{d}\boldsymbol{x}, \tag{13}$$

where the star represents the Fenchel conjugate

$$\phi^*(r) := \max_s \{sr - \phi(s)\}. \tag{14}$$

Then, for any function $v \in H^1_0(\Omega)$ and any vector-valued field $\boldsymbol{\varsigma}$ satisfying $\boldsymbol{\varsigma} \in \boldsymbol{H}(\mathrm{div}, \Omega)$ with $\nabla{\cdot}\boldsymbol{\varsigma} = f$, we have the following result (cf. Corollary 1.5.6)

$$0 \leq \mathcal{J}(v) - \mathcal{J}(u) \leq \mathcal{J}(v) - \mathcal{J}^*(\boldsymbol{\varsigma}). \tag{15}$$

In particular, we use the bound (15) to define a computable error estimator with constant-free upper bound, (cf. Lemma 1.7.1)

$$0 \leq \underbrace{\mathcal{J}(u_\ell^{j,k}) - \mathcal{J}(u)}_{\text{total error } (e_{\mathrm{tot}}^{\ell,j,k})^2} \leq \underbrace{\mathcal{J}(u_\ell^{j,k}) - \mathcal{J}^*(\boldsymbol{t}_\ell^{j,k})}_{\text{total est. } (\eta_{\mathrm{tot}}^{\ell,j,k})^2}, \tag{16}$$

where the flux $\boldsymbol{t}_\ell^{j,k} \in \boldsymbol{H}(\mathrm{div}, \Omega)$ with $\nabla{\cdot}\boldsymbol{t}_\ell^{j,k} = f$ is computed at each step ℓ, j, k and is an approximate solution of the dual problem achieved by means of some local (patchwise) problems. The construction of this object will be the discussion of §vi.3.

vi.2 Other ways of measuring and estimating the error

In addition to measuring the errror via the difference of the energies as in (15), a second type of error measure we will consider will be that of the energy norm. For $v \in H^1_0(\Omega)$, we namely consider

$$|||v||| := \alpha^{1/2} \|\nabla v\|, \tag{17}$$

for some real number $\alpha > 0$. One can also consider the dual norm of the residual. Let the residual $\mathcal{R}(v) \in H^{-1}(\Omega)$ be given by

$$\langle \mathcal{R}(v), w \rangle := (f, w) - (a(|\nabla v|)\nabla v, \nabla w), \quad w \in H^1_0(\Omega). \tag{18}$$

Then the dual norm of the residual is given by

$$|||\mathcal{R}(v)|||_{-1} := \sup_{\varphi \in H^1_0(\Omega),\, |||\varphi|||=1} \langle \mathcal{R}(v), \varphi \rangle. \tag{19}$$

These three error measure turn out to be equivalent in the linear case $a(s) =$ constant (cf. §1.4.4)

$$\mathcal{J}(v) - \mathcal{J}(u) = \tfrac{1}{2}|||v - u|||^2 = \tfrac{1}{2}|||\mathcal{R}(v)|||^2_{-1}.$$

This equivalence extends to the nonlinear case but the equivalence constant depends on the nonlinear function a. The corresponding error estimator for these two error measures is (cf. § 1.7.2–1.7.3)

$$0 < \tfrac{1}{2}|||u_\ell^{j,k} - u|||^2 \leq \tfrac{1}{2}|||\mathcal{R}(u_\ell^{j,k})|||^2_{-1} \leq \tfrac{1}{2} \underbrace{\alpha^{-1}\|\boldsymbol{A}(\nabla u_\ell^{j,k}) + \boldsymbol{t}_\ell^{j,k}\|^2}_{\text{total est. } (\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k})^2}.$$

vi.3 Equilibrated flux

In the previous section, we observed that the computability of the estimators hinged on being able to produce an object $\boldsymbol{t}_\ell^{j,k}$ in $\boldsymbol{H}(\mathrm{div}, \Omega)$ with a specified divergence. This is achieved by solving linear, local, and mutually independent problems on patches of mesh elements. This resulting object is referred to as the equilibrated flux and is based on principles first established in the works of Ladevèze and Leguillon [84], Destuynder and Métivet [41], Braess

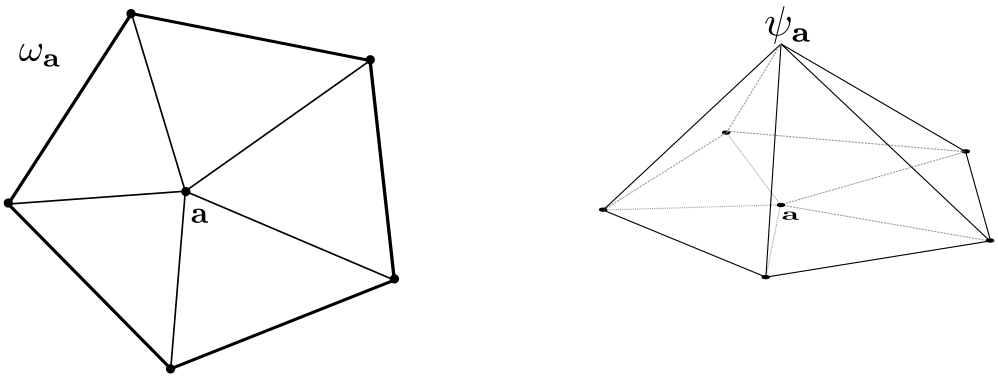


Figure 7: A nodal patch and its associated hat function.

and Schöberl [21], and Ern and Vohralík [53]. One major advantage of this strategy is the so-called p-robustness, i.e., the resulting estimator is uniformly efficient for arbitrary polynomial order p. For the linearized problem (10) this construction boils down to solving, on each nodal patch $\omega_{\mathbf{a}}$ (see Figure 7) corresponding to the node $\mathbf{a}$ of the mesh $\mathcal{T}_\ell$ a mixed problem of the form: find $(\boldsymbol{t}_{\mathbf{a}}^{j,k}, q_{\mathbf{a}}) \in \boldsymbol{V}_{\mathbf{a}} \times Q_{\mathbf{a}}$ such that

$$(\boldsymbol{t}_{\mathbf{a}}^{j,k}, \boldsymbol{v}_\ell)_{\omega_{\mathbf{a}}} - (q_{\mathbf{a}}, \nabla\cdot\boldsymbol{v}_\ell)_{\omega_{\mathbf{a}}} = -(\psi_{\mathbf{a}}(\boldsymbol{A}_\ell^{j,k-1}\nabla u_\ell^{j,k} - \boldsymbol{b}_\ell^{j,k}), \boldsymbol{v}_\ell)_{\omega_{\mathbf{a}}}, \tag{20a}$$

$$(\nabla\cdot\boldsymbol{t}_{\mathbf{a}}^{j,k}, r_\ell)_{\omega_{\mathbf{a}}} = (f\psi_{\mathbf{a}} - (\boldsymbol{A}_\ell^{j,k-1}\nabla u_\ell^{j,k} - \boldsymbol{b}_\ell^{j,k})\cdot\nabla\psi_{\mathbf{a}}, r_\ell)_{\omega_{\mathbf{a}}} \tag{20b}$$

for all pairs $(\boldsymbol{v}_\ell, r_\ell) \in \boldsymbol{V}_{\mathbf{a}} \times Q_{\mathbf{a}}$. See §1.6.1 for full details. We give details of our implementation in the Julia programming language in Chapter 4 in the simpler but very much relevant context of the Poisson equation.

vi.4 Adaptive mesh refinement

One common use of a posteriori error estimation is driving adaptive mesh refinement algorithms. Adaptive mesh refinement algorithms are well studied, see e.g., [119, 6, 97, 28] and references therein. To preserve mesh continuity and mesh shape regularity during refinement, we use the newest-vertex bisection (NVB) algorithm, see Figure 8 for an illustration. The NVB algorithm is applied in tandem with Dörfler marking [46] (cf. Algorithm 3 below) for the selection of elements to refine. I have implemented the newest vertex bisection algorithm in the `Gridap.jl` [118, 7] open source finite element package via the pull request #901, https://github.com/gridap/Gridap.jl/pull/901. It can be applied via a single line call to a given mesh, as demonstrated in the following example.

```
using Gridap
# Make a uniform simplicial mesh on the unit square with 10 x 10 x 2 elements
Ω = CartesianDiscreteModel((0,1,0,1), (10,10)) |> simplexify
using Gridap.Adaptivity
# Refine (at least) the cells 5, 13, and 23 using newest vertex bisection
Ω_ref = refine(Ω, refinement_method = "nvb", cells_to_refine = [5, 14, 23])
```

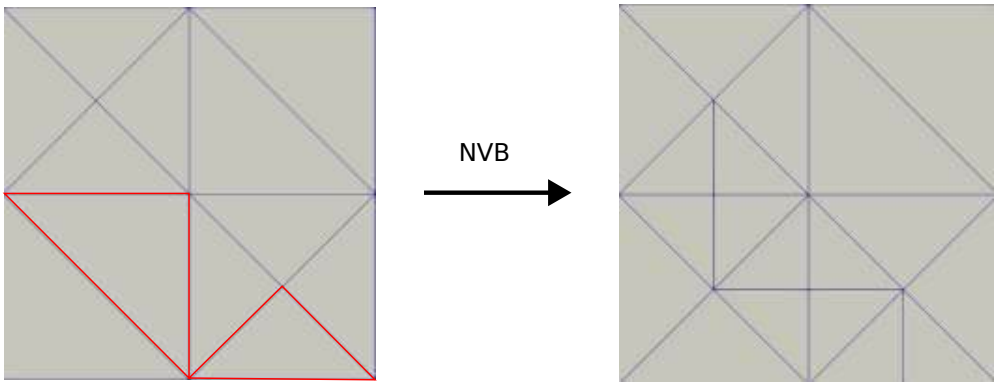


Figure 8: The red elements are marked for refinement. The newest vertex bisection algorithm (NVB) ensures the mesh continuity by bisecting additional elements to those marked, and always choosing to bisect by adding a new node opposite to the longest edge.

vii Contents and contributions of the thesis

This manuscript consists of five chapters excluding this introductory one. We now give their summary.

vii.1 Chapter 1: Adaptive regularization, discretization, and linearization for nonsmooth problems based on primal-dual gap estimators

This chapter corresponds to the article [58] in Computer Methods in Applied Mechanics in Engineering. It is a joint work with François Févotte and Martin Vohralík.

In this chapter, we are interested in two main questions: 1) how can can we we obtain a good approximation to the solution of a nonsmooth PDE by applying Newton's method to a sequence of auxiliary regularized and discretized problems and 2) can we do so while bounding the error in difference of energies as discussed in §iii.2. To this end, we introduce a three-level adaptive algorithm controlling regularization (with index j), discretization (with index ℓ), and linearization (with index k) in the numerical resolution of nonsmooth PDEs. The regularization is applied at the PDE level by modifying the nonlinear nonsmooth function defining the PDE. Our adaptive algorithm is driven by component estimators that isolate the errors coming from regularization, discretization, and linearization. These component estimators provide a constant-free upper bound decomposition of the so-called primal–dual gap estimator of the form (16).

The main results of this chapter are as follows: 1) we introduce a global regularization of a nonsmooth function in the spirit of Figure 3, thus obtaining a sequence of smooth PDEs; 2) we derive a decomposition of the standard primal–dual gap estimator (16) into components, (cf. Theorem 1.7.3)

$$(\eta_{\mathrm{tot}}^{\ell,j,k})^2 \leq \underbrace{\left| \int_\Omega \phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{d}_\ell^{j,k}|) + \nabla u_\ell^{j,k} \cdot \boldsymbol{t}_\ell^{j,k} \,\mathrm{d}\boldsymbol{x} \right|}_{\text{discretization est. } (\eta_{\mathrm{dis}}^{\ell,j,k})^2}$$

$$+\underbrace{\left|\int_\Omega \phi^*(|\boldsymbol{d}_\ell^{j,k}+\boldsymbol{r}_\ell^{j,k}|)-\phi^*(|\boldsymbol{d}_\ell^{j,k}|)\,\mathrm{d}\boldsymbol{x}\right|}_{\text{regularization est. } (\eta_{\mathrm{reg}}^{\ell,j,k})^2}$$
$$+\underbrace{\left|\int_\Omega \phi^*(|\boldsymbol{t}_\ell^{j,k}|)-\phi^*(|\boldsymbol{d}_\ell^{j,k}+\boldsymbol{r}_\ell^{j,k}|)\,\mathrm{d}\boldsymbol{x}\right|}_{\text{linearization est. } (\eta_{\mathrm{lin}}^{\ell,j,k})^2};$$

3) we prove that these component error estimators converge to zero in their respective limits. More precisely, we have the following

Lemma 1.7.4: $\lim_{j,k\to\infty} \eta_{\mathrm{reg}}^{\ell,j,k} = 0,$

Lemma 1.7.5: $\lim_{k\to\infty} \eta_{\mathrm{lin}}^{\ell,j,k} = 0,$

Lemma 1.7.6: $\lim_{j,k,\ell\to\infty} \eta_{\mathrm{dis}}^{\ell,j,k} = 0;$

4) we design an adaptive algorithm (Algorithm 2) balancing the error components: see Figure 9 for an example of its execution. More precisely, we plot the values of the component estimators for each combination of indices the ℓ, j, k. Some important remarks about the behavior of the estimators are:

1. The linearization estimator converges rapidly for a given mesh ℓ and regularization parameter j
2. The regularization estimator decreases in j and is kept strictly below the discretizaiton estimator for almost all the iterations
3. The discretization estimator decreases in ℓ

5) we consider the convergence of the total estimator with respect to total degrees of freedom (DOFs). Specifically, we apply the algorithm for both a known smooth solution as well as an unknown solution on an L-shaped domain. In particular, in the smooth case we recover the optimal rate of convergence with respect to DOFs for polynomial degree $p = 1$ using uniform refinement, see Figure 10. For the unknown solution on the L-shaped domain, we observe the optimal rate of convergence for the total estimator using adaptive mesh refinement for polynomial order up to 3. Indeed, in Figure 11 we compare the results of using uniform and adaptive refinement for Algorithm 2 (in both cases the adaptive regularization and linearization are performed as usual), and in particular the sub-optimality for uniform refinement due to the re-entrant corner.

vii.2 Chapter 2: Robust energy a posteriori estimates for nonlinear elliptic problems

This chapter corresponds to article [68]. It is a joint work with André Harnist, Koondanibha Mitra, and Martin Vohralík. This article has been submitted for publication and is currently under review.

In this chapter, we conduct a rigorous mathematical study of the robustness of the primal–dual gap estimator (16) with respect to the strength of the nonlinearity of the function a. We do not consider regularization in this chapter, hence the index j does not appear. In

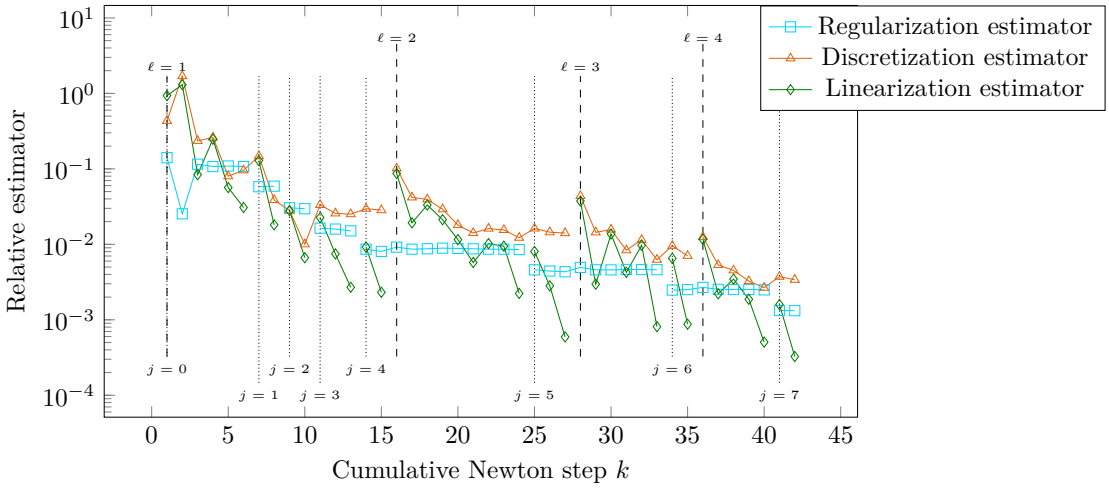


Figure 9: Evolution of the component estimators during the execution of the adaptive Algorithm 2. The estimators are defined in §1.7.1. Chapter 1, Figure 1.8 and https://doi.org/10.1016/j.cma.2023.116558.

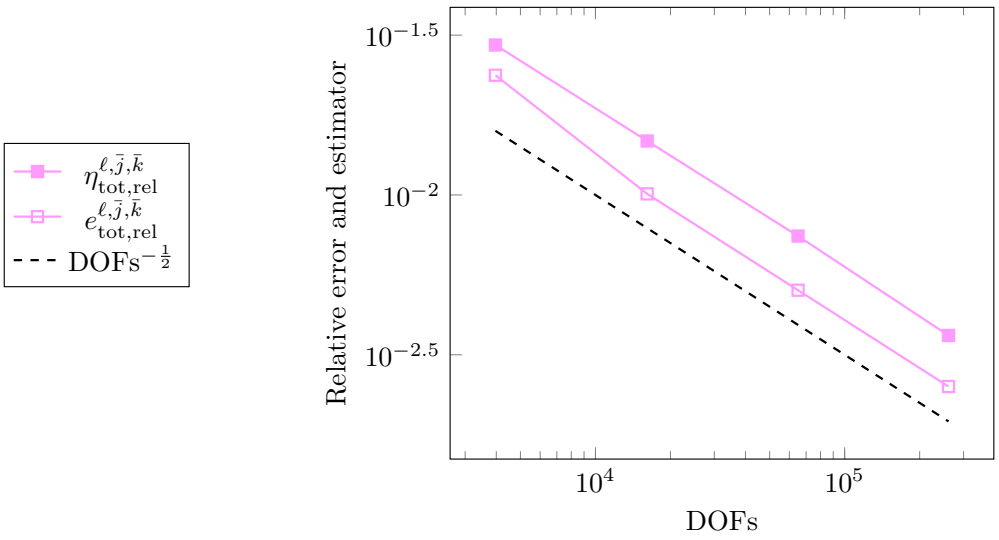


Figure 10: Optimal rate of convergence with respect to DOFs for the total estimator and error (16). See Figure 1.9 and https://doi.org/10.1016/j.cma.2023.116558.

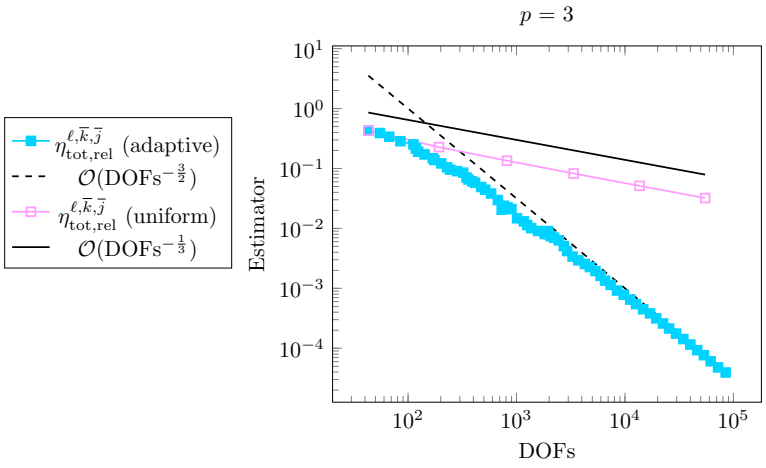


Figure 11: Comparing uniform and adaptive mesh refinement for an unknown solution on an L-shaped domain for the total error and estimator (16). The uniform refinement results in sub-optimal convergence with respect to DOFs, while the adaptive refinement recovers the optimal rate, including iterative linearization and iterative regularization. See Figure 1.13 and https://doi.org/10.1016/j.cma.2023.116558

particular, we seek to show that the primal-dual gap estimator is not only an upper bound but also, up to a generic constant, a lower bound of the difference of energies. As a second step, we consider an augmented version of the difference of energies which takes into account the iterative linearization, e.g., the Picard/fixed point iteration (a), the Zarantonello iteration (b), or the Newton iteration (c). The linearized energy functional $\mathcal{J}_\ell^{k-1} : H_0^1(\Omega) \to \mathbb{R}$ defined for all $v \in H_0^1(\Omega)$ and associated to a linearization of the form (10) is given by

$$\mathcal{J}_\ell^{k-1}(v) := \frac{1}{2}\left\|(\boldsymbol{A}_\ell^{k-1})^{\frac{1}{2}}\nabla v\right\|^2 - (f, v) - (\boldsymbol{b}_\ell^{k-1}, \nabla v). \tag{21}$$

The augmented energy difference then reads as

$$\mathcal{E}_\ell^k := \frac{1}{2}\left(\underbrace{(2(\mathcal{J}(u_\ell^k) - \mathcal{J}(u)))^{\frac{1}{2}}}_{\text{nonlinear}} + \lambda_\ell^k \underbrace{\left(2\big(\mathcal{J}_\ell^{k-1}(u_\ell^k) - \mathcal{J}_\ell^{k-1}(u_{\langle\ell\rangle}^k)\big)\right)^{\frac{1}{2}}}_{\text{linearized}}\right) \tag{22}$$

where λ_ℓ^k (cf. (2.41)) is defined to "balance" the nonlinear and linear components of the energy difference, and $u_{\langle\ell\rangle}^k$ is the minimum of the energy functional (21) over the infinite-dimensional space $H_0^1(\Omega)$. We have an equivalent augmented estimator:

$$\eta_\ell^k := \frac{1}{2}\left(\underbrace{(2\mathcal{J}(u_\ell^k) - \mathcal{J}^*(\boldsymbol{\sigma}_\ell^k))^{\frac{1}{2}}}_{\text{nonlinear}} + \lambda_\ell^k \underbrace{\left(2\mathcal{J}^{k-1}(u_\ell^k) - \mathcal{J}^{*,k-1}(\boldsymbol{\sigma}_\ell^k)\right)^{\frac{1}{2}}}_{\text{linearized}}\right), \tag{23}$$

where $\mathcal{J}^*$ is the dual energy of (13) and $\boldsymbol{\sigma}_\ell^k$ is an equilibrated flux, cf. (2.25). For this augmented energy, we are able to prove robustness for the Zarantonello iteration, and a reduced local dependence for the other linearization schemes that we study. More precisely, we have (cf. Theorem 2.5.5)

$$\mathcal{E}_\ell^k \leq \eta_\ell^k + \text{quadrature type and oscillation terms}$$

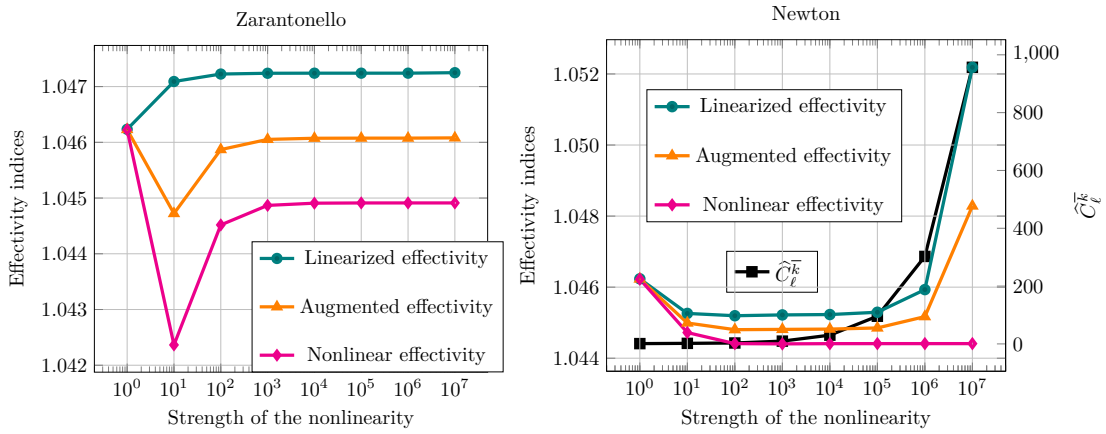


Figure 12: The effectivity indices are the ratios of the respective estimators in (23) divided by the errors in (22). We observe robustness of the estimators with respect to the strength of the nonlinearity for the Zarantonello linearization (left) and lack thereof for the Newton linearization (right) where the computable constant $\widehat{C}_\ell^k$ blows up for large values as discussed in Chapter 2, Figure 2.3 and `https://hal.science/hal-04033438`.

$$\eta_\ell^k \lesssim \widehat{C}_\ell^k \mathcal{E}_\ell^k + \text{quadrature type and oscillation terms},$$

where the constant $\widehat{C}_\ell^k = 1$ for the Zarantonello linearization. This in particular means that this estimator is *robust with respect to the strength of the nonlineaity* for the Zarantonello linearization. As a consequence, the effectivty index for the augmented estimator/error is independent of the strength of the nonlinearity expressed as the ratio Lipshitz continuity / monotonicity. This is demonstrated numerically in Figure 12 (left), where the effectivity index is stable over seven orders of magnitude for the strength of the nonlinearity. In fact, we observe that the components of the augmented estimator and error (nonlinear and linearized) also exhibit robustness in this case.

We derive a similar result (cf. Theorem 2.4.4) involving only the nonlinear component of the estimator with a constant $\widehat{C}_\ell^k$ in the lower bound. However, in this case, even for the Zarantonello the constant is not fully independent of the strength of the nonlinearity. However, the constant $\widehat{C}_\ell^k$ is computable in the augmented case. In particular in Figure 12 (right), we compute the effectivity indices for the three different estimator/error pairs on the left axis, as well as the constant $\widehat{C}_\ell^k$ on the right axis. In particular, we observe that the constant $\widehat{C}_\ell^k$ explodes in this case with respect to the strength of the nonlinearity. The linearized effectivtiy (and hence the augmented effectivity) also increase with the constant $\widehat{C}_\ell^k$, but the increase is much less dramatic.

These results are to the best of our knowledge the first rigorous study for the robustness of primal–dual gap estimators with respect to strength of the nonlinearity.

vii.3 Chapter 3: Adaptive regularization for the Richards equation

This chapter corresponds to the article [58], a joint work with François Févotte and Martin Vohralík. This article has been submitted for publication.

This chapters applies some of the framework of Chapter 1 to the setting of the Richards equation, (1). We propose regularized versions of the nonsmooth and degenerate functions

appearing in the Richards equation for the two most common models (Brook–Corey and van Genuchten–Mualem see Figures 3.1 and 3.8). We then consider an implicit time discretization of the Richards equations resulting in a sequence of nonlinear PDEs on each timestep. Unfortunately, these PDEs do not correspond to an energy minimization problem, so we cannot use primal–dual gap estimators. However, we design simpler component estimators based on the dual norm of the residual as studied in [94], namely, (cf. (3.27)) for a timestep n, regularization step j, and linearization step k, we define

$$\begin{aligned}
\eta_{\mathrm{dis}}^{n,j,k} &:= \|\boldsymbol{F}_{\epsilon^j}^k + \boldsymbol{\sigma}_\ell^{j,k}\| && \text{(discretization)},\\
\eta_{\mathrm{lin}}^{n,j,k} &:= \|\boldsymbol{F}_{\epsilon^j}(p_\ell^{j,k}) - \boldsymbol{F}_{\epsilon^j}^k\| && \text{(linearization)},\\
\eta_{\mathrm{reg}}^{n,j,k} &:= \|\boldsymbol{F}(p_\ell^{j,k}) - \boldsymbol{F}_{\epsilon^j}(p_\ell^{j,k})\| && \text{(regularization)}.
\end{aligned}$$

We thus obtain a decomposition into error components that steers an adaptive algorithm, cf. Algorithm 4. See Figure 3.12 for the evolution of these estimators during the first timestep in the numerical test of §3.6.3.

We study the performance of our adaptive algorithm applied to several test cases in the literature that are known to cause difficulties for Newton's method. In all the cases considered, our adaptive algorithm is able to finish the simulation. We compare the numerical solution produced by the adaptive algorithm with a solution obtained using an ad-hoc linearization for Richards equation, the so-called modified Picard [31] method. Our proposed regularized Newton method outperforms the modified Picard method in terms of total number of iterations, and produces a comparable approximate solution. As an example, in Figure 13, we plot the cumulative and stepwise number of iterations corresponding to the numerical test in §3.6.3. In particular, Newton's method without regularization takes over 300 iterations on the first timestep so we stop it there (no convergence). The modified Picard method is able to reach the end of the simulation, but takes substantially more iterations per timestep than our proposed adaptive regularization Newton method. To show that the approximate solution is of similar quality to the unregularized one resulting from modified Picard, we plot a side by side comparison in Figure 14. In particular, here we are plotting the saturation $S(p_h)$ for our numerically computed solution p_h. Finally, we show the evolution of the component estimators over the course of three timesteps in Figure 15. We see a similar behavior to that observed in Chapter 1, where the regularization error decreases in j and ends up below the discretization error (here at the end of a timestep as opposed to before switch meshes in the Chapter 1).

Our proposed approach is novel compared to other proposed linearization methods for Richards equation. Firstly, we do not change the underlying Newton solver, which is advantageous when the solver is being treated as a black box. Another advantage is that Newton's method is applied on a sequence of smooth and nondegenerate problems, therefore good convergence properties are recovered. We can also easily keep the original physical unknowns, which is often useful in engineering practice. Finally, a failsafe mechanism allows the solver to recover in the case of failure of convergence.

vii.4 Chapter 4: Notes on implementation of equilibrated fluxes

This chapter describes the implementation of the equilibrated flux, which is realized in the following GitHub repository `https://github.com/aerappa/EquilibratedFlux.jl`. *This repository received contributions from François Févotte.*

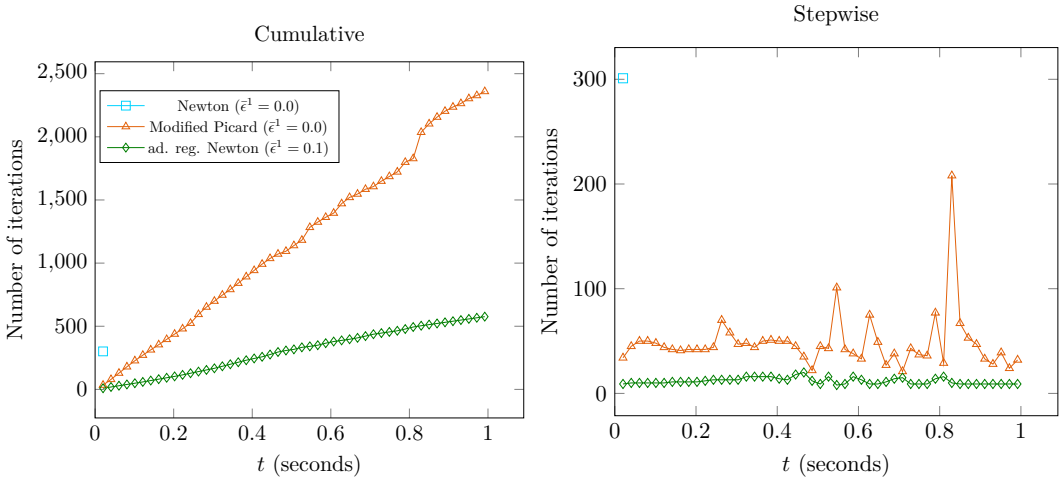


Figure 13: Performance comparison between the three methods considered in Chapter 3, Figure 3.11 and https://hal.science/hal-04266827.

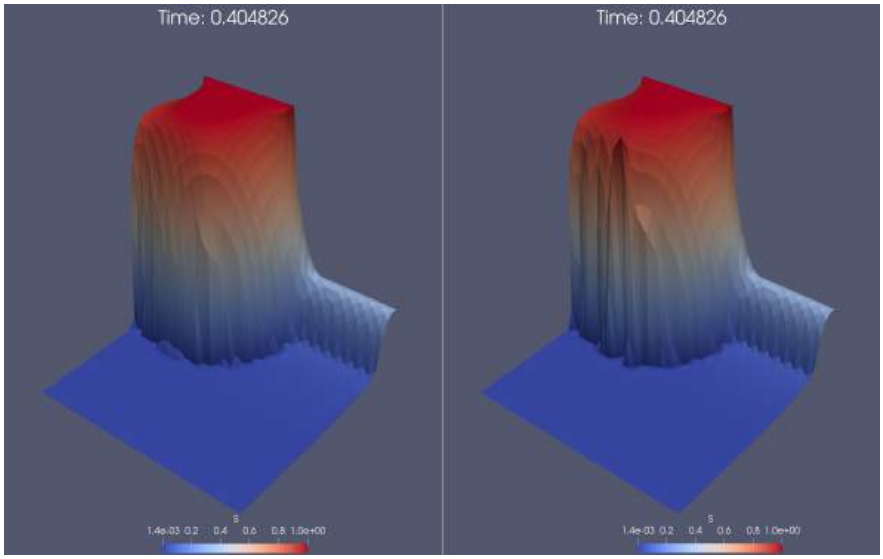


Figure 14: Comparison of the resulting saturation profiles for the regularized and unregularized algorthims solving Richards equations in Chapter 3, Figure 3.13 and https://hal.science/hal-04266827.

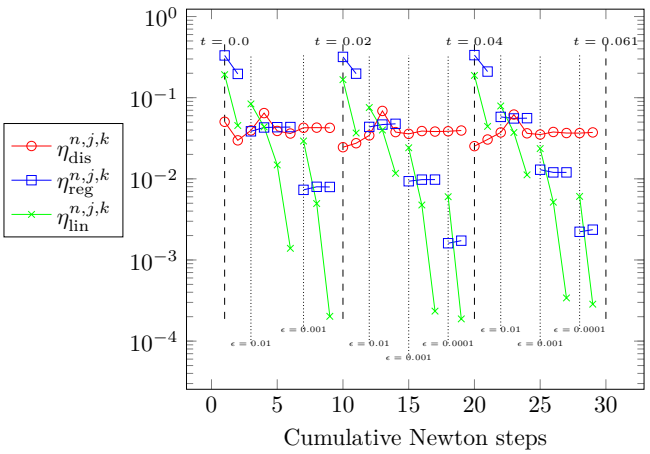


Figure 15: The evolution of the component estimators during the first timestep for the adaptive algorithm solving the Richards equation in Chapter 3, Figure 3.12 and https://hal.science/hal-04266827.

Underlying much of the aforementioned developments is the equilibrated flux of §vi.3. Thus, having a performant implementation is crucial to the realization of the subsequent adaptive algorithms. In Chapter 4, we present an implementation of the equilibrated flux in the Julia programming language with the help of the `Gridap.jl` library [118, 7]. As described in §vi.3, see the formula (20), the flux is realized by solving mutually independent problems on nodal patches of a given mesh. This independence naturally lends itself to parallelism. However, there are several considerations that go into realizing a parallel implementation. The first question one must ask is whether to attempt shared memory or distributed memory parallelism. Here, we opt for shared memory parallelism using the threading model in Julia. This leads to the next big question of how to handle memory and reduce allocations as much as possible. This is especially crucial for shared memory programs where multiple threads are accessing the same data structures in main memory. The novelty in this chapter is the design of an implementation of the equilibrated flux which completely avoids allocations when solving the patch problems by performing an initial loop on cells. Furthermore, the loop on cells is a common paradigm in finite element codes, and many of the details can be handed off to optimized routines in the library. The work on each cells is also parallel, resulting in the possibility for more parallelism. The (parallel) performance of our implementation is evaluated for a model problem and results of scaling are presented.

vii.5 Chapter 5: Adaptive safeguarded iterative refinement

Up until this point we have only considered the direct resolution of the sparse linear systems of equations arising from the solution strategy of §iv.1–§iv.3. We now would like to explore in more detail an iterative method for solving sparse linear systems. We will in particular be interested in improving performance (speedup) by working with mixed precision floating point arithmetic as described in §v.2. There are two main ways that using mixed precision arithmetic can accelerate the performance of a numerical method.

1. **Processor intrinsics:** Modern processors have Single Instruction Multiple Data (SIMD) instructions (like SSE, AVX). These are optimized for performing multiple

operations in parallel, making them ideal for vectorized computations. Lower precision formats allow more data elements to fit into a single SIMD vector register. This means that each SIMD instruction can process more elements at once, compared to higher precision formats.

2. **Reducing the overall memory footprint:** Smaller data size means more data can fit into the same memory space. This leads to better utilization of caches and reduced memory bandwidth requirements.

We will focus here on the latter, since it is especially pertinent for iterative methods for sparse linear systems. This is due to the *memory-bound* nature of sparse linear algebra. Memory-bound operations are defined by their low arithmetic intensity: the bottleneck is the time required to move data through the memory hierarchy rather than to perform actual floating point operations (FLOPs) on the data. In this chapter, we specifically develop a performance model for sparse matrix-vector multiplication (SpMV), a basic building block of nearly all iterative methods for sparse linear systems and which is memory-bound. We consider the possible speedup in relation to a reduced footprint achieved by storing single precision floating point numbers rather than double precision. Our scheme, which we describe below in more detail, consists primarily in SpMVs, and the speedup due to storing and performing certain operations in single precision is demonstrated in Figure 17.

While mixed precision can have great benefits in terms of savings for memory bound kernels, the inherent challenge of using lower precision is its reduced robustness with respect to ill-conditioning. More precisely, lower precision formats, with fewer bits for number representation, result in greater rounding errors (especially in operations like subtraction), and a heightened sensitivity to perturbations (reduced numerical stability). Thus, we must be very careful in designing mixed precision algorithms. As discussed in §v.2, iterative refinement is one strategy in the context of linear solvers to perform some operations in reduced precision while maintaining numerical stability.

In this chapter, we will consider a special case of iterative refinement with geometric multigrid (an iterative method consisting primarily in SpMVs) as the inner solver. More precisely, we consider the lowest order version of the geometric multigrid solver developed in [89, 91] as the inner solver for the iterative refinement. This version of geometric multigrid has the attractive property of providing an error estimator at each iteration. We thus use this estimator to design a more robust version of iterative refinement. In particular, we present an example where a classical iterative refinement diverges and where our adaptive algorithm is able to detect divergence and switch to fully double precision, see Figure 16.

The main results are 1) the introduction of an iterative refinement variant where the stopping criterion is based on a rigorous error estimator 2) the validation of a performance model of a matrix-vector product based inner solver and 3) the design of an adaptive algorithm that is more robust for highly ill-conditioned problems.

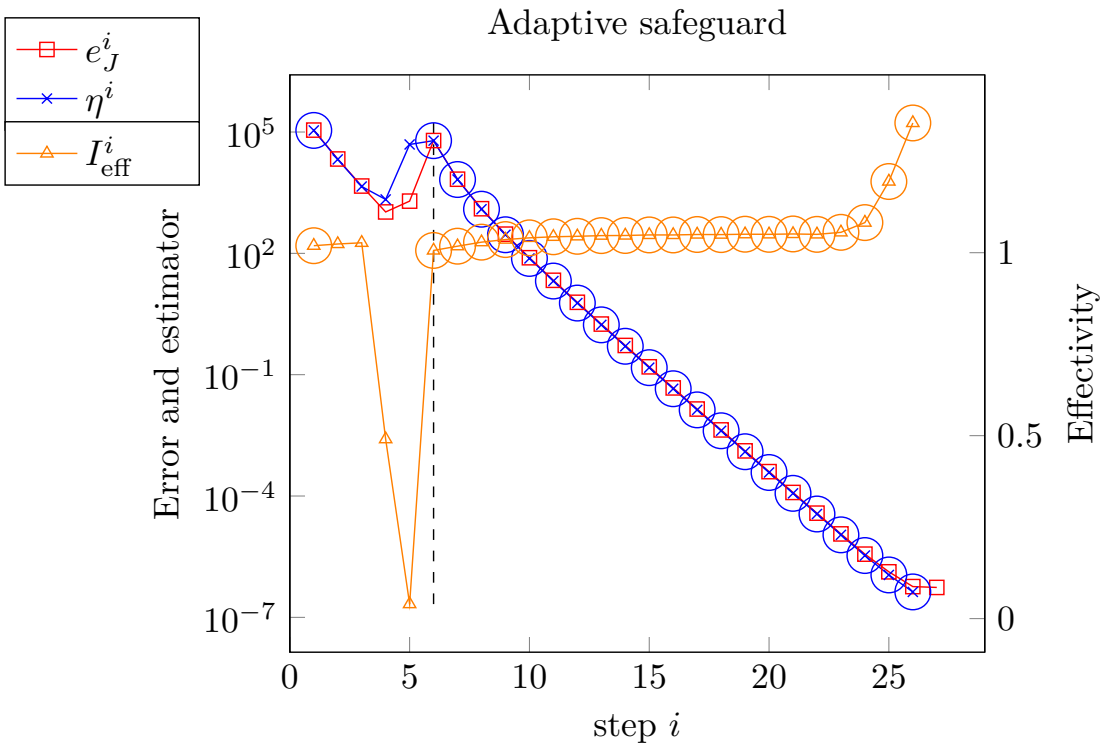


Figure 16: Demonstrating the divergence and subsequent switch to double precision for a badly-conditioned problem. Adaptive safeguarded iterative refinement, Chapter 5, Figure 5.6.

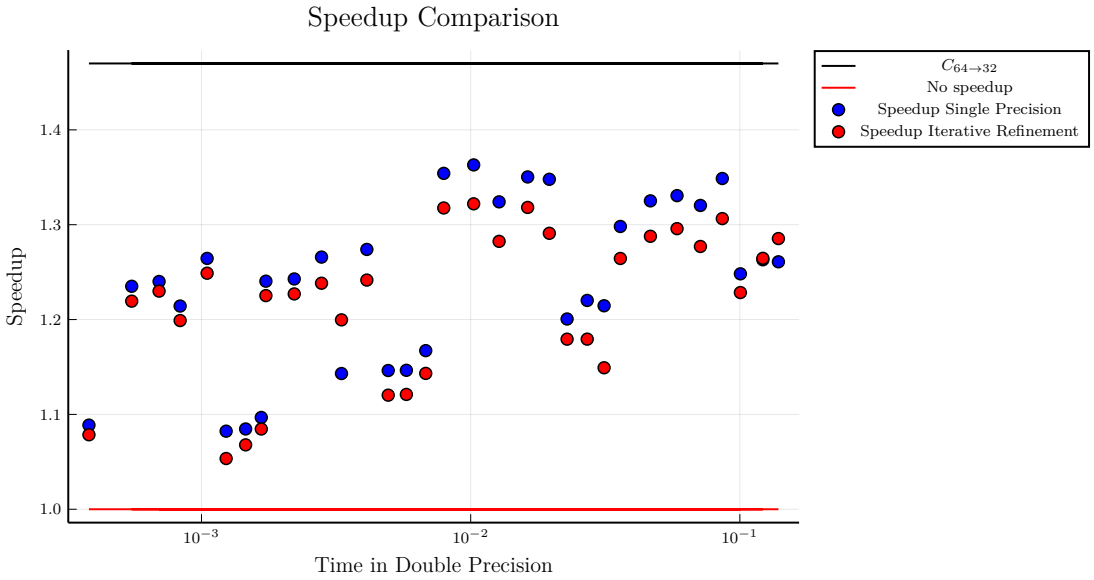


Figure 17: The computed speedup Chapter 5, Figure 5.4.

Chapter 1

Adaptive regularization, discretization, and linearization for nonsmooth problems based on primal-dual gap estimators

This chapter corresponds to the contents of the paper https://doi.org/10.1016/j.cma.2023.116558 in Computational Methods in Applied Mechanics and Engineering.

1.1 Introduction

Given a Hilbert space V, consider the abstract minimization problem

$$u := \operatorname*{arg\,min}_{v \in V} \mathcal{J}(v)$$

where $\mathcal{J}$ is a convex functional. When $\mathcal{J}$ satisfies certain regularity conditions, one can form the associated Euler–Langrange conditions which are expressed as a nonlinear elliptic partial differential equation (PDE). We are particularly interested in cases where it is difficult to iteratively solve the resulting nonlinear PDE by the standard Newton method, cf. [80, 42], due to non-smoothness. More specifically, applying the standard Newton method can lead to slow convergence or even failure. The difficulty in many cases is the appearance of kink functions, i.e., continuous functions that are non-differentiable on a finite set. In Figure 1.1, we give three examples in the context of degenerate PDEs. In this work, we, in particular, seek to recover good convergence of Newton's method by adaptively regularizing the nonlinear PDE.

By replacing the original problem by a regularized one, a regularization error appears. For inverse problems [73, 98, 48], regularization is a common strategy and the error due to regularization has been extensively studied. In [98], the authors study the so-called Tikhonov regularization and its associated error. The regularization parameter is chosen adaptively and various criteria are discussed. Regularization is also considered for degenerate PDEs where the operators change type as a function of either space or time [100, 43, 96]. In these cases, a regularized problem is introduced that does not suffer from degeneracy. It is proven in [100] that the regularized solutions converge to the true solution in an approximate sense.

In a similar spirit, for Newton-type methods, regularization (smoothing) Newton methods replace non-differentiable nonlinearities with smooth counterparts, see [122, 103, 102] and the references therein. In this case, the amount of added regularization is proportional to a parameter that is driven to zero as the Newton iterations progress, thereby approaching the original problem.

From a practical point of view, the choice of the regularization parameter should ideally be updated in a dynamic way as the chosen numerical method converges. This leads to the question of how to choose the regularization parameter based on information from a solution iterate. In this work, we adaptively update the regularization parameter based on information from a posteriori error estimators.

A posteriori error estimation for PDEs is a well established subject, see for example the books of Verfürth [120], Ainsworth and Oden [1], Repin [107], and the references therein. A posteriori errors estimators can be utilized to 1) certify the error; 2) drive adaptive refinement strategies; and 3) provide stopping criteria for iterative solvers. In general, important properties of such estimators are their *reliability* (upper bound for the error) and *efficiency* (lower bound for the error), where the constants in the upper bounds are ideally explicit, independent of the PDE data and finite-dimensional approximation parameters. More specifically in the context of adaptive stopping criteria, it is especially attractive to have a *constant-free* upper bound. As for the error lower bound, the goal is to obtain a generic constant ideally independent of the model parameters. In the case of strongly monotone and Lipschitz continuous operators, robustness with respect to the ratio of the Lipschitz constant to the monotonicity constant is of particular interest. We will present numerical evidence of robustness in such a setting, where theoretical developments are presented in [68]. However, unlike in [68], we consider regularization, explicitly estimate the regularization error, and introduce a solver strategy with adaptive regularization when the nonlinearity does not satisfy the hypotheses for Newton's method to converge. Furthermore, in this work we identify component error estimators and show that these estimators converge to zero in their respective limits. This in turn shows that the total error (measured in the same way as in [68]) converges to zero in a triple limit as detailed later.

In the context of energy minimization, it is advantageous to study a certain class of a posteriori estimators, namely the so-called primal-dual gap estimators [109, 108, 107, 13, 129]. These estimators rely on results from convex analysis to bound the "difference of energies" which we make precise in §1.4.1. In particular, these estimators do indeed provide a constant-free upper bound on the difference of energies.

In the recent works of [13, 12] Bartels et. al. employ the primal-dual gap estimator to drive a posteriori mesh refinement for singular solutions. It is also applied directly at the level of the energy minimization so that rough problems, e.g., posed in the space of functions of bounded variation (BV), can be treated without appealing to the Euler–Lagrange conditions. The energy minimization is solved directly via the so-called variable-alternating direction method of multipliers [12]. In this method the primal and dual problems are solved in a globally coupled, iterative manner. We note here that duality refers to the dual optimization problem and should not be confused with the notion of duality in adjoint-based a posteriori error analysis.

In the present context, we use a continuous Galerkin finite element discretization for the primal problem and perform a local equilibrated flux reconstruction to obtain a vector field in $\boldsymbol{H}(\mathrm{div}, \Omega)$ with the divergence prescribed by the load. This is achieved by solving linear, local, and mutually independent problems on patches of mesh elements. This resulting object

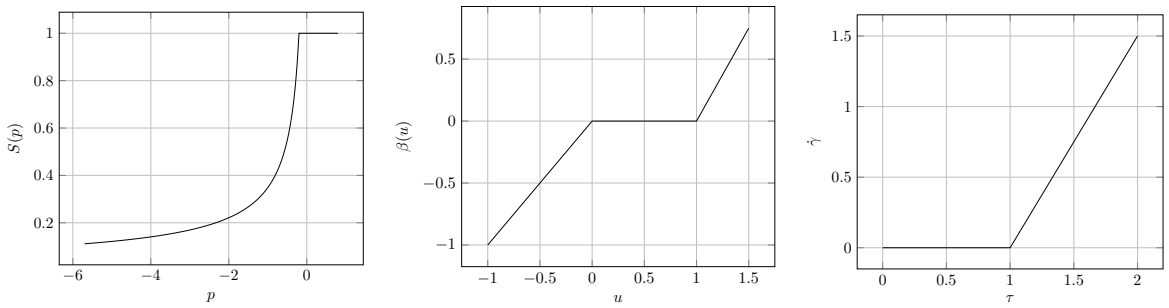


Figure 1.1: Examples of kink-type nonlinear algebraic closures arising in the study of degenerate PDEs. From left to right, we have the saturation function for the Richards equation, namely for the Brooks–Corey model [25], the enthalphy function for the Stefan problem [57], and the shear stress/shear rate relation for Bingham plastics [76].

is referred to as the equilibrated flux and is based on principles first established in Prager and Synge [101] and more recently in the works of Ladevèze and Leguillon [84], Destuynder and Métivet [41], Braess and Schöberl [21], and Ern and Vohralík [53]. One major advantage of this strategy is the so-called p-robustness, i.e., the resulting estimator is uniformly efficient for arbitrary polynomial order.

The main contribution of this work is an adaptive algorithm for solving nonsmooth problems by incorporating regularization into the algorithm. This in particular allows us to apply the standard Newton method to nonsmooth nonlinearities. This adaptive solution strategy resembles the one in [53], where the authors distinguish discretization, linearization, algebraic, and quadrature errors through computable component estimators. However, in [53] no regularization is considered when the nonlinearity is not differentiable, and the question of what to do in this case is not addressed. Here, we construct estimators for the errors due to regularization, discretization, and linearization. These estimators then lead to adaptive stopping criteria to steer an adaptive algorithm. We test our algorithm numerically and recover the optimal convergence rate under uniform refinement for a known smooth solution. We also consider a numerical test for an unknown solution on an L-shaped domain and observe the optimal rate of convergence with respect to total degrees of freedom (DOFs) as well as with respect to the cost for our estimator.

The rest of the paper is organized as follows. In §1.2 we introduce the relevant mathematical details of the problem, as well as our regularization strategy. In §1.3 we define the discrete spaces as well as the particular form of the Newton algorithm. In §1.4 we discuss some common notions of error and their relations to the difference of energies. Next, in §1.5 we introduce the necessary ideas from duality theory to describe the primal-dual gap estimators. In §1.6 we give the details of the flux reconstruction in the current setting. We introduce our decomposition of the upper bound provided by the primal-dual estimator in §1.7. We discuss the efficiency of the estimators in §1.8. We subsequently introduce the adaptive algorithm in §1.9 and we present numerical results in §1.10. Finally, we conclude in §1.11 and discuss future work.

1.2 Continuous problem statement and regularization

In this section we will fix continuous-level notation and then introduce in detail the model problem which we study throughout the rest of the paper.

1.2.1 Notation

For $d = 2, 3$, let $\Omega \subset \mathbb{R}^d$ be a polygon or polyhedron with Lipschitz boundary, $\partial\Omega$. We define the Euclidean norm on $\mathbb{R}^d$ by $|\cdot|$. We introduce the space of Lebesgue square-integrable functions $L^2(\omega)$ with scalar product $(\cdot,\cdot)_\omega$ and norm $\|\cdot\|_\omega$ on $\omega \subseteq \Omega$. We drop the subscript when $\omega = \Omega$. We use the same notation for vector-valued functions. Next, we define, for scalar-valued functions, the standard Sobolev space $H^1(\Omega) = \{v \in L^2 : \partial_{x_i} v \in L^2(\Omega), \forall\, 1 \leq i \leq d\}$ with $H^1_0(\Omega)$ being the subspace of $H^1(\Omega)$ of functions with vanishing trace on $\partial\Omega$. For vector-valued functions we consider the space $\boldsymbol{H}(\mathrm{div}, \Omega) := \{\boldsymbol{v} \in [L^2(\Omega)]^d; \nabla\cdot\boldsymbol{v} \in L^2(\Omega)\}$.

1.2.2 Energy minimization and equivalent formulations

Let $\phi : \mathbb{R} \to \mathbb{R}$ be a given function and $f \in L^2(\Omega)$. Consider the functional $\mathcal{J} : H^1_0(\Omega) \to \mathbb{R}$ given by

$$\mathcal{J}(v) := \int_\Omega \phi(|\nabla v|)\,\mathrm{d}\boldsymbol{x} - \int_\Omega f v\,\mathrm{d}\boldsymbol{x}. \tag{1.1}$$

We will make the following assumptions on the function ϕ.

Assumption 1.2.1 (Assumptions on the energy function)**.** *We assume that the function ϕ is convex and of class $C^1(\mathbb{R})$ with*

$$\phi(0) = \phi'(0) = 0. \tag{1.2}$$

We further assume ϕ satisfies, for real constants $0 < \alpha \leq L$,

$$|\phi'(r) - \phi'(s)| \leq L|r - s| \quad \forall r, s \in \mathbb{R}, \tag{1.3a}$$

$$(\phi'(r) - \phi'(s))(r - s) \geq \alpha(r - s)^2 \quad \forall r, s \in \mathbb{R}. \tag{1.3b}$$

Note that (1.3b) in particular implies $\phi'(r) \geq \alpha r$, $\forall r \geq 0$, so that $\phi' : \mathbb{R}^+ \to \mathbb{R}^+$.

We will be interested in the solution to the minimization problem

$$u := \operatorname*{arg\,min}_{v \in H^1_0(\Omega)} \mathcal{J}(v). \tag{1.4}$$

Due to the convexity of the functional $\mathcal{J}$ following from Assumption 1.2.1 and the fact that $H^1_0(\Omega)$ is complete, the solution exists and is unique, see e.g. [8]. Another way to characterize the solution to problem (1.4) is through its Euler–Langrange equations. To this end, we introduce the nonlinear functions $a : \mathbb{R} \to \mathbb{R}$, $\boldsymbol{A} : \mathbb{R}^d \to \mathbb{R}^d$,

$$a(s) := \frac{\phi'(s)}{s}, \quad \boldsymbol{A}(\boldsymbol{q}) := a(|\boldsymbol{q}|)\boldsymbol{q}. \tag{1.5}$$

A consequence of this definition is the following.

Lemma 1.2.2 (Strongly monotone and Lipschitz continuous operator)**.** *For L and α from Assumption 1.2.1, the operator $\boldsymbol{A}$ given by (1.5) is strongly monotone*

$$\alpha\|\nabla(v - w)\|^2 \leq (\boldsymbol{A}(\nabla v) - \boldsymbol{A}(\nabla w), \nabla(v - w)) \quad \forall v, w \in H^1_0(\Omega). \tag{1.6}$$

It is also Lipschitz continuous

$$\|\boldsymbol{A}(\nabla v) - \boldsymbol{A}(\nabla w)\| \leq L\|\nabla(v - w)\| \quad \forall v, w \in H^1_0(\Omega). \tag{1.7}$$

The proof is standard and is detailed in, e.g., [68, Proposition A.1]. Then the solution to (1.4) also solves the following weak formulation (the Euler–Langrange equations of (1.4)): find $u \in H_0^1(\Omega)$ such that

$$(\boldsymbol{A}(\nabla u), \nabla v) = (f, v) \quad \forall v \in H_0^1(\Omega). \tag{1.8}$$

Consequently, the strong formulation of (1.8) and (1.4) is given by the boundary-value problem

$$-\nabla\cdot\boldsymbol{A}(\nabla u) = f \quad \text{in } \Omega, \tag{1.9a}$$
$$u = 0 \quad \text{on } \partial\Omega. \tag{1.9b}$$

1.2.3 Regularization

According to Assumption 1.2.1 it is possible that the function ϕ belongs to the class $C^1(\mathbb{R})$ but not $C^2(\mathbb{R})$. In particular, this means that the nonlinear functions a and subsequently $\boldsymbol{A}$ defined by (1.5) are not necessarily (Fréchet) differentiable. We give an example in §1.2.4 below. Thus the classical Newton method to iteratively linearize (1.8) can struggle to converge as we demonstrate later with numerical examples. To overcome this issue, our approach will be to introduce an auxiliary regularized problem that we use to create a sequence of solutions that approach the solution to the non-regularized problem. We will more precisely introduce a regularized function defined from ϕ', parameterized by $\epsilon > 0$, which we call ϕ'_ϵ. We will make the following assumption.

Assumption 1.2.3 (Regularization of ϕ)**.** *For every $\epsilon > 0$, the regularized function satisfies*

$$\phi'_\epsilon(0) = 0, \tag{1.10}$$
$$\phi'_\epsilon \in C^2(\mathbb{R}). \tag{1.11}$$

Next, the regularized function satisfies inequalities similar to (1.3):

$$|\phi'_\epsilon(r) - \phi'_\epsilon(s)| \le \overline{L}|r - s| \quad \forall r, s \in \mathbb{R}, \tag{1.12a}$$
$$(\phi'_\epsilon(r) - \phi'_\epsilon(s))(r - s) \ge \underline{\alpha}(r - s)^2 \quad \forall r, s \in \mathbb{R}, \tag{1.12b}$$

where $0 < \underline{\alpha} \le \overline{L}$ are real constants independent of ϵ. Moreover, in all points $s \in \mathbb{R}$, there holds

$$(\phi'_\epsilon - \phi')(s) \overset{\epsilon\to 0}{\to} 0. \tag{1.13}$$

The regularized function implicitly defines regularized versions a_ϵ and $\boldsymbol{A}_\epsilon$ as in (1.5) through

$$a_\epsilon(s) := \frac{\phi'_\epsilon(s)}{s}, \boldsymbol{A}_\epsilon(\boldsymbol{q}) := a_\epsilon(|\boldsymbol{q}|)\boldsymbol{q}, \tag{1.14}$$

which, by the same reasoning as in Lemma 1.2.2, satisfy

$$\underline{\alpha}\|\nabla(v - w)\|^2 \le (\boldsymbol{A}_\epsilon(\nabla v) - \boldsymbol{A}_\epsilon(\nabla w), \nabla(v - w)) \quad \forall v, w \in H_0^1(\Omega), \tag{1.15a}$$
$$\|\boldsymbol{A}_\epsilon(\nabla v) - \boldsymbol{A}_\epsilon(\nabla w)\| \le \overline{L}\|\nabla(v - w)\| \quad \forall v, w \in H_0^1(\Omega). \tag{1.15b}$$

We now prove a consequence of Assumption 1.2.3 which will be useful later.

Lemma 1.2.4 (L^2 convergence of the regularization)**.** *For any vector field* $\boldsymbol{v} \in L^2(\Omega)$ *we have that*

$$\lim_{\epsilon \to 0} \|\phi'_\epsilon(|\boldsymbol{v}|) - \phi'(|\boldsymbol{v}|)\| \to 0. \tag{1.16}$$

Proof. We will make use of the Lebesgue dominated convergence theorem in $L^2(\Omega)$, see e.g. [113]. Indeed, consider an arbitrary real sequence $\{\epsilon^n\}_{n\in\mathbb{N}}$ tending to zero and consider the sequence of functions

$$g_n(\boldsymbol{x}) := \phi'_{\epsilon^n}(|\boldsymbol{v}(\boldsymbol{x})|) - \phi'(|\boldsymbol{v}(\boldsymbol{x})|).$$

Then by (1.13) from Assumption 1.2.3, we have that $g_n \to 0$ pointwise for almost all $\boldsymbol{x} \in \Omega$. Next, to establish a dominating function, observe that since $\phi'(0) = \phi'_\epsilon(0) = 0$ by (1.2) and (1.10), we have

$$|g_n(x)| \leq |\phi'_{\epsilon^n}(|\boldsymbol{v}(\boldsymbol{x})|)| + |\phi'(|\boldsymbol{v}(\boldsymbol{x})|)| \overset{(1.3a),(1.12a)}{\leq} (\overline{L} + L)|\boldsymbol{v}(\boldsymbol{x})| =: g(\boldsymbol{x}) \in L^2(\Omega),$$

so we can choose g as the dominating function. Finally, since ϵ^n was arbitrary, the sequential criterion for a limit ensures (1.16). □

For algorithmic reasons, we will consider a monotonically decreasing sequence, $\{\epsilon^j\}_{j\geq 0}$ of positive real values which is in particular determined by two values $\epsilon^0 > 0$ and $0 < C_\epsilon < 1$, where, for $j \geq 1$,

$$\epsilon^j := C_\epsilon \epsilon^{j-1}. \tag{1.17}$$

All these considerations lead us to a regularized version of the problem (1.8): for a fixed $j \geq 0$, find $u^j \in H^1_0(\Omega)$ such that

$$(\boldsymbol{A}_{\epsilon^j}(\nabla u^j), \nabla v) = (f, v) \quad \forall v \in H^1_0(\Omega). \tag{1.18}$$

1.2.4 An example nonsmooth nonlinearity with a kink

To make our notions more concrete, we introduce a simple but instructive example for our study. Consider $\phi \in C^1(\mathbb{R}) \setminus C^2(\mathbb{R})$ given by

$$\phi(s) := \begin{cases} \frac{1}{2}(s - s_0)^2 + s_0 s - \frac{1}{2}s_0^2 & s \leq s_0, \\ \frac{m}{2}(s - s_0)^2 + s_0 s - \frac{1}{2}s_0^2, & s > s_0 \end{cases} \tag{1.19}$$

with continuous derivative

$$\phi'(s) = \begin{cases} s, & s \leq s_0, \\ m(s - s_0) + s_0, & s > s_0, \end{cases} \tag{1.20}$$

where $s_0 > 0$ determines the location of the discontinuity in the second derivative and $m \geq 1$ determines the slope to the right of s_0. An illustration is given in Figure 1.2. We call the function (1.20) a kink function due to the fact that $\phi'(s)$ is not strongly differentiable at the point s_0. This function satisfies Assumption 1.2.1 with $L = m$ and $\alpha = 1$ since the weak derivative of ϕ' is given by

$$\phi''(s) = \begin{cases} 1, & s < s_0, \\ m, & s > s_0. \end{cases} \tag{1.21}$$

For this particular choice of function ϕ, applying the standard Newton method leads to failure of convergence as illustrated in the example of §1.10.1.2.

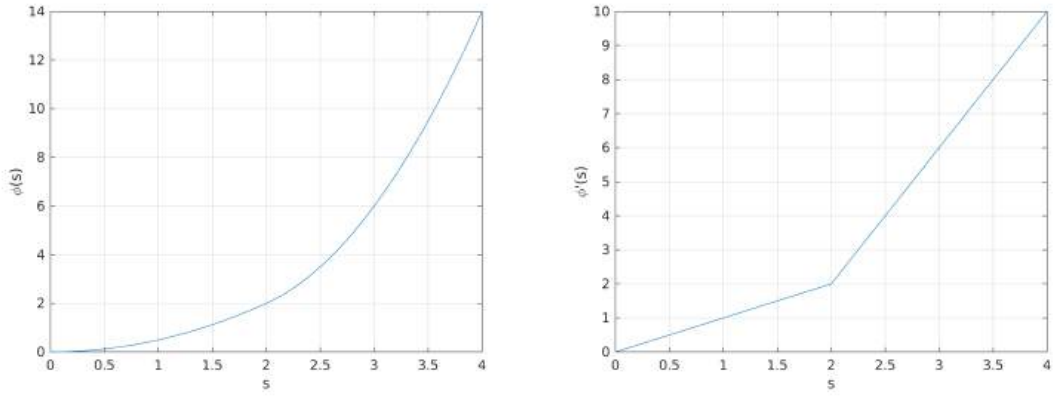


Figure 1.2: [Kink function (1.19) with $m = 4$, $s_0 = 2$] The kink function ϕ (1.19) and its derivative ϕ' (1.20) Notice that ϕ' is not strongly differentiable at s_0.

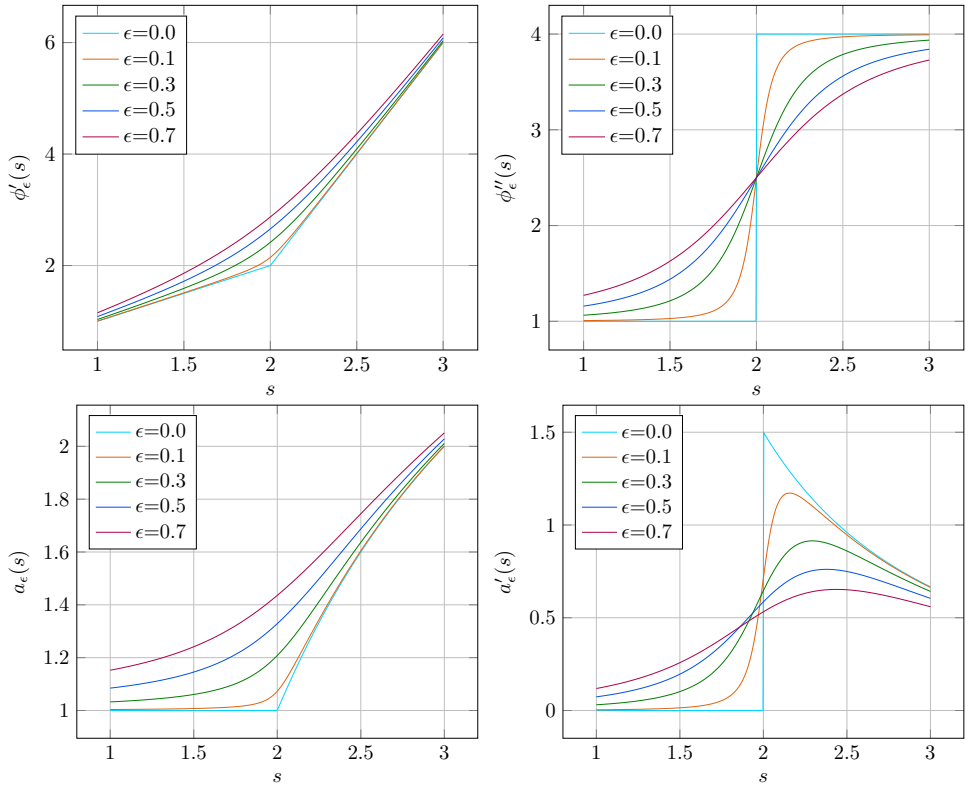


Figure 1.3: [Kink function (1.19) with $m = 4$, $s_0 = 2$] Regularization of the kink function (1.19) by replacing the absolute value function with its differentiable counterpart, see (1.25).

We now introduce a regularized version of the function ϕ' of (1.20) that in turn defines regularized versions of the nonlinear functions a and $\boldsymbol{A}$ in (1.14). We first notice that (1.19) can be equivalently rewritten as

$$\phi'(s) = \frac{m-1}{2}|s - s_0| + \frac{m+1}{2}(s - s_0) + s_0. \tag{1.22}$$

We then consider, for a fixed value of $\epsilon > 0$, the smooth approximation of the absolute value function

$$|s|_\epsilon := \sqrt{s^2 + \epsilon^2}. \tag{1.23}$$

We then replace the absolute value in (1.22) by the smooth version

$$\hat{\phi}'_\epsilon(s) := \frac{m-1}{2}|s - s_0|_\epsilon + \frac{m+1}{2}(s - s_0) + s_0. \tag{1.24}$$

We then set

$$\phi'_\epsilon(s) := \hat{\phi}'_\epsilon(s) - \hat{\phi}'_\epsilon(0) \tag{1.25}$$

to achieve $\phi'_\epsilon(0) = 0$. An illustration is given in Figure 1.3. We now show the following

Lemma 1.2.5 (Example regularization (1.25))**.** *The definition* (1.25) *satisfies Assumption 1.2.3.*

Proof. The requirement (1.10) is obvious from the definition. Next, the function is actually $C^\infty(\mathbb{R})$—in particular $C^2(\mathbb{R})$—so (1.11) is satisfied. Next, for (1.12), observe that

$$\phi''_\epsilon(s) = \frac{m-1}{2}\frac{s}{\sqrt{s^2 + \epsilon^2}} + \frac{m+1}{2}$$

is strictly increasing and $\lim_{s\to-\infty} \phi''_\epsilon(s) = 1$, as well as $\lim_{s\to\infty} \phi''_\epsilon(s) = m$. Thus, $1 \leq \phi''_\epsilon \leq m$ and appealing to, e.g., [68, Proposition A.2], we conclude that (1.12) is satisfied with $\overline{L} = m$ and $1 = \underline{\alpha}$,

Finally, we show that for all $s \in \mathbb{R}$,

$$|s|_\epsilon - |s| = \sqrt{s^2 + \epsilon^2} - \sqrt{s^2} = \frac{\epsilon^2}{\sqrt{s^2 + \epsilon^2} + \sqrt{s^2}} \to 0$$

when $\epsilon \to 0$, which confirms (1.13). □

1.3 Discrete problem and linearization

We now give details for the discretization of the regularized problem (1.18) via the continuous Galerkin finite element method [49] and subsequent linearization.

1.3.1 Finite element discretization

Let $\mathcal{T}_0$ be a simplicial mesh of the physical domain Ω with no "hanging nodes" i.e., $\mathcal{T}_0 = \cup_K\{K\}$, where the intersection of (the closure of) two arbitrary simplices $K, K' \in \mathcal{T}_0$ are either empty or an l-dimensional simplex for $0 \leq l \leq d-1$. From the initial mesh $\mathcal{T}_0$, we generate a hierarchy $\{\mathcal{T}_\ell\}_{\ell=1}^L$ of nested meshes, i.e., $\mathcal{T}_\ell \subset \mathcal{T}_{\ell+1}$ for all $\ell \geq 0$. We assume that each mesh in the hierarchy is also free of hanging nodes in the same sense as for $\mathcal{T}_0$. We also

assume that the hierarchy of meshes is shape regular, i.e., that there exists a constant $\kappa_{\mathcal{T}}$ such that, for all ℓ,

$$\max_{K\in\mathcal{T}_\ell} \kappa_K \leq \kappa_{\mathcal{T}}, \tag{1.26}$$

where $\kappa_K := \frac{h_K}{\rho_K}$, h_K is the diameter of K, and ρ_K is the radius of the largest inscribed ball of K. For an arbitrary collection of simplices $\mathcal{T}$ of some mesh $\mathcal{T}_\ell$ and its corresponding subdomain $\omega \subset \Omega$, we define the broken polynomial space of order $p \geq 0$ by

$$\mathcal{P}_p(\mathcal{T}) := \{v \in L^2(\omega) : v|_K \in \mathcal{P}_p(K), \forall K \in \mathcal{T}\}. \tag{1.27}$$

Finally, we introduce, for a fixed polynomial degree $p \geq 1$ and for each ℓ an $H_0^1(\Omega)$-conforming finite-dimensional space,

$$V_\ell^p := H_0^1(\Omega) \cap \mathcal{P}_p(\mathcal{T}_\ell). \tag{1.28}$$

We now consider a discrete equivalent of the regularized continuous problem (1.18) where we seek, for $j, \ell \geq 0$ the solution $u_\ell^j \in V_\ell^p$ such that

$$(\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j), \nabla v_\ell) = (f, v_\ell) \quad \forall v_\ell \in V_\ell^p. \tag{1.29}$$

Note that while this problem is finite-dimensional, it is still nonlinear.

1.3.2 Linearization

We now define a linearization scheme for the regularized finite-dimensional problem (1.29). For fixed $\ell, j \geq 0$, let $u_\ell^{j,0} \in V_\ell^p$ be the initial guess. Denoting by $k \geq 1$ the linearization iterate, a step of the linearization procedure to approximately solve (1.29) takes the form: find $u_\ell^{j,k} \in V_\ell^p$ such that

$$(\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}), \nabla v) = (f, v) \quad \forall v \in V_\ell^p, \tag{1.30}$$

where the operator $\boldsymbol{A}_{\epsilon^j}^{k-1}$ is affine and takes the form

$$\boldsymbol{A}_{\epsilon^j}^{k-1}(\boldsymbol{q}) := \mathbb{A}_{\epsilon^j}^{k-1}\boldsymbol{q} - \boldsymbol{b}_{\epsilon^j}^{k-1} \tag{1.31}$$

for a matrix-valued function $\mathbb{A}_{\epsilon^j}^{k-1} : \Omega \to \mathbb{R}^{d\times d}$ and a vector-valued function $\boldsymbol{b}_{\epsilon^j}^{k-1} : \Omega \to \mathbb{R}^d$. Once a basis of V_ℓ^p is chosen, the problem (1.30) is equivalent to solving a linear system of algebraic equations.

We make the following assumptions on the linearization.

Assumption 1.3.1 (Assumptions on the linearization)**.** *Let the regularization step $j \geq 0$ and mesh level $\ell \geq 0$ be fixed. We assume that the linearized operator converges in the sense that*

$$\lim_{k\to\infty} \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j)\| = 0. \tag{1.32}$$

Finally, we assume that $\mathbb{A}_{\epsilon^j}^{k-1}$ is uniformly bounded and symmetric positive definite, that is, the following conditions hold for all $\boldsymbol{x} \in \Omega$ and all $\boldsymbol{v} \in \mathbb{R}^d$,

$$|\mathbb{A}_{\epsilon^j}^{k-1}(\boldsymbol{x})\boldsymbol{v}| \leq \overline{\lambda}|\boldsymbol{v}| \qquad \textit{(boundedness)}, \tag{1.33a}$$

$$\underline{\lambda}|\boldsymbol{v}|^2 \leq (\mathbb{A}_{\epsilon^j}^{k-1}(\boldsymbol{x})\boldsymbol{v}) \cdot \boldsymbol{v} \qquad \textit{(positive definiteness)}, \tag{1.33b}$$

where $0 < \underline{\lambda} \leq \overline{\lambda}$ are real constants.

A prototypical example is the Picard, or fixed point, linearization where

$$\boldsymbol{A}_{\epsilon^j}^{k-1}(\boldsymbol{q}) := a_{\epsilon^j}(|\nabla u_\ell^{j,k-1}|)\boldsymbol{q}, \tag{1.34}$$

whereas for a Newton step the linearized function is given by

$$\boldsymbol{A}_{\epsilon^j}^{k-1}(\boldsymbol{q}) := a_{\epsilon^j}(|\nabla u_\ell^{j,k-1}|)\boldsymbol{q} + \big(\partial_{\boldsymbol{q}} a_{\epsilon^j}(|\nabla u_\ell^{j,k-1}|) \otimes \nabla u_\ell^{j,k-1}\big)(\boldsymbol{q} - \nabla u_\ell^{j,k-1}), \tag{1.35}$$

and, in turn,

$$\partial_{\boldsymbol{q}} a_{\epsilon^j}(|\boldsymbol{q}|) = \frac{a'_{\epsilon^j}(|\boldsymbol{q}|)}{|\boldsymbol{q}|}\boldsymbol{q} = \big(\phi''_\epsilon(|\boldsymbol{q}|) - \phi'_\epsilon(|\boldsymbol{q}|)|\boldsymbol{q}|^{-1}\big)|\boldsymbol{q}|^{-2}\boldsymbol{q}. \tag{1.36}$$

In [68, Section 2.3.2] it is demonstrated that these two linearization schemes satisfy (1.33) for $\underline{\lambda} = \underline{\alpha}$ and $\overline{\lambda} = \overline{L}$.

Remark 1.3.2 (Newton linearization)**.** *Note that the derivative $\phi''_\epsilon(s)$ appears in (1.36). For nonsmooth nonlinearities, where merely the function $\phi \in C^1(\mathbb{R}) \setminus C^2(\mathbb{R})$, the second derivative ϕ'' does not exist in a strong sense. This is the motivation for introducing the regularization.*

1.4 Three ways of measuring the error and their mutual relations

We have now established and characterized different solutions, namely the true solution u of (1.8) and the approximate solution to the regularized, discretized, and linearized problem of (1.30) $u_\ell^{j,k}$. The next step is to define a concrete notion of error between these two objects, and then, ideally, to have a computable means of estimating it. We postpone the discussion of error estimation to §1.7. In this section we will recall, following [107, 129, 10], three notions of error between u and $u_\ell^{j,k}$, namely the difference of energies (1.1), a notion related to a (weighted by $\alpha^{1/2}$) $H_0^1(\Omega)$ norm, and the dual (weighted by $\alpha^{-1/2}$) norm of the residual. One salient feature when comparing these error measures is that they all coincide in the linear case where $\phi(s) = s^2/s$ and $\boldsymbol{A}(\boldsymbol{q}) = \boldsymbol{q}$, as will be made precise in §1.4.4. Recall that we assume that the constants L and α are given by Assumption 1.2.1.

1.4.1 Energy difference

A physically-motivated notion of error is the difference of energies. For $v \in H_0^1(\Omega)$, this is given as

$$0 \leq \mathcal{J}(v) - \mathcal{J}(u), \tag{1.37}$$

where the energy functional $\mathcal{J}$ is defined in (1.1). Since the true solution u is the unique minimum of $\mathcal{J}$ in $H_0^1(\Omega)$ as per (1.4), this quantity is guaranteed to be nonnegative and only 0 when $v = u$.

1.4.2 Energy norm

A second type of error measure we will consider will be that of the energy norm. For $v \in H_0^1(\Omega)$ we namely consider

$$|||v||| := \alpha^{1/2}\|\nabla v\|, \tag{1.38}$$

where $\alpha > 0$ is the monotonicity constant from (1.3b). Thus, the error between $v \in H^1_0(\Omega)$ and the solution u of (1.8) is here expressed as

$$0 \leq \tfrac{1}{2}|||u - v|||^2. \tag{1.39}$$

The reason for the choice of squaring and dividing by two will become clear in §1.4.4.

1.4.3 Dual norm of the residual

Finally we consider the abstract error quantity obtained through the dual norm of the residual. First, we define, for a fixed $v \in H^1_0(\Omega)$, the residual functional $\mathcal{R}(v) \in H^{-1}(\Omega)$ by

$$\langle \mathcal{R}(v), w\rangle := (f, w) - (\boldsymbol{A}(\nabla v), \nabla w), \quad w \in H^1_0(\Omega), \tag{1.40}$$

where the duality pairing $\langle\cdot,\cdot\rangle$ is between $H^{-1}(\Omega)$ and $H^1_0(\Omega)$. Next we introduce the dual norm for $\mathcal{R} \in H^{-1}(\Omega)$

$$|||\mathcal{R}(v)|||_{-1} := \sup_{\varphi \in H^1_0(\Omega),\, |||\varphi|||=1} \langle \mathcal{R}(v), \varphi\rangle. \tag{1.41}$$

Here the error in the energy dual norm of the residual is given, for $v \in H^1_0(\Omega)$,

$$0 \leq \tfrac{1}{2}|||\mathcal{R}(v)|||^2_{-1}. \tag{1.42}$$

(1.8). The choice of squaring and dividing by two will again be made clear in §1.4.4. Finally, if we consider the standard definition of the dual norm,

$$\|\mathcal{R}(v)\|_{H^{-1}(\Omega)} := \sup_{\varphi \in H^1_0(\Omega),\, \|\nabla\varphi\|=1} \langle \mathcal{R}(v), \varphi\rangle, \tag{1.43}$$

our definition satisfies the scaling

$$|||\mathcal{R}(v)|||_{-1} = \alpha^{-1/2}\|\mathcal{R}(v)\|_{H^{-1}(\Omega)}. \tag{1.44}$$

1.4.4 Equivalence in the linear case

In this section, we recall the special relationship between the three different error measures of the previous sections in the case where $\phi(s) = s^2/2$, i.e., $\alpha = L = 1$, which writes as

$$\mathcal{J}(v) - \mathcal{J}(u) = \tfrac{1}{2}|||v - u|||^2 = \tfrac{1}{2}|||\mathcal{R}(v)|||^2_{-1}. \tag{1.45}$$

For the sake of completeness, we recall the proof of (1.45). As for the first equality,

$$\begin{aligned}
\mathcal{J}(v) - \mathcal{J}(u) &= \int_\Omega \tfrac{1}{2}|\nabla v|^2 - fv \,\mathrm{d}\boldsymbol{x} - \left(\int_\Omega \tfrac{1}{2}|\nabla u|^2 - fu \,\mathrm{d}\boldsymbol{x}\right) \\
&\overset{(1.8)}{=} \int_\Omega \tfrac{1}{2}|\nabla v|^2 - \nabla u\cdot\nabla v \,\mathrm{d}\boldsymbol{x} - \left(\int_\Omega \tfrac{1}{2}|\nabla u|^2 - \nabla u\cdot\nabla u \,\mathrm{d}\boldsymbol{x}\right) \\
&= \int_\Omega \tfrac{1}{2}|\nabla v|^2 - \nabla u\cdot\nabla v + \tfrac{1}{2}|\nabla u|^2 \,\mathrm{d}\boldsymbol{x} = \tfrac{1}{2}|||u - v|||^2.
\end{aligned}$$

The latter one is then simply

$$\begin{aligned}
|||u - v||| &= \sup_{\varphi \in H^1_0(\Omega),\, \|\nabla\varphi\|=1} (\nabla(u - v), \nabla\varphi) \\
&= \sup_{\varphi \in H^1_0(\Omega),\, \|\nabla\varphi\|=1} \{(f, \varphi) - (\nabla v, \nabla\varphi)\} \\
&= |||\mathcal{R}(v)|||_{-1}.
\end{aligned}$$

1.4.5 Relations in the nonlinear case

In the nonlinear case, the measures are only equivalent up to factors of L/α. Indeed, between the dual norm of the residual and the energy norm, a factor of exactly L/α relates the two:

Proposition 1.4.1 (Relation energy norm-dual residual norm)**.** *Let u solve (1.8) and let $v \in H^1_0(\Omega)$ be arbitrary. Then there holds*

$$\tfrac{1}{2}|||v-u|||^2 \leq \tfrac{1}{2}|||\mathcal{R}(v)|||_{-1}^2 \leq \frac{L^2}{2\alpha^2}|||v-u|||^2. \tag{1.46}$$

Proof. By the definition of the energy norm (1.38) and the dual residual norm (1.41), together with the monotonicity assumption (1.3b) implying (1.6) and definition (1.8),

$$\begin{aligned} |||v-u||| = \alpha^{1/2}\|\nabla(v-u)\| &\overset{(1.6)}{\leq} \frac{(\boldsymbol{A}(\nabla v)-\boldsymbol{A}(\nabla u),\nabla(v-u))}{\alpha^{1/2}\|\nabla(v-u)\|} \\ &\leq \sup_{\varphi\in H^1_0(\Omega),\,|||\varphi|||=1} (\boldsymbol{A}(\nabla v)-\boldsymbol{A}(\nabla u),\nabla\varphi) \\ &\overset{(1.8)}{=} \sup_{\varphi\in H^1_0(\Omega),\,|||\varphi|||=1} \{(\boldsymbol{A}(\nabla v),\nabla\varphi)-(f,\varphi)\} \\ &= |||\mathcal{R}(v)|||_{-1}. \end{aligned} \tag{1.47}$$

For the second inequality, observe that, using the Lipschitz continuity assumption (1.3a) implying (1.7),

$$\begin{aligned} |||\mathcal{R}(v)|||_{-1} &\overset{(1.44)}{=} \alpha^{-1/2}\|\mathcal{R}(v)\|_{H^{-1}(\Omega)} \\ &\overset{(1.8)}{=} \alpha^{-1/2} \sup_{\varphi\in H^1_0(\Omega),\|\nabla\varphi\|=1} (\boldsymbol{A}(\nabla v)-\boldsymbol{A}(\nabla u),\nabla\varphi) \\ &\leq \alpha^{-1/2} \sup_{\varphi\in H^1_0(\Omega),\|\nabla\varphi\|=1} \|\boldsymbol{A}(\nabla v)-\boldsymbol{A}(\nabla u)\|\,\|\nabla\varphi\| \\ &= \alpha^{-1/2}\|\boldsymbol{A}(\nabla v)-\boldsymbol{A}(\nabla u)\| \\ &\overset{(1.7)}{\leq} L\alpha^{-1/2}\|\nabla(v-u)\| \\ &\overset{(1.38)}{=} \frac{L}{\alpha}|||v-u|||. \end{aligned} \tag{1.48}$$

□

When comparing the energy difference with the energy norm, a factor of $\sqrt{L/\alpha}$ is instead introduced.

Proposition 1.4.2. *Let u solve (1.8) and let $v \in H^1_0(\Omega)$ be arbitrary. Then there holds*

$$\tfrac{1}{2}|||v-u|||^2 \leq \mathcal{J}(v)-\mathcal{J}(u) \leq \frac{L}{2\alpha}|||v-u|||^2. \tag{1.49}$$

Proof. See [59, Lemma 5.1]. □

1.5 Duality theory

In order to bound the energy difference as introduced in §1.4.1, we will proceed using duality for convex functions, following [17, 107, 129].

1.5.1 Fenchel conjugate and its properties

Let us introduce the Fenchel conjugate (also known as the Legendre transform). For a convex function ϕ, it is given by

$$\phi^*(r) := \int_0^r (\phi')^{-1}(s)\,\mathrm{d}s. \tag{1.50}$$

We also define the conjugate operator

$$\boldsymbol{A}^*(\boldsymbol{q}) := \frac{(\phi^*)'(|\boldsymbol{q}|)}{|\boldsymbol{q}|}\boldsymbol{q}. \tag{1.51}$$

To illustrate, let us compute explicitly the Fenchel conjugate for the kink example (1.19) of §1.2.4. We first compute the inverse of the derivative

$$(\phi')^{-1}(s) = \begin{cases} s, & s \le s_0 \\ \frac{s+(m-1)s_0}{m}, & s > s_0 \end{cases}. \tag{1.52}$$

Then the Fenchel conjugate (1.50) takes the form

$$\phi^*(r) = \int_0^r (\phi')^{-1}(s)\,\mathrm{d}s = \begin{cases} \frac{1}{2}r^2, & r \le s_0 \\ \frac{1}{2m}[(r+(m-1)s_0)^2 - s_0^2 m(m-1)], & r > s_0 \end{cases}. \tag{1.53}$$

Indeed, for the case where $r \ge s_0$,

$$\begin{aligned} \int_0^r (\phi')^{-1}(s)\,\mathrm{d}s &= \int_0^{s_0} s\,\mathrm{d}s + \int_{s_0}^r \frac{s+(m-1)s_0}{m}\,\mathrm{d}s \\ &= \frac{1}{2m}\left(r^2 + 2(m-1)s_0 r - s_0^2(m-1)\right) \\ &= \frac{1}{2m}[(r+(m-1)s_0)^2 - s_0^2 m(m-1)]. \end{aligned} \tag{1.54}$$

Definition (1.50) yields the following properties, see [107, 17] or Appendix 1.A:

Proposition 1.5.1 (Properties of the Fenchel conjugate)**.** *Let ϕ be a convex function with $\phi(0) = \phi'(0) = 0$ and let $\phi^* : \mathbb{R} \to \mathbb{R}$ be its Fenchel conjugate given by (1.50). Then the following properties hold.*

$$\phi^*(r) = r(\phi')^{-1}(r) - \phi((\phi')^{-1}(r)) = \max_s\{sr - \phi(s)\}, \tag{1.55a}$$

$$\phi^* \text{ is convex}, \tag{1.55b}$$

$$\phi^* \in C^1(\mathbb{R}) \text{ and } (\phi^*)' = (\phi')^{-1}, \tag{1.55c}$$

$$\phi^*(0) = (\phi^*)'(0) = 0. \tag{1.55d}$$

This proposition can be used to derive the following well-known result [107, 129, 17, 106] or Appendix 1.A:

Corollary 1.5.2 (Young's inequality for convex functions)**.** *Let $\phi \in C^1(\mathbb{R})$ be convex and let ϕ^* be given by (1.50). Then*

$$sr \le \phi(s) + \phi^*(r) \quad \text{for all } s, r \ge 0, \tag{1.56}$$

where the equality holds for $r = \phi'(s)$ or equivalently $s = (\phi^*)'(r)$.

Next, we consider the relationship between the vector-valued counterparts.

Corollary 1.5.3 ($\boldsymbol{A}$ and $\boldsymbol{A}^*$)**.** *Let $\boldsymbol{A}$ be given by (1.5) and $\boldsymbol{A}^*$ be given by (1.51). Then the following holds for all $\boldsymbol{q} \in \mathbb{R}^d$*

$$\boldsymbol{A}(\boldsymbol{A}^*(\boldsymbol{q})) = \boldsymbol{q} \tag{1.57a}$$
$$\boldsymbol{A}(\boldsymbol{q}) \cdot \boldsymbol{q} = \phi(|\boldsymbol{q}|) + \phi^*(|\boldsymbol{A}(\boldsymbol{q})|). \tag{1.57b}$$

Finally, there holds:

Lemma 1.5.4 (Lipschitz continuity of $(\phi^*)'$)**.** *Let ϕ satisfy the Assumption 1.2.1. Then $(\phi^*)'$ is Lipschitz continuous with Lipschitz constant equal to α^{-1}, i.e.,*

$$|(\phi^*)'(r) - (\phi^*)'(s)| \leq \alpha^{-1}|r - s|. \tag{1.58}$$

The proof of these results is again given in Appendix 1.A.

1.5.2 The two energies principle

We are led to investigate the dual optimization problem to (1.4). The dual problem to (1.4) can be stated as

$$\boldsymbol{\sigma} := \operatorname*{arg\,max}_{\substack{\boldsymbol{\varsigma} \in \boldsymbol{H}(\operatorname{div},\Omega) \\ \nabla\cdot\boldsymbol{\varsigma} = f}} \mathcal{J}^*(\boldsymbol{\varsigma}), \tag{1.59}$$

where the dual functional $\mathcal{J}^* : \boldsymbol{H}(\operatorname{div},\Omega) \to \mathbb{R}$ is given by

$$\mathcal{J}^*(\boldsymbol{\varsigma}) := -\int_\Omega \phi^*(|\boldsymbol{\varsigma}|)\,\mathrm{d}\boldsymbol{x}. \tag{1.60}$$

It turns out that the flux $\boldsymbol{\sigma} = -\boldsymbol{A}(\nabla u)$, where u is given by (1.4), or, equivalently, by (1.8). We see this in the following way. First of all, the Euler–Langrange conditions for (1.59) are: find $\boldsymbol{\sigma} \in \boldsymbol{H}(\operatorname{div},\Omega)$ with $\nabla\cdot\boldsymbol{\sigma} = f$ such that

$$(\boldsymbol{A}^*(\boldsymbol{\sigma}), \boldsymbol{v}) = 0 \quad \forall \boldsymbol{v} \in \boldsymbol{H}(\operatorname{div},\Omega) \text{ with } \nabla\cdot\boldsymbol{v} = 0. \tag{1.61}$$

We consider the equivalent mixed formulation of (1.61): find $(\boldsymbol{\sigma}, \tilde{u}) \in \boldsymbol{H}(\operatorname{div},\Omega) \times L^2(\Omega)$ such that

$$(\boldsymbol{A}^*(\boldsymbol{\sigma}), \boldsymbol{v}) - (\tilde{u}, \nabla\cdot\boldsymbol{v}) = 0 \qquad \forall \boldsymbol{v} \in \boldsymbol{H}(\operatorname{div},\Omega), \tag{1.62a}$$
$$(\nabla\cdot\boldsymbol{\sigma}, q) = (f, q) \qquad \forall q \in L^2(\Omega). \tag{1.62b}$$

By the definition of the weak gradient, (1.62a) implies

$$\nabla\tilde{u} = -\boldsymbol{A}^*(\boldsymbol{\sigma}) \overset{(1.57a)}{\Longrightarrow} -\boldsymbol{A}(\nabla\tilde{u}) = \boldsymbol{\sigma}. \tag{1.63}$$

Finally taking $q \in H^1_0(\Omega) \subset L^2(\Omega)$ as the test function in (1.62b) shows that $\tilde{u} = u$ is the solution to problem (1.8).

To make the connection between the primal and dual problems more precise, we proceed to introduce the saddle point functional by

$$\mathcal{L}(v, \boldsymbol{\varsigma}) := \mathcal{J}^*(\boldsymbol{\varsigma}) - (\nabla v, \boldsymbol{\varsigma}) - (f, v), \quad \boldsymbol{\varsigma} \in \boldsymbol{H}(\operatorname{div},\Omega),\ v \in H^1_0(\Omega). \tag{1.64}$$

Then we have the following

Lemma 1.5.5 (Two energies principle)**.** *Let u be the solution to the minimization problem (1.4) and $\boldsymbol{\sigma}$ be the solution to (1.59). Let $\mathcal{L}$ be as in (1.64). Then*

$$\max_{\substack{\boldsymbol{\varsigma}\in\boldsymbol{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\varsigma}=f}} \mathcal{J}^*(\boldsymbol{\varsigma}) = \mathcal{J}^*(\boldsymbol{\sigma}) = \mathcal{L}(u,\boldsymbol{\sigma}) = \mathcal{J}(u) = \min_{v\in H^1_0(\Omega)} \mathcal{J}(v). \tag{1.65}$$

There also holds

$$\mathcal{L}(u,\boldsymbol{\sigma}) = \max_{\substack{\boldsymbol{\varsigma}\in\boldsymbol{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\varsigma}=f}} \min_{v\in H^1_0(\Omega)} \mathcal{L}(v,\boldsymbol{\varsigma}). \tag{1.66}$$

Proof. The first and last equalities of (1.65) follow by definition. For the second equality of (1.65), note that from $\boldsymbol{\sigma}\in\boldsymbol{H}(\mathrm{div},\Omega)$ and $\nabla\cdot\boldsymbol{\sigma}=f$, for any $v\in H^1_0(\Omega)$, we have $-(\nabla v,\boldsymbol{\sigma})-(f,v)=0$. For the third equality of (1.65),

$$\begin{aligned}\mathcal{L}(u,\boldsymbol{\sigma}) &= \int_\Omega -\phi^*(|\boldsymbol{\sigma}|) - \nabla u\cdot\boldsymbol{\sigma} - fu\,\mathrm{d}\boldsymbol{x}\\ &\overset{(1.63)}{=} \int_\Omega -\phi^*(|\boldsymbol{A}(\nabla u)|) + \nabla u\cdot\boldsymbol{A}(\nabla u) - fu\,\mathrm{d}\boldsymbol{x} \quad \overset{(1.57b)}{=} \int_\Omega \phi(|\nabla u|) - fu\,\mathrm{d}\boldsymbol{x}\\ &\overset{(1.1)}{=} \mathcal{J}(u).\end{aligned}$$

Finally, (1.66) follows since, as above,

$$\max_{\substack{\boldsymbol{\varsigma}\in\boldsymbol{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\varsigma}=f}} \min_{v\in H^1_0(\Omega)} \mathcal{L}(v,\boldsymbol{\varsigma}) \overset{(1.64)}{=} \max_{\substack{\boldsymbol{\varsigma}\in\boldsymbol{H}(\mathrm{div},\Omega)\\ \nabla\cdot\boldsymbol{\varsigma}=f}} \mathcal{J}^*(\boldsymbol{\varsigma}) = \mathcal{J}^*(\boldsymbol{\sigma}) = \mathcal{L}(u,\boldsymbol{\sigma}).$$

□

The above developments directly lead to an upper bound on the energy difference [109, 129, 9, 17]:

Corollary 1.5.6 (Two energies principle)**.** *Under the assumptions of Lemma 1.5.5, the following holds for any $\boldsymbol{\varsigma}\in\boldsymbol{H}(\mathrm{div},\Omega)$ with $\nabla\cdot\boldsymbol{\varsigma}=f$ and any $v\in H^1_0(\Omega)$,*

$$0 \le \mathcal{J}(v) - \mathcal{J}(u) \le \mathcal{J}(v) - \mathcal{J}^*(\boldsymbol{\varsigma}). \tag{1.67}$$

Proof. We apply the properties of the objects involved,

$$0 \overset{(1.4)}{\le} \mathcal{J}(v) - \mathcal{J}(u) \overset{(1.65)}{=} \mathcal{J}(v) - \mathcal{J}^*(\boldsymbol{\sigma}) \overset{(1.59)}{\le} \mathcal{J}(v) - \mathcal{J}^*(\boldsymbol{\varsigma}). \tag{1.68}$$

□

1.6 Equilibrated flux and its components

In this section, we detail an algorithm to construct a dual object $\boldsymbol{t}_\ell^{j,k}\in\boldsymbol{H}(\mathrm{div},\Omega)$ with $\nabla\cdot\boldsymbol{t}_\ell^{j,k}=f$ as required by the duality theory of §1.5. We consider patch-wise minimizations corresponding to local Neumann mixed finite element problems. This strategy has already been employed in many contexts cf. [21, 53, 41]. First we make the following assumption to simplify the analysis. The treatment of general f has been studied carefully in e.g. [53, 68].

Assumption 1.6.1 (No data oscillation)**.** *We suppose for simplicity that the source term is a piecewise polynomial, $f\in\mathcal{P}_p(\mathcal{T}_\ell)$.*

1.6.1 Equilibrated flux

We begin with some additional geometric information. For a given mesh level $\ell \geq 0$, let $\mathcal{V}_\ell$ be the set of mesh vertices partitioned to $\mathcal{V}_\ell = \mathcal{V}_\ell^{\text{int}} \cup \mathcal{V}_\ell^{\text{ext}}$ by interior and boundary vertices. Next let $\omega_{\boldsymbol{a}}$ be the subdomain corresponding to the set of elements of $\mathcal{T}_\ell$ for which $\boldsymbol{a}$ is a vertex, denoted by $\mathcal{T}_{\boldsymbol{a}}$. We also make use of the hat functions $\psi_{\boldsymbol{a}} \in \mathcal{P}_1(\mathcal{T}_\ell) \cap C^0(\overline{\Omega})$ associated with the vertex $\boldsymbol{a} \in \mathcal{V}_\ell$.

For a collection of simplices $\mathcal{T}$ and their corresponding domain ω, we introduce the broken Raviart–Thomas–Nédélec finite element space [23] of order $p \geq 0$,

$$\boldsymbol{RT}_p(\mathcal{T}) := \{\boldsymbol{v}_\ell \in [L^2(\omega)]^d : \boldsymbol{v}_\ell|_K \in [\mathcal{P}_p(K)]^d + \boldsymbol{x}\mathcal{P}_p(K),\ \forall K \in \mathcal{T}\}. \tag{1.69}$$

Next, to account for normal face continuity, we define the vertex patch space

$$\boldsymbol{V}_\ell^p(\omega_{\boldsymbol{a}}) := \boldsymbol{RT}_p(\mathcal{T}_{\boldsymbol{a}}) \cap \boldsymbol{H}_0(\operatorname{div}, \omega_{\boldsymbol{a}}), \tag{1.70}$$

where $\boldsymbol{H}_0(\operatorname{div}, \omega_{\boldsymbol{a}})$ is the subspace of $\boldsymbol{H}(\operatorname{div}, \omega_{\boldsymbol{a}})$ of functions with vanishing normal trace on $\partial\omega_{\boldsymbol{a}}$ when $\boldsymbol{a} \in \mathcal{V}_\ell^{\text{int}}$ and on $\partial\omega_{\boldsymbol{a}} \setminus \{\psi_{\boldsymbol{a}} > 0\}$ when $\boldsymbol{a} \in \mathcal{V}_\ell^{\text{ext}}$. For $v \in L^2(\Omega)$, define the L^2-projection $\Pi_{\ell,p} v \in \mathcal{P}_p(\mathcal{T}_\ell)$ by $(v - \Pi_{\ell,p} v, v_\ell) = 0$ for all $v_\ell \in \mathcal{P}_p(\mathcal{T}_\ell)$. Note that it acts elementwise. Finally, for $\mathcal{T} \subset \mathcal{T}_\ell$ and the corresponding domain $\omega \subseteq \Omega$, define the mean-free space by $\mathcal{P}_p^*(\mathcal{T}) := \{v \in \mathcal{P}_p(\mathcal{T}) : \int_\omega v \,\mathrm{d}\boldsymbol{x} = 0\}$.

Definition 1.6.2 (Total flux $\boldsymbol{t}_\ell^{j,k}$)**.** *Let $u_\ell^{j,k}$ be the solution to (1.30). For all vertices $\boldsymbol{a} \in \mathcal{V}_\ell$, define $\boldsymbol{t}_{\boldsymbol{a}}^{j,k} \in \boldsymbol{V}_\ell^p(\omega_{\boldsymbol{a}})$ and $q_{\boldsymbol{a}} \in \mathcal{P}_p^*(\mathcal{T}_{\boldsymbol{a}})$ as the solution to the patch-local mixed finite element problem*

$$(\boldsymbol{t}_{\boldsymbol{a}}^{j,k}, \boldsymbol{v}_\ell)_{\omega_{\boldsymbol{a}}} - (q_{\boldsymbol{a}}, \nabla\cdot\boldsymbol{v}_\ell)_{\omega_{\boldsymbol{a}}} = -(\psi_{\boldsymbol{a}} \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}), \boldsymbol{v}_\ell)_{\omega_{\boldsymbol{a}}}, \tag{1.71a}$$

$$(\nabla\cdot\boldsymbol{t}_{\boldsymbol{a}}^{j,k}, r_\ell)_{\omega_{\boldsymbol{a}}} = (f\psi_{\boldsymbol{a}} - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) \cdot \nabla\psi_{\boldsymbol{a}}, r_\ell)_{\omega_{\boldsymbol{a}}} \tag{1.71b}$$

for all pairs $(\boldsymbol{v}_\ell, r_\ell) \in \boldsymbol{V}_\ell^p(\omega_{\boldsymbol{a}}) \times \mathcal{P}_p^*(\mathcal{T}_{\boldsymbol{a}})$. After solving this local problem on each patch and extending $\boldsymbol{t}_{\boldsymbol{a}}^{j,k}$ by $\boldsymbol{0}$ outside of $\omega_{\boldsymbol{a}}$, assemble the global flux by

$$\boldsymbol{t}_\ell^{j,k} = \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \boldsymbol{t}_{\boldsymbol{a}}^{j,k}. \tag{1.71c}$$

The patch problem (1.71) is equivalent to solving the local minimization problem,

$$\boldsymbol{t}_{\boldsymbol{a}}^{j,k} := \operatorname*{arg\,min}_{\substack{\boldsymbol{v}_\ell \in \boldsymbol{V}_\ell^p(\omega_{\boldsymbol{a}}) \\ \nabla\cdot\boldsymbol{v}_\ell = \Pi_{\ell,p}(\psi_{\boldsymbol{a}} f - \nabla\psi_{\boldsymbol{a}} \cdot \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}))}} \|\psi_{\boldsymbol{a}} \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) + \boldsymbol{v}_\ell\|_{\omega_{\boldsymbol{a}}}. \tag{1.72}$$

We study the wall time cost of constructing the total flux (1.71) in Appendix 4.10. The flux satisfies the divergence constraint.

Lemma 1.6.3 (Divergence of the equilibrated flux)**.** *Given Assumption 1.6.1, the flux $\boldsymbol{t}_\ell^{j,k}$ given by Definition 1.6.2 satisfies*

$$\nabla\cdot\boldsymbol{t}_\ell^{j,k} = f. \tag{1.73}$$

Proof. By construction,

$$\nabla\cdot\boldsymbol{t}_\ell^{j,k} = \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \nabla\cdot\boldsymbol{t}_{\boldsymbol{a}}^{j,k} = \Pi_{\ell,p} \left(\sum_{\boldsymbol{a} \in \mathcal{V}_\ell} (\psi_{\boldsymbol{a}} f - \nabla\psi_{\boldsymbol{a}} \cdot \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})) \right) = \Pi_{\ell,p} f = f,$$

see e.g. [21, 53]. □

We have the following stability result obtained by proceeding as in [112].

Lemma 1.6.4 (Stability of the equilibrated flux)**.** *For a fixed vertex $\boldsymbol{a} \in \mathcal{V}_\ell$, the solution to the patch problem (1.71) satisfies*

$$\|\boldsymbol{t}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} \lesssim \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathbf{a}}} + h_{\omega_{\mathbf{a}}}\|\psi_{\boldsymbol{a}} f\|_{\omega_{\mathbf{a}}}. \tag{1.74}$$

The hidden constant only depends on the space dimension d and the mesh shape regularity constant $\kappa_{\mathcal{T}}$ of (1.26).

Proof. We first recall the result used in [112, Lemma 4.1], i.e., for any $\boldsymbol{\tau}_{\boldsymbol{a}} \in \boldsymbol{RT}_p(\mathcal{T}_{\boldsymbol{a}})$ and $g_{\boldsymbol{a}} \in \mathcal{P}_p(\mathcal{T}_{\boldsymbol{a}})$,

$$\min_{\substack{\boldsymbol{v}_\ell \in \boldsymbol{V}_\ell^p(\omega_{\mathbf{a}}) \\ \nabla\cdot\boldsymbol{v}_\ell = g_{\boldsymbol{a}}}} \|\boldsymbol{\tau}_{\boldsymbol{a}} + \boldsymbol{v}_\ell\|_{\omega_{\mathbf{a}}} \le \sup_{\substack{v \in H^1_*(\omega_{\mathbf{a}}) \\ \|\nabla v\|_{\omega_{\mathbf{a}}} = 1}} \{(g_{\boldsymbol{a}}, v)_{\omega_{\mathbf{a}}} - (\boldsymbol{\tau}_{\boldsymbol{a}}, \nabla v)_{\omega_{\mathbf{a}}}\}, \tag{1.75}$$

where $H^1_*(\omega_{\mathbf{a}})$ is the subspace of functions in $H^1(\omega_{\mathbf{a}})$ that have mean value zero on the patch subdomain $\omega_{\mathbf{a}}$ if $\boldsymbol{a} \in \mathcal{V}_\ell^{\mathrm{int}}$ is an interior vertex, or that vanish on $\partial\omega_{\mathbf{a}} \cap \{\psi_{\boldsymbol{a}} > 0\}$ when $\boldsymbol{a} \in \mathcal{V}_\ell^{\mathrm{ext}}$ is a boundary vertex.

Next, set

$$g_{\boldsymbol{a}}^{j,k} := \Pi_{\ell,p}(\psi_{\boldsymbol{a}} f - \nabla\psi_{\boldsymbol{a}} \cdot \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})), \quad \boldsymbol{\tau}_{\boldsymbol{a}}^{j,k} := \psi_{\boldsymbol{a}} \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}). \tag{1.76}$$

For technical reasons, we will need to introduce another auxiliary problem. First, we introduce $\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RT}}$, the $[L^2]^d$-orthogonal projection from $[L^2(\Omega)]^d$ to $\boldsymbol{RT}_{p-1}(\mathcal{T}_\ell)$. Note that it acts elementwise. We consider the vector-valued data

$$\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k} := \psi_{\boldsymbol{a}} \boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RT}}(\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})) \tag{1.77}$$

and the associated minimization problem

$$\tilde{\boldsymbol{t}}_{\boldsymbol{a}}^{j,k} := \min_{\substack{\boldsymbol{v}_\ell \in \boldsymbol{V}_\ell^p(\omega_{\mathbf{a}}) \\ \nabla\cdot\boldsymbol{v}_\ell = g_{\boldsymbol{a}}^{j,k}}} \|\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k} + \boldsymbol{v}_\ell\|_{\omega_{\mathbf{a}}}. \tag{1.78}$$

We are now prepared to derive the bound (1.74). We start with

$$\begin{aligned}
\|\boldsymbol{t}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} &\le \|\boldsymbol{t}_{\boldsymbol{a}}^{j,k} + \boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} + \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} \\
&\overset{(1.72)}{\le} \|\tilde{\boldsymbol{t}}_{\boldsymbol{a}}^{j,k} + \boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} + \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} \\
&\le \|\tilde{\boldsymbol{t}}_{\boldsymbol{a}}^{j,k} + \tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} + \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k} - \tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} + \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} \\
&\overset{(1.78)}{=} \min_{\substack{\boldsymbol{v}_\ell \in \boldsymbol{V}_\ell^p(\omega_{\mathbf{a}}) \\ \nabla\cdot\boldsymbol{v}_\ell = g_{\boldsymbol{a}}^{j,k}}} \|\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k} + \boldsymbol{v}_\ell\|_{\omega_{\mathbf{a}}} + \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k} - \tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} + \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} \\
&\overset{(1.75)}{\le} \underbrace{\sup_{\substack{v \in H^1_*(\omega_{\mathbf{a}}) \\ \|\nabla v\|_{\omega_{\mathbf{a}}} = 1}} \{\underbrace{(g_{\boldsymbol{a}}^{j,k}, v)_{\omega_{\mathbf{a}}}}_{a} - \underbrace{(\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}, \nabla v)_{\omega_{\mathbf{a}}}}_{b}\}}_{\mathfrak{T}_1} + \underbrace{\|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k} - \tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}}}_{\mathfrak{T}_2} + \underbrace{\|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}}}_{\mathfrak{T}_3}.
\end{aligned}$$

We now bound these three terms individually. For the third term,

$$\begin{aligned}
\mathfrak{T}_3 &= \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathbf{a}}} \\
&\overset{(1.76)}{=} \|\psi_{\boldsymbol{a}} \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathbf{a}}} \\
&\le \|\psi_{\boldsymbol{a}}\|_{L^\infty(\omega_{\mathbf{a}})} \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathbf{a}}} \\
&= \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathbf{a}}}.
\end{aligned} \tag{1.79}$$

For the second term,

$$\begin{aligned}\mathfrak{T}_2 &\leq \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\| + \|\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathrm{a}}} \\ &\overset{(1.77)}{=} \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\| + \|\psi_{\boldsymbol{a}}\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RT}}(\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}))\|_{\omega_{\mathrm{a}}} \\ &\leq \|\boldsymbol{\tau}_{\boldsymbol{a}}^{j,k}\| + \|\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RT}}(\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}))\|_{\omega_{\mathrm{a}}} \\ &\overset{(1.79)}{\leq} \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}}.\end{aligned}$$

Finally, for the first term, fix $v \in H_*^1(\omega_{\mathrm{a}})$ with $\|\nabla v\|_{H_*^1(\omega_{\mathrm{a}})} = 1$. For term b,

$$(\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}, \nabla v)_{\omega_{\mathrm{a}}} \overset{\text{C.S.}}{\leq} \|\tilde{\boldsymbol{\tau}}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathrm{a}}},$$

whence the preceding arguments can be applied again. For term a,

$$\begin{aligned}(g_{\boldsymbol{a}}^{j,k}, v)_{\omega_{\mathrm{a}}} &\overset{\text{C.S.}}{\leq} \|g_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathrm{a}}}\|v\|_{\omega_{\mathrm{a}}} \\ &\overset{\text{Poincaré}}{\lesssim} h_{\omega_{\mathrm{a}}}\|g_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathrm{a}}} \\ &\overset{(1.76)}{=} h_{\omega_{\mathrm{a}}}\|\Pi_{\ell,p}(\psi_{\boldsymbol{a}} f - \nabla\psi_{\boldsymbol{a}} \cdot \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}))\|_{\omega_{\mathrm{a}}} \\ &\leq h_{\omega_{\mathrm{a}}}\|\psi_{\boldsymbol{a}} f - \nabla\psi_{\boldsymbol{a}} \cdot \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}} \\ &\leq h_{\omega_{\mathrm{a}}}\left(\|\psi_{\boldsymbol{a}} f\|_{\omega_{\mathrm{a}}} + \|\nabla\psi_{\boldsymbol{a}} \cdot \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}}\right) \\ &\leq h_{\omega_{\mathrm{a}}}\left(\|\psi_{\boldsymbol{a}} f\|_{\omega_{\mathrm{a}}} + \|\nabla\psi_{\boldsymbol{a}}\|_{L^\infty(\omega_{\mathrm{a}})}\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}}\right) \\ &\lesssim h_{\omega_{\mathrm{a}}}\left(\|\psi_{\boldsymbol{a}} f\|_{\omega_{\mathrm{a}}} + h_{\omega_{\mathrm{a}}}^{-1}\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}}\right) \\ &= h_{\omega_{\mathrm{a}}}\|\psi_{\boldsymbol{a}} f\|_{\omega_{\mathrm{a}}} + \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}}.\end{aligned}$$

Combining these terms concludes the proof. □

1.6.2 Component fluxes

In addition to the flux in Definition 1.6.2, we introduce three more fluxes. The idea, as in [53], is that $\boldsymbol{t}_\ell^{j,k}$ will contain information about the total error, and the additional fluxes will isolate components of the error. This definition is more precisely intended to distinguish the errors coming from regularization, linearization, and discretization.

Definition 1.6.5 (Decomposition of $\boldsymbol{t}_\ell^{j,k}$ into components)**.** *Let $u_\ell^{j,k}$ be the solution to (1.30), for $\ell \geq 0, j \geq 0$, and $k \geq 1$. Let $\boldsymbol{A}, \boldsymbol{A}_{\epsilon^j}$, and $\boldsymbol{A}_{\epsilon^j}^{k-1}$ be given by (1.5), (1.14), and (1.31), respectively. Then define*

$$\boldsymbol{r}_\ell^{j,k} := \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k}) - \boldsymbol{A}(\nabla u_\ell^{j,k}) \qquad \text{[regularization error flux]}, \tag{1.80a}$$

$$\boldsymbol{l}_\ell^{j,k} := \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k}) \qquad \text{[linearization error flux]}, \tag{1.80b}$$

$$\boldsymbol{d}_\ell^{j,k} := \boldsymbol{t}_\ell^{j,k} + \boldsymbol{A}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) \qquad \text{[discretization flux]}. \tag{1.80c}$$

Based on our Assumption 1.2.3 on ϕ_ϵ', we have the following result.

Lemma 1.6.6 (H^1_0-convergence of the regularized approximation)**.** *Consider a discrete version of* (1.8)*, i.e., find* $u_\ell \in V_\ell^p$ *such that*

$$(\boldsymbol{A}(\nabla u_\ell), \nabla v_\ell) = (f, v_\ell) \quad \forall v \in V_\ell^p. \tag{1.81}$$

Then the solution to the regularized discrete problem (1.29) *satisfies*

$$\lim_{j\to\infty} \|\nabla(u_\ell - u_\ell^j)\| = 0. \tag{1.82}$$

Proof. Using the strong monotonicity of $\boldsymbol{A}_{\epsilon^j}$,

$$\begin{aligned}
\underline{\alpha}\|\nabla(u_\ell - u_\ell^j)\|^2 &\overset{(1.15a)}{\leq} \big(\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j), \nabla(u_\ell - u_\ell^j)\big) \\
&= \big(\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell) + \boldsymbol{A}(\nabla u_\ell) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j), \nabla(u_\ell - u_\ell^j)\big) \\
&\overset{(1.29),(1.81)}{=} \big(\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell), \nabla(u_\ell - u_\ell^j)\big) \\
&\overset{\text{C.S.}}{\leq} \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\|\|\nabla(u_\ell - u_\ell^j)\| \\
&\overset{(1.5),(1.14)}{=} \|\phi'_{\epsilon^j}(|\nabla u_\ell|) - \phi'(|\nabla u_\ell|)\|\|\nabla(u_\ell - u_\ell^j)\|.
\end{aligned}$$

Thus, $\|\nabla(u_\ell - u_\ell^j)\| \leq \underline{\alpha}^{-1}\|\phi'_{\epsilon^j}(|\nabla u_\ell|) - \phi'(|\nabla u_\ell|)\| \overset{j\to\infty}{\longrightarrow} 0$ by (1.16). □

We have a similar result for the linearized problem.

Lemma 1.6.7 (H^1_0-convergence of the linearized approximation)**.** *Let the regularization step* $j \geq 0$ *and mesh level* $\ell \geq 0$ *be fixed. Then*

$$\lim_{k\to\infty} \|\nabla(u_\ell^j - u_\ell^{j,k})\| = 0. \tag{1.83}$$

Proof. We use the coercivity of the linearization matrix of Assumption 1.3.1,

$$\begin{aligned}
\underline{\lambda}\|\nabla(u_\ell^{j,k} - u_\ell^j)\|^2 &\overset{(1.33b)}{\leq} (\mathbb{A}_{\epsilon^j}^{k-1}\nabla(u_\ell^{j,k} - u_\ell^j), \nabla(u_\ell^{j,k} - u_\ell^j)) \\
&\overset{(1.31)}{=} (\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j), \nabla(u_\ell^{j,k} - u_\ell^j)) \\
&= (\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) + \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j), \nabla(u_\ell^{j,k} - u_\ell^j)) \\
&\overset{(1.29),(1.30)}{=} (\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j), \nabla(u_\ell^{j,k} - u_\ell^j)) \\
&\overset{\text{C.S.}}{\leq} \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j)\|\|\nabla(u_\ell^{j,k} - u_\ell^j)\|.
\end{aligned}$$

Thus, $\|\nabla(u_\ell^{j,k} - u_\ell^j)\| \leq \underline{\lambda}^{-1}\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j)\| \overset{k\to\infty}{\longrightarrow} 0$ by (1.32). □

We are now prepared to prove the following.

Lemma 1.6.8 (Convergence of the regularization error flux)**.** *For a fixed mesh index* $\ell \geq 0$*, the regularization error flux* $\boldsymbol{r}_\ell^{j,k}$ *given in* (1.80a) *satisfies*

$$\lim_{j,k\to\infty} \|\boldsymbol{r}_\ell^{j,k}\| = 0. \tag{1.84}$$

Proof. From the definition of the regularization component flux,

$$\begin{aligned}
\lim_{j,k\to\infty} \|\boldsymbol{r}_\ell^{j,k}\| &\overset{(1.80a)}{=} \lim_{j,k\to\infty} \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k}) - \boldsymbol{A}(\nabla u_\ell^{j,k})\| \\
&\leq \lim_{j,k\to\infty} \Big(\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell)\| + \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\| + \|\boldsymbol{A}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell^{j,k})\| \Big) \\
&\overset{(1.15b),(1.7)}{\leq} \lim_{j,k\to\infty} \Big(\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\| + (L+\overline{L})\|\nabla(u_\ell - u_\ell^{j,k})\| \Big) \\
&\leq \lim_{j,k\to\infty} \Big(\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\| + (L+\overline{L})(\|\nabla(u_\ell - u_\ell^{j}\| + \|\nabla(u_\ell^j - u_\ell^{j,k})\|) \Big) \\
&\overset{(1.83),(1.82)}{=} \lim_{j\to\infty} \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\| \\
&\overset{(1.5),(1.14)}{=} \lim_{j\to\infty} \|\phi'_{\epsilon^j}(|\nabla u_\ell|) - \phi'(|\nabla u_\ell|)\| \overset{(1.16)}{=} 0.
\end{aligned}$$

□

We now turn our attention to the convergence of the linearization error flux component.

Lemma 1.6.9 (Convergence of the linearization error flux)**.** *Let the regularization step $j \geq 0$ and mesh level $\ell \geq 0$ be fixed. The linearization error flux $\boldsymbol{l}_\ell^{j,k}$ given by (1.80b) satisfies*

$$\lim_{k\to\infty} \|\boldsymbol{l}_\ell^{j,k}\| = 0. \tag{1.85}$$

Proof. From the definition of the linearization error flux,

$$\begin{aligned}
\lim_{k\to\infty} \|\boldsymbol{l}_\ell^{j,k}\| &\overset{(1.80b)}{=} \lim_{k\to\infty} \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k})\| \\
&\leq \lim_{k\to\infty} \Big(\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j})\| + \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j})\| \\
&\quad + \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k})\| \Big) \\
&\overset{(1.32)}{=} \lim_{k\to\infty} \Big(\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j})\| + \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k})\| \Big) \\
&\overset{(1.15b)}{\leq} \lim_{k\to\infty} \Big(\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j})\| + \overline{L}\|\nabla(u_\ell^j - u_\ell^{j,k})\| \Big) \\
&\overset{(1.31)}{=} \lim_{k\to\infty} \Big(\|\mathbb{A}_{\epsilon^j}^{k-1}\nabla(u_\ell^{j,k} - u_\ell^{j})\| + \overline{L}\|\nabla(u_\ell^j - u_\ell^{j,k})\| \Big) \\
&\overset{(1.33a)}{\leq} \lim_{k\to\infty} (\overline{\lambda} + \overline{L})\|\nabla(u_\ell^j - u_\ell^{j,k})\| \overset{(1.83)}{=} 0.
\end{aligned}$$

□

Combining these results with the stability of the equilibrated flux in Lemma 1.6.4 results in the following.

Lemma 1.6.10 (Boundedness of the total equilibrated flux)**.** *The equilibrated flux $\boldsymbol{t}_\ell^{j,k}$ of (1.71c) is bounded in the indices j and k, i.e.,*

$$\lim_{j,k\to\infty} \|\boldsymbol{t}_\ell^{j,k}\| = C_{f,\ell} < \infty. \tag{1.86}$$

Proof. We first observe that

$$\|\boldsymbol{t}_\ell^{j,k}\|^2 \le (d+1)\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}\|\boldsymbol{t}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathrm{a}}}^2. \tag{1.87}$$

Now, for a fixed $\boldsymbol{a}, j, k$, letting $C_{f,\boldsymbol{a}} := h_{\omega_{\mathrm{a}}}\|\psi_{\boldsymbol{a}} f\|_{\omega_{\mathrm{a}}}$,

$$\begin{aligned}
\|\boldsymbol{t}_{\boldsymbol{a}}^{j,k}\|_{\omega_{\mathrm{a}}} &\overset{(1.74)}{\lesssim} \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|_{\omega_{\mathrm{a}}} + C_{f,\boldsymbol{a}}\\
&\le \|\boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}} + \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}} + \|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j)\|_{\omega_{\mathrm{a}}} + C_{f,\boldsymbol{a}}\\
&\le \|\boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}} + \underbrace{\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}}}_{\mathfrak{T}_1} + \underbrace{\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j)\|_{\omega_{\mathrm{a}}}}_{\mathfrak{T}_2}\\
&+ \underbrace{\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^j)\|_{\omega_{\mathrm{a}}}}_{\mathfrak{T}_3} + C_{f,\boldsymbol{a}}
\end{aligned}$$

First, we have

$$\begin{aligned}
\lim_{j\to\infty}\mathfrak{T}_1 &\le \lim_{j\to\infty}\Big(\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^j) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}} + \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}}\Big)\\
&\overset{(1.15b)}{\le} \lim_{j\to\infty}\Big(\overline{L}\|\nabla(u_\ell^j - u_\ell)\|_{\omega_{\mathrm{a}}} + \|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell) - \boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}}\Big)\\
&\overset{(1.5)}{=} \lim_{j\to\infty}\Big(\overline{L}\|\nabla(u_\ell^j - u_\ell)\|_{\omega_{\mathrm{a}}} + \|\phi'_{\epsilon^j}(|\nabla u_\ell|) - \phi'(|\nabla u_\ell|)\|_{\omega_{\mathrm{a}}}\Big) \overset{(1.16),(1.82)}{=} 0.
\end{aligned}$$

Next,

$$\lim_{j,k\to\infty}\mathfrak{T}_2 \overset{(1.32)}{=} 0.$$

Finally, we have

$$\lim_{j,k\to\infty}\mathfrak{T}_3 \overset{(1.31)}{=} \lim_{j,k\to\infty}\|\mathbb{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k} - u_\ell^j)\|_{\omega_{\mathrm{a}}} \overset{(1.33a)}{\le} \lim_{j,k\to\infty}\overline{\lambda}\|\nabla(u_\ell^{j,k} - u_\ell^j)\|_{\omega_{\mathrm{a}}} \overset{(1.83)}{=} 0.$$

Combining these results with (1.87), we conclude

$$\lim_{j,k\to\infty}\|\boldsymbol{t}_\ell^{j,k}\|^2 \le (d+1)\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\|\boldsymbol{A}(\nabla u_\ell)\|_{\omega_{\mathrm{a}}} + C_{f,\boldsymbol{a}})^2 =: (C_{f,\ell})^2.$$

□

Lemma 1.6.11 (Convergence of the discretization flux)**.** *For a fixed mesh index $\ell \ge 0$, the discretization flux $\boldsymbol{d}_\ell^{j,k}$ given in (1.80c) satisfies*

$$\lim_{j,k\to\infty}\|\boldsymbol{d}_\ell^{j,k} - \boldsymbol{t}_\ell^{j,k}\| = 0. \tag{1.88}$$

Consequently, we have that the discretization flux satisfies

$$\lim_{j,k\to\infty}\|\boldsymbol{d}_\ell^{j,k}\| = C_{f,\ell} < \infty, \tag{1.89}$$

where $C_{f,\ell}$ is given in (1.86).

Proof. Equation (1.88) is a direct consequence of Lemmas 1.6.8 and 1.6.9 taken with the definition (1.80c). For (1.89),

$$\lim_{j,k\to\infty} \|\boldsymbol{d}_\ell^{j,k}\| \leq \lim_{j,k\to\infty} \Big(\|\boldsymbol{d}_\ell^{j,k} - \boldsymbol{t}_\ell^{j,k}\| + \|\boldsymbol{t}_\ell^{j,k}\| \Big) \overset{(1.88),(1.86)}{\leq} C_{f,\ell}. \tag{1.90}$$

□

We use this fact to justify calling $\boldsymbol{d}_\ell^{j,k}$ the discretization flux. Indeed, the total flux $\boldsymbol{t}_\ell^{j,k}$ is captured by the discretization component $\boldsymbol{d}_\ell^{j,k}$ upon convergence of both the regularization and linearization indices j and k respectively. We now present the results separately for the three ways of measuring the error introduced in §1.4.

1.7 A posteriori error estimates distinguishing the error components

In this section we present error estimates that provide an upper bound and decompose the total error in a given numerical solution. The components estimate the error due to regularization, discretization, and linearization.

1.7.1 Energy difference

The results of §1.5 lead us directly to the following upper bound on the error in the difference of energies.

Proposition 1.7.1 (Upper bound on the energy difference)**.** *Let $u \in H_0^1(\Omega)$ be the solution to (1.4), and let $\mathcal{J}$ and $\mathcal{J}^*$ be given by (1.1) and (1.60), respectively. For a mesh index $\ell \geq 0$, a regularization step $j \geq 0$, and linearization step $k \geq 1$, let $u_\ell^{j,k} \in V_\ell^p$ be the solution to (1.30) and $\boldsymbol{t}_\ell^{j,k}$ be given by Definition 1.6.2. Then there holds*

$$0 \leq \underbrace{\mathcal{J}(u_\ell^{j,k}) - \mathcal{J}(u)}_{\text{total error } (e_{\mathrm{tot}}^{\ell,j,k})^2} \leq \underbrace{\mathcal{J}(u_\ell^{j,k}) - \mathcal{J}^*(\boldsymbol{t}_\ell^{j,k})}_{\text{total est. } (\eta_{\mathrm{tot}}^{\ell,j,k})^2}. \tag{1.91}$$

Proof. We apply Corollary 1.5.6 with $v = u_\ell^{j,k}$ and $\boldsymbol{\varsigma} = \boldsymbol{t}_\ell^{j,k}$. □

Remark 1.7.2 (Equivalent definition of the total estimator)**.** *The definition of the total estimator (1.91) has an equivalent form that is more amenable to local adaptive mesh refinement. This has been already discussed in, e.g., [13, Proposition 4.9]. Indeed, we have that*

$$\begin{aligned}
(\eta_{\mathrm{tot}}^{\ell,j,k})^2 &\overset{(1.91)}{=} \mathcal{J}(u_\ell^{j,k}) - \mathcal{J}^*(\boldsymbol{t}_\ell^{j,k}) \\
&\overset{(1.1),(1.60)}{=} \int_\Omega \phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{t}_\ell^{j,k}|) - f u_\ell^{j,k} \,\mathrm{d}\boldsymbol{x} \\
&\overset{(1.73)}{=} \int_\Omega \phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{t}_\ell^{j,k}|) - (\nabla\cdot\boldsymbol{t}_\ell^{j,k}) u_\ell^{j,k} \,\mathrm{d}\boldsymbol{x} \\
&\overset{\text{I.B.P.}}{=} \int_\Omega \phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{t}_\ell^{j,k}|) + \nabla u_\ell^{j,k} \cdot \boldsymbol{t}_\ell^{j,k} \,\mathrm{d}\boldsymbol{x} \\
&= \sum_{K\in\mathcal{T}_\ell} \int_K \underbrace{\phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{t}_\ell^{j,k}|) + \nabla u_\ell^{j,k} \cdot \boldsymbol{t}_\ell^{j,k}}_{\eta_{\mathrm{tot},K}^{\ell,j,k} \geq 0 \text{ by (1.56)}} \,\mathrm{d}\boldsymbol{x}.
\end{aligned} \tag{1.92}$$

The advantage of this definition is that the last integrand is non-negative by the generalized Young's inequality for convex functions (1.56)*. Indeed, the reasoning goes as follows:*

$$\phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{t}_\ell^{j,k}|) \overset{(1.56)}{\geq} |\nabla u_\ell^{j,k}||\boldsymbol{t}_\ell^{j,k}| \overset{C.S.}{\geq} -\nabla u_\ell^{j,k} \cdot \boldsymbol{t}_\ell^{j,k}. \tag{1.93}$$

.

We now present a decomposition of total estimator, employing Definition 1.6.2 and (1.92).

Theorem 1.7.3 (Decomposition of the energy difference upper bound)**.** *Let the assumptions of Proposition 1.7.1 hold. Let in addition $\boldsymbol{d}_\ell^{j,k}, \boldsymbol{r}_\ell^{j,k}$, and $\boldsymbol{l}_\ell^{j,k}$ be given by Definition 1.6.5. Then the total estimator in* (1.91) *can be further bounded from above as*

$$\begin{aligned}(\eta_{\text{tot}}^{\ell,j,k})^2 \leq & \underbrace{\left| \int_\Omega \phi(|\nabla u_\ell^{j,k}|) + \phi^*(|\boldsymbol{d}_\ell^{j,k}|) + \nabla u_\ell^{j,k} \cdot \boldsymbol{t}_\ell^{j,k} \,\mathrm{d}\boldsymbol{x} \right|}_{\text{discretization est. } (\eta_{\text{dis}}^{\ell,j,k})^2} \\ & + \underbrace{\left| \int_\Omega \phi^*(|\boldsymbol{d}_\ell^{j,k} + \boldsymbol{r}_\ell^{j,k}|) - \phi^*(|\boldsymbol{d}_\ell^{j,k}|) \,\mathrm{d}\boldsymbol{x} \right|}_{\text{regularization est. } (\eta_{\text{reg}}^{\ell,j,k})^2} \\ & + \underbrace{\left| \int_\Omega \phi^*(|\boldsymbol{t}_\ell^{j,k}|) - \phi^*(|\boldsymbol{d}_\ell^{j,k} + \boldsymbol{r}_\ell^{j,k}|) \,\mathrm{d}\boldsymbol{x} \right|}_{\text{linearization est. } (\eta_{\text{lin}}^{\ell,j,k})^2}.\end{aligned} \tag{1.94}$$

Proof. The proof follows by adding and subtracting $\phi^*(|\boldsymbol{d}_\ell^{j,k} + \boldsymbol{r}_\ell^{j,k}|)$ and $\phi^*(|\boldsymbol{d}_\ell^{j,k}|)$ to the integrand of (1.92) and using the triangle inequality. □

We now show our definition of the regularization component behaves in the way that we would expect.

Lemma 1.7.4 (Convergence of the regularization error estimator)**.** *The regularization component estimator $\eta_{\text{reg}}^{\ell,j,k}$ of* (1.94) *tends to 0 as $j, k \to \infty$.*

Proof. From the definition of the regularization component estimator,

$$\begin{aligned}\lim_{j,k\to\infty} (\eta_{\text{reg}}^{\ell,j,k})^2 &\overset{(1.94)}{=} \lim_{j,k\to\infty} \left| \int_\Omega \phi^*(|\boldsymbol{d}_\ell^{j,k}|) - \phi^*(|\boldsymbol{d}_\ell^{j,k} + \boldsymbol{r}_\ell^{j,k}|) \,\mathrm{d}\boldsymbol{x} \right| \\ &= \lim_{j,k\to\infty} \left| \int_\Omega \int_{|\boldsymbol{d}_\ell^{j,k}+\boldsymbol{r}_\ell^{j,k}|}^{|\boldsymbol{d}_\ell^{j,k}|} (\phi^*)'(s) \,\mathrm{d}s \,\mathrm{d}\boldsymbol{x} \right| \\ &\overset{(1.55d),(1.58)}{\leq} \alpha^{-1} \lim_{j,k\to\infty} \left| \int_\Omega \int_{|\boldsymbol{d}_\ell^{j,k}+\boldsymbol{r}_\ell^{j,k}|}^{|\boldsymbol{d}_\ell^{j,k}|} s \,\mathrm{d}s \,\mathrm{d}\boldsymbol{x} \right| \\ &= (2\alpha)^{-1} \lim_{j,k\to\infty} \left| \int_\Omega |\boldsymbol{d}_\ell^{j,k}|^2 - |\boldsymbol{d}_\ell^{j,k} + \boldsymbol{r}_\ell^{j,k}|^2 \,\mathrm{d}\boldsymbol{x} \right| \\ &= (2\alpha)^{-1} \lim_{j,k\to\infty} \left| \|\boldsymbol{d}_\ell^{j,k}\|^2 - \|\boldsymbol{d}_\ell^{j,k} + \boldsymbol{r}_\ell^{j,k}\|^2 \right| \overset{(1.84)}{=} 0\end{aligned}$$

where we have also used that $\boldsymbol{d}_\ell^{j,k}$ is uniformly bounded in j, k by Lemma 1.6.11 to interchange the limit and the integral in the last equality. □

The same argument holds for the linearization error component estimator.

Lemma 1.7.5 (Convergence of the linearization error estimator)**.** *The linearization component estimator* $\eta_{\mathrm{lin}}^{\ell,j,k}$ *of* (1.94) *tends to 0 as* $k \to \infty$.

Finally, we have the following result pertaining to the discretization error component estimator.

Lemma 1.7.6 (Convergence of the discretization error estimator)**.** *The discretization component estimator satisfies*

$$\lim_{j,k\to\infty} (\eta_{\mathrm{dis}}^{\ell,j,k})^2 = \lim_{j,k\to\infty} (\eta_{\mathrm{tot}}^{\ell,j,k})^2. \tag{1.95}$$

Proof. In addition to (1.94), there holds

$$(\eta_{\mathrm{dis}}^{\ell,j,k})^2 \leq (\eta_{\mathrm{tot}}^{\ell,j,k})^2 + (\eta_{\mathrm{reg}}^{\ell,j,k})^2 + (\eta_{\mathrm{lin}}^{\ell,j,k})^2.$$

Thus, Lemma 1.7.4 and Lemma 1.7.5 finish the proof. □

1.7.2 Dual norm of the residual

Considering the dual norm of the residual as an error estimator leads to a different set of estimators. These have been studied previously in a variety of contexts [53, 43, 47]. First, we consider the total estimator, which provides an upper bound on the quantity (1.42).

Lemma 1.7.7 (Upper bound on the dual norm of the residual)**.** *Let the Assumption 1.6.1 hold. Let* $\mathcal{R}$ *be defined by* (1.40) *and let* $u_\ell^{j,k} \in V_\ell^p$ *be the solution of* (1.30)*. Then there holds*

$$0 \leq \tfrac{1}{2} \underbrace{|||\mathcal{R}(u_\ell^{j,k})|||_{-1}^2}_{\text{total error } (\tilde{e}_{\mathrm{tot}}^{\ell,j,k})^2} \leq \tfrac{1}{2} \underbrace{\alpha^{-1}\|\boldsymbol{A}(\nabla u_\ell^{j,k}) + \boldsymbol{t}_\ell^{j,k}\|^2}_{\text{total est. } (\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k})^2}. \tag{1.96}$$

Proof. Let us first fix a function $\varphi \in H_0^1(\Omega)$ with $|||\varphi||| = \|\alpha^{1/2}\nabla\varphi\| = 1$. Then,

$$\begin{aligned}
\langle \mathcal{R}(u_\ell^{j,k}), \nabla\varphi\rangle &\overset{(1.40)}{=} (f,\varphi) - (\boldsymbol{A}(\nabla u_\ell^{j,k}), \nabla\varphi) \overset{(1.73)}{=} -(\boldsymbol{A}(\nabla u_\ell^{j,k}) + \boldsymbol{t}_\ell^{j,k}, \nabla\varphi) \\
&= -\alpha^{-1/2}(\boldsymbol{A}(\nabla u_\ell^{j,k}) + \boldsymbol{t}_\ell^{j,k}, \alpha^{1/2}\nabla\varphi) \leq \alpha^{-1/2}\|\boldsymbol{A}(\nabla u_\ell^{j,k}) + \boldsymbol{t}_\ell^{j,k}\|\|\alpha^{1/2}\nabla\varphi\| \\
&= \tilde{\eta}_{\mathrm{tot}}^{\ell,j,k}.
\end{aligned} \tag{1.97}$$

Since φ was arbitrary, (1.96) follows the definition (1.41). □

Corollary 1.7.8 (Decomposition of the upper bound)**.** *Let assumptions of Lemma 1.7.7 hold. Then*

$$\begin{aligned}
\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k} \leq\ & \underbrace{\alpha^{-1/2}\|\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) + \boldsymbol{t}_\ell^{j,k}\|}_{\text{discretization est. } \tilde{\eta}_{\mathrm{dis}}^{\ell,j,k}} + \underbrace{\alpha^{-1/2}\|\boldsymbol{A}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k})\|}_{\text{regularization est. } \tilde{\eta}_{\mathrm{reg}}^{\ell,j,k}} \\
& + \underbrace{\alpha^{-1/2}\|\boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|}_{\text{linearization est. } \tilde{\eta}_{\mathrm{lin}}^{\ell,j,k}}.
\end{aligned} \tag{1.98}$$

Proof. From the definition (1.96) of $\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k}$,

$$\begin{aligned}
\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k} &= \alpha^{-1/2}\|\boldsymbol{t}_\ell^{j,k} + \boldsymbol{A}(\nabla u_\ell^{j,k})\| \\
&= \alpha^{-1/2}\|\boldsymbol{t}_\ell^{j,k} + \big(\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k}) + \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k}) - \boldsymbol{A}_{\epsilon^j}(\nabla u_\ell^{j,k})\big) + \boldsymbol{A}(\nabla u_\ell^{j,k})\| \\
&\leq \tilde{\eta}_{\mathrm{dis}}^{\ell,j,k} + \tilde{\eta}_{\mathrm{reg}}^{\ell,j,k} + \tilde{\eta}_{\mathrm{lin}}^{\ell,j,k}.
\end{aligned}$$

□

Remark 1.7.9 (Decomposition of the residual)**.** *We can also split the residual* $\mathcal{R}(v)$ *introduced in* (1.40) *as*

$$\mathcal{R}(v) = \mathcal{R}_{\mathrm{dis}}^{\ell,j,k}(v) + \mathcal{R}_{\mathrm{reg}}^{\ell,j,k}(v) + \mathcal{R}_{\mathrm{lin}}^{\ell,j,k}(v), \tag{1.99}$$

where

$$\langle \mathcal{R}_{\mathrm{dis}}^{\ell,j,k}(v), w\rangle := (f, w) - (\boldsymbol{A}_{\epsilon^j}^{k-1}\nabla v, \nabla w), \tag{1.100a}$$

$$\langle \mathcal{R}_{\mathrm{reg}}^{\ell,j,k}(v), w\rangle := (\boldsymbol{A}_{\epsilon^j}(\nabla v) - \boldsymbol{A}(\nabla v), \nabla w), \tag{1.100b}$$

$$\langle \mathcal{R}_{\mathrm{lin}}^{\ell,j,k}(v), w\rangle := (\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla v) - \boldsymbol{A}_{\epsilon^j}(\nabla v), \nabla w). \tag{1.100c}$$

Then there holds

$$|||\mathcal{R}(u_\ell^{j,k})|||_{-1} \leq |||\mathcal{R}_{\mathrm{dis}}^{\ell,j,k}(u_\ell^{j,k})|||_{-1} + |||\mathcal{R}_{\mathrm{reg}}^{\ell,j,k}(u_\ell^{j,k})|||_{-1} + |||\mathcal{R}_{\mathrm{lin}}^{\ell,j,k}(u_\ell^{j,k})|||_{-1}. \tag{1.101}$$

1.7.3 Energy norm

In light of the relation between the dual norm of the residual and the energy norm presented in Proposition 1.4.1, we observe that the results of the previous section imply,

Lemma 1.7.10 (Upper bound on the energy error)**.** *Let the assumptions of Lemma 1.7.7 hold. Then,*

$$|||u_\ell^{j,k} - u||| \leq |||\mathcal{R}(u_\ell^{j,k})|||_{-1} \leq \tilde{\eta}_{\mathrm{tot}}^{\ell,j,k} \leq \tilde{\eta}_{\mathrm{dis}}^{\ell,j,k} + \tilde{\eta}_{\mathrm{reg}}^{\ell,j,k} + \tilde{\eta}_{\mathrm{lin}}^{\ell,j,k}. \tag{1.102}$$

1.8 Efficiency of the estimators

Up to this point, we have only considered the property that the estimators we construct are global upper bounds of the errors that we have defined and identified the error components. However, it has already been demonstrated that these estimators provide lower bounds to the error as well, even local lower bounds.

1.8.1 Dual norm of the residual

The setting of [53, 19] covers the present one. In particular, the flux of Definition 1.6.2, which in turn defines $\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k}$ as in (1.96), provides a local lower bound under certain conditions on stopping criteria of the form (1.105a) and (1.105b) below. Roughly speaking, the efficiency is achieved when discretization component estimator is larger than the other components. This efficiency is robust with respect to the Lipschitz/montonicity parameters L/α, i.e.,

$$\tilde{\eta}_{\mathrm{tot}}^{\ell,\bar{j},\bar{k}} \lesssim |||\mathcal{R}(u_\ell^{\bar{j},\bar{k}})|||_{-1} + \text{oscillation and quadrature terms},$$

where the hidden constant has no dependence on L and α and the oscillation and quadrature terms are discussed in §1.8.3 below.

1.8.2 Energy norm

In the case of the energy norm, we can also obtain a (local) lower bound for the estimator $\tilde{\eta}_{\mathrm{tot}}^{\ell,j,k}$ of (1.102) by using the lower bound of (1.46):

$$|||\mathcal{R}(u_\ell^{\bar{j},\bar{k}})|||_{-1} \leq \frac{L}{\alpha}|||u_\ell^{\bar{j},\bar{k}} - u|||. \tag{1.103}$$

Unfortunately, this makes appear the "strength of the nonlinearity" factor L/α. We will show numerically in §1.10.1.1, that this bound actually appears to be sharp. Thus, if L/α is large, the a posteriori estimate is pessimistic for the error measured in the energy norm.

1.8.3 Energy difference

In [68, Theorems 3.4 and 4.1], we study the efficiency of $\eta_{\mathrm{tot}}^{\ell,j,k}$ as well as of a related estimator that incorporates the error in the difference of energies for the linear minimization problem corresponding to the linearization step (1.30). In particular, results of the form

$$\mathcal{J}(u_\ell^{j,k}) - \mathcal{J}(u) \leq (\eta_{\mathrm{tot}}^{\ell,j,k})^2 \lesssim (C_\ell^k)^2(\mathcal{J}(u_\ell^{j,k}) - \mathcal{J}(u)) + (\text{oscillation and quadrature terms})^2 \tag{1.104}$$

are obtained where a typical data oscillation term is of the form $\left(\sum_{K\in\mathcal{T}_\ell}\left(\frac{h_K}{\pi}\|(I-\Pi_{\ell,p})f\|_K^2\right)\right)^{1/2}$ and a typical quadrature term is of the form $\|(\boldsymbol{I}-\boldsymbol{\Pi}_\ell^{\boldsymbol{RT}})\boldsymbol{A}_{\epsilon^j}^{k-1}(\nabla u_\ell^{j,k})\|$, where the projection operators $\Pi_{\ell,p}$ and $\boldsymbol{\Pi}_\ell^{\boldsymbol{RT}}$ are defined in §1.6. Importantly, the constant C_ℓ^k only depends locally on the ratio of the biggest and smallest eigenvalues of $\mathbb{A}_{\epsilon^j}^{k-1}$ and the hidden constant has no dependence on α and L at all.

1.9 Adaptive algorithm

In this section we will use the estimators based on the energy difference as in §1.7.1 to devise an adaptive algorithm. In particular, the algorithm will construct a sequence of solutions $u_\ell^{j,k}$ over mesh levels ℓ, regularization iterations j, and Newton iterations k. The main ideas will be to 1) spend the maximum amount of computing time on coarser meshes where computations are cheap and 2) decrease the regularization parameter adaptively so as to make Newton converge but avoid polluting the solution to the approximate problem. The algorithm accepts user-defined parameters $\gamma_{\mathrm{lin}} > 0$ and $\gamma_{\mathrm{reg}} > 0$ that express the requested relative sizes of the corresponding error components. Additionally, the algorithm takes as parameters $C_\epsilon \in (0,1)$ and $\epsilon^0 > 0$ that determine the sequence of regularization parameters according to (1.17) and a user-specified tolerance `tol`, the requested maximal overall error. We consider three stopping criteria with bars denoting the stopping indices as

$$\eta_{\mathrm{lin}}^{\ell,j,\bar{k}} < \gamma_{\mathrm{lin}}\eta_{\mathrm{reg}}^{\ell,j,\bar{k}}, \tag{1.105a}$$

$$\eta_{\mathrm{reg}}^{\ell,\bar{j},\bar{k}} < \gamma_{\mathrm{reg}}\eta_{\mathrm{dis}}^{\ell,\bar{j},\bar{k}}, \tag{1.105b}$$

$$\eta_{\mathrm{tot}}^{\bar{\ell},\bar{j},\bar{k}} < \mathtt{tol}. \tag{1.105c}$$

The first criterion (1.105a) indicates that the Newton solver should not continue on a given regularized problem if it has sufficiently converged. The problem should be changed (increasing the difficulty) by lowering the regularization parameter. The second criterion (1.105b) says that once the regularization parameter is sufficiently small on a given mesh, we can then pass to a finer mesh through the refinement procedure `REFINE` (newest vertex bisection or uniform refinement). Finally, the last criterion (1.105c) checks whether the estimator for the total error is below the user-specified threshold. Details are given in Algorithm 2.

Algorithm 2: Adaptive regularized Newton algorithm

Initialization Choose an initial guess $u_0^{0,0} \in V_0^p$ and initialize $\ell = j := 0$

:

Parameters **:** $\gamma_{\text{reg}}, \gamma_{\text{lin}}, \texttt{tol}, \epsilon^0, C_\epsilon$

Loop for discretization
  Loop for regularization
    Initialize $k := 0$
    Loop for linearization
      Increment $k := k + 1$
      From $u_\ell^{j,k-1}$ compute the linearized operator $\boldsymbol{A}_{\epsilon^j}^{k-1}$ by (1.35)
      Solve for $u_\ell^{j,k}$ in (1.30)
      Compute $\boldsymbol{t}_\ell^{j,k}$ following Definition 1.6.2 and $\eta_{\text{tot}}^{\ell,j,k}$ following (1.91)
      Compute estimators $\eta_{\text{dis}}^{\ell,j,k}, \eta_{\text{reg}}^{\ell,j,k}, \eta_{\text{lin}}^{\ell,j,k}$ following (1.94)
    until $\eta_{\text{lin}}^{\ell,j,k} < \gamma_{\text{lin}} \eta_{\text{reg}}^{\ell,j,k}$
    Update $\bar{k} := k$
    if $\eta_{\text{reg}}^{\ell,j,\bar{k}} \geq \gamma_{\text{reg}} \eta_{\text{dis}}^{\ell,j,\bar{k}}$ **then**
      Increment $j := j + 1$
      Update $\epsilon^j := C_\epsilon \epsilon^{j-1}$
    end
  until $\eta_{\text{reg}}^{\ell,j,\bar{k}} < \gamma_{\text{reg}} \eta_{\text{dis}}^{\ell,j,\bar{k}}$
  Update $\bar{j} := j$
  Increment $\ell := \ell + 1$
  $V_\ell^p := \texttt{REFINE}(V_{\ell-1}^p, \eta_{\text{tot},K}^{\ell,\bar{j},\bar{k}})$
  $u_{\ell j,0} := u_{\ell-1}^{\bar{j},\bar{k}}$
until $\eta_{\text{tot}}^{\ell,\bar{j},\bar{k}} < \texttt{tol}$
Update $\bar{\ell} := \ell$
return $u_{\bar{\ell}}^{\bar{j},\bar{k}}$

1.10 Numerical experiments

We now present numerical experiments to substantiate the theory developed in the preceding sections. In particular, we compare and contrast the three error measures discussed in §1.4 for a polynomial manufactured solution defined on the unit square. Next, we explore several solver strategies for this same polynomial solution, including the adaptive Algorithm 2. In this case we will use uniform mesh refinement. Finally, we consider an unknown solution on an L-shaped domain and test the adaptive Algorithm 2 comparing both adaptive and uniform mesh refinement, in addition to adaptivity in regularization and linearization.

We will consider the effectivity index defined as the ratio of the estimator to the error, and in particular we have, using the notation of Proposition 1.7.1, Lemma 1.7.7, and Lemma 1.7.10,

$$I_{\text{tot}}^{\ell,j,k} := \frac{\eta_{\text{tot}}^{\ell,j,k}}{e_{\text{tot}}^{\ell,j,k}}, \quad \tilde{I}_{\text{tot}}^{\ell,j,k} := \frac{\tilde{\eta}_{\text{tot}}^{\ell,j,k}}{\tilde{e}_{\text{tot}}^{\ell,j,k}}, \quad \hat{I}_{\text{tot}}^{\ell,j,k} := \frac{\tilde{\eta}_{\text{tot}}^{\ell,j,k}}{|||u_\ell^{j,k} - u|||}. \tag{1.106}$$

We also consider a relative version of the various quantities (both errors and estimators) by

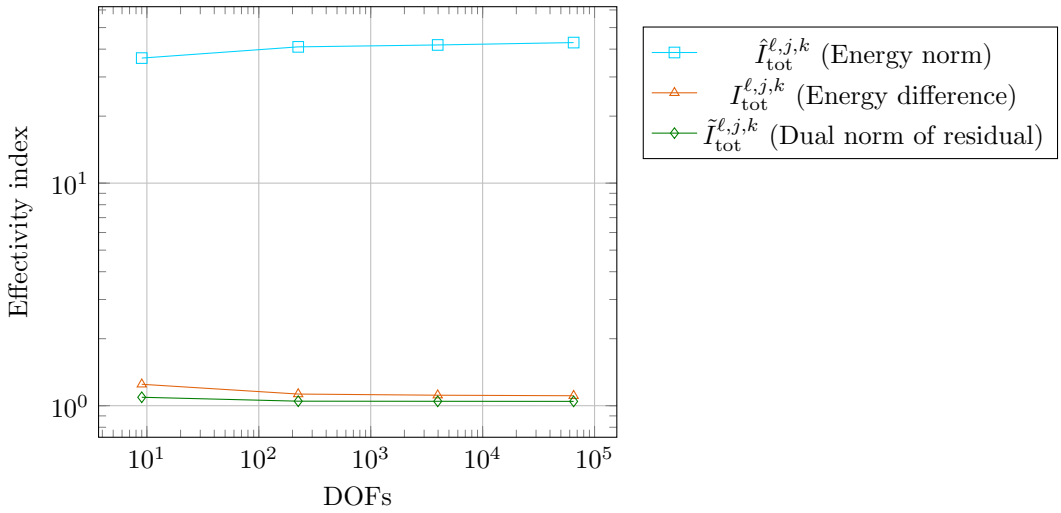


Figure 1.4: [Polynomial solution (1.108), kink nonlinearity (1.19) with $s_0 = 1$ and m =64, polynomial degree $p = 1$, #DOFs varying, no regularization $\epsilon = 0$] Robustness with respect to the number of DOFs for the three error measures.

dividing by the energy of the approximate solution, e.g.,

$$(\eta^{\ell,j,k}_{\mathrm{lin,rel}})^2 := \frac{(\eta^{\ell,j,k}_{\mathrm{lin}})^2}{\mathcal{J}(u^{j,k}_\ell)}. \tag{1.107}$$

We will start with piecewise linear continuous finite elements i.e., we first set the polynomial order $p = 1$ in (1.28), but later we test adaptivity for $2 \le p \le 5$. All numerical experiments are conducted with the help of the `Gridap.jl` library [7, 118] in the Julia programming language.

1.10.1 Polynomial solution on a square

In this case, we consider a square domain $\Omega = (0,1)^2 \subset \mathbb{R}^2$ and we take a manufactured solution,

$$u(x) = 10\,x(x-1)y(y-1) \tag{1.108}$$

to generate the source term f. We neglect that $f \notin \mathcal{P}_0(\mathcal{T}_\ell)$. We will take ϕ in the energy (1.1) as in the definition of the kink function as (1.19). Thus, according to (1.21), we have that the monotonicity constant $\alpha = 1$ and the Lipschitz continuity constant $L = m$.

1.10.1.1 Comparison of the three error measures of §1.4

In this section, we will numerically investigate the relationships between the error measures discussed in section §1.4. The results are given in Figures 1.4 and 1.5. For this example we set the regularization parameter to zero, i.e., $\epsilon^0 := 0$. We first consider the scaling of the effectivity indices (1.106) with respect to the number of DOFs. We remark that all three error measures appear to be stable under uniform mesh refinement. This is consistent with the theory since the constants in the reliability and efficiency bounds are independent of the mesh size/number of DOFs. We do, however, note that the effectivity for the energy difference is much larger for each value of the mesh than for the other two error measures.

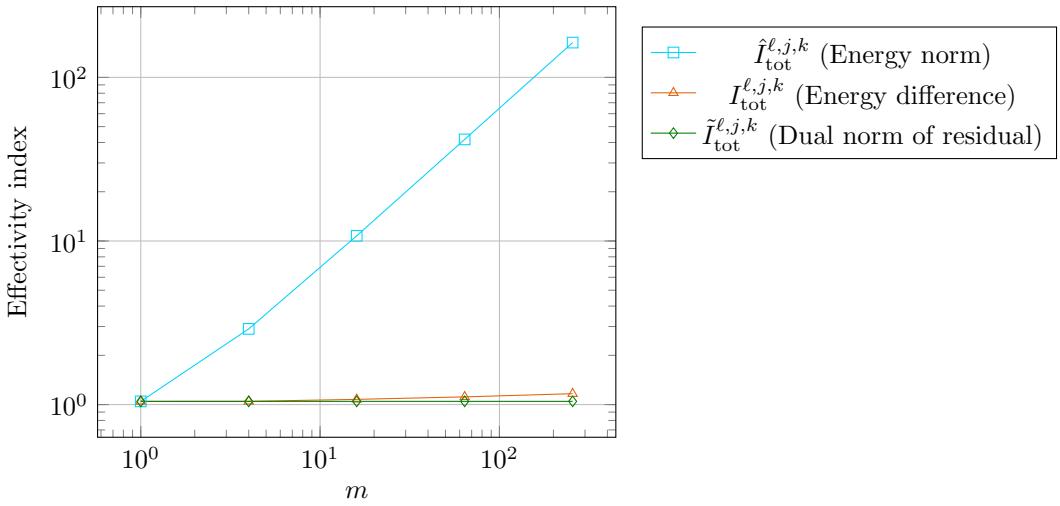


Figure 1.5: [Polynomial solution (1.108), kink nonlinearity with $s_0 = 1$ and m varying, polynomial degree $p = 1$, #DOFs=3969, no regularization $\epsilon = 0$] The effectivity associated with the energy norm scales linearly in m whereas the other effectivities remain constant.

Now we consider the scaling with respect to the "size of the nonlinearity" i.e., $m = L/\alpha$ in Figure 1.5. We begin with the dual norm of the residual. We observe that the estimator (1.96) is not only a constant-free upper bound on the dual norm of the residual, but it is also a (local) lower bound [53, 19] robust with respect to m.

Next we observe that the energy norm effectivity scales like m. This is consistent with the theoretical bounds, (1.46) and (1.103) and confirms non-robustness.

Finally, we consider the effectivity based on the difference of energies. This estimator too appears to be robust with respect to the scaling for this range of parameters. This behavior has been studied in [68]. In particular, a robustness result was demonstrated for a modified estimator, see (1.104).

1.10.1.2 Need for regularization for large ratios L/α

For the rest of this section, we will only consider the error and estimator based on the difference of energies of Proposition 1.7.1. We now study a much larger value of the "size of the nonlinearity" $m = L/\alpha$ and test the standard Newton algorithm, i.e., performing Algorithm 2 without regularization (with $\epsilon^0 = 0$). In particular, we set $m = 10000$ and we consider the same manufactured solution (1.108) on a fixed uniform mesh with 261121 DOFs. In Figure 1.6, we plot both the relative total error and estimator along with the corresponding effectivity index in function of the Newton iterations. In the lower figure, the components as in (1.94) are shown. Based on the behavior of the linearization estimator $\eta^{\ell,j,k}_{\mathrm{lin}}$, we conclude that the Newton solver fails to meet the specified convergence criteria after 50 iterations and we artificially terminate the algorithm. This manifests possible non-convergence of the Newton linearization for nonsmooth nonlinearities.

In Figure 1.7 we consider the same problem but now we fix a relatively large value for the initial regularization of $\epsilon^0 := 0.125$. We set the parameters $\gamma_{\mathrm{reg}} := 1.0\mathrm{e}16$, $\gamma_{\mathrm{lin}} := 1.0\mathrm{e}{-5}$ to ensure the algorithm will converge fully in linearization but will not perform any adaptivity in either regularization or discretization.

We first remark that the effectivity index of our a posteriori error estimator oscillates

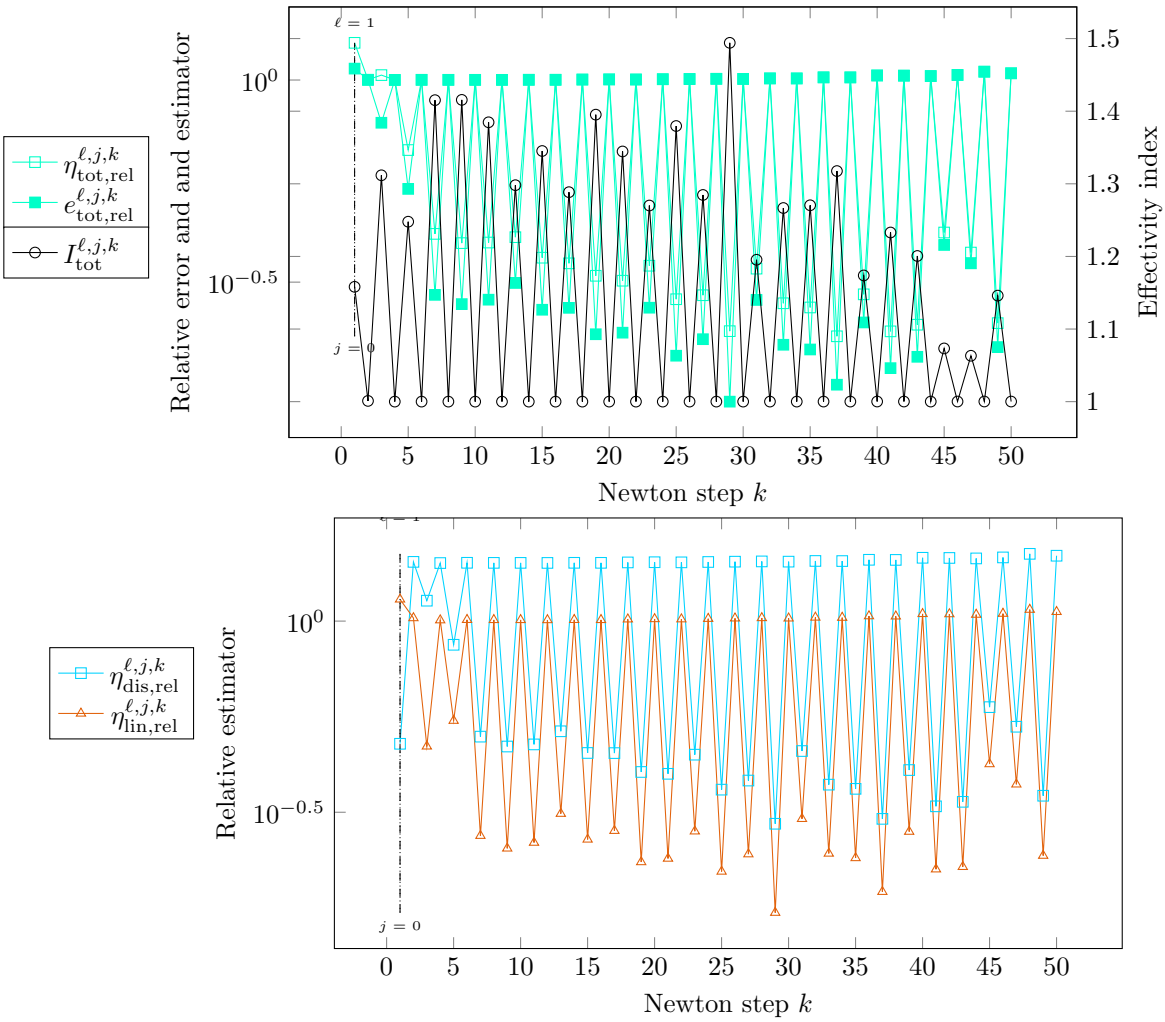


Figure 1.6: [Polynomial solution (1.108), kink nonlinearity (1.19) with $s_0 = 1$ and $m = 10000$, polynomial degree $p = 1$, #DOFs = 261121, no regularization $\epsilon^0 = 0$, refinement index ℓ, regularization index j, linearization index k,] The classical Newton method fails to converge for the unregularized problem corresponding to $\epsilon = 0$.

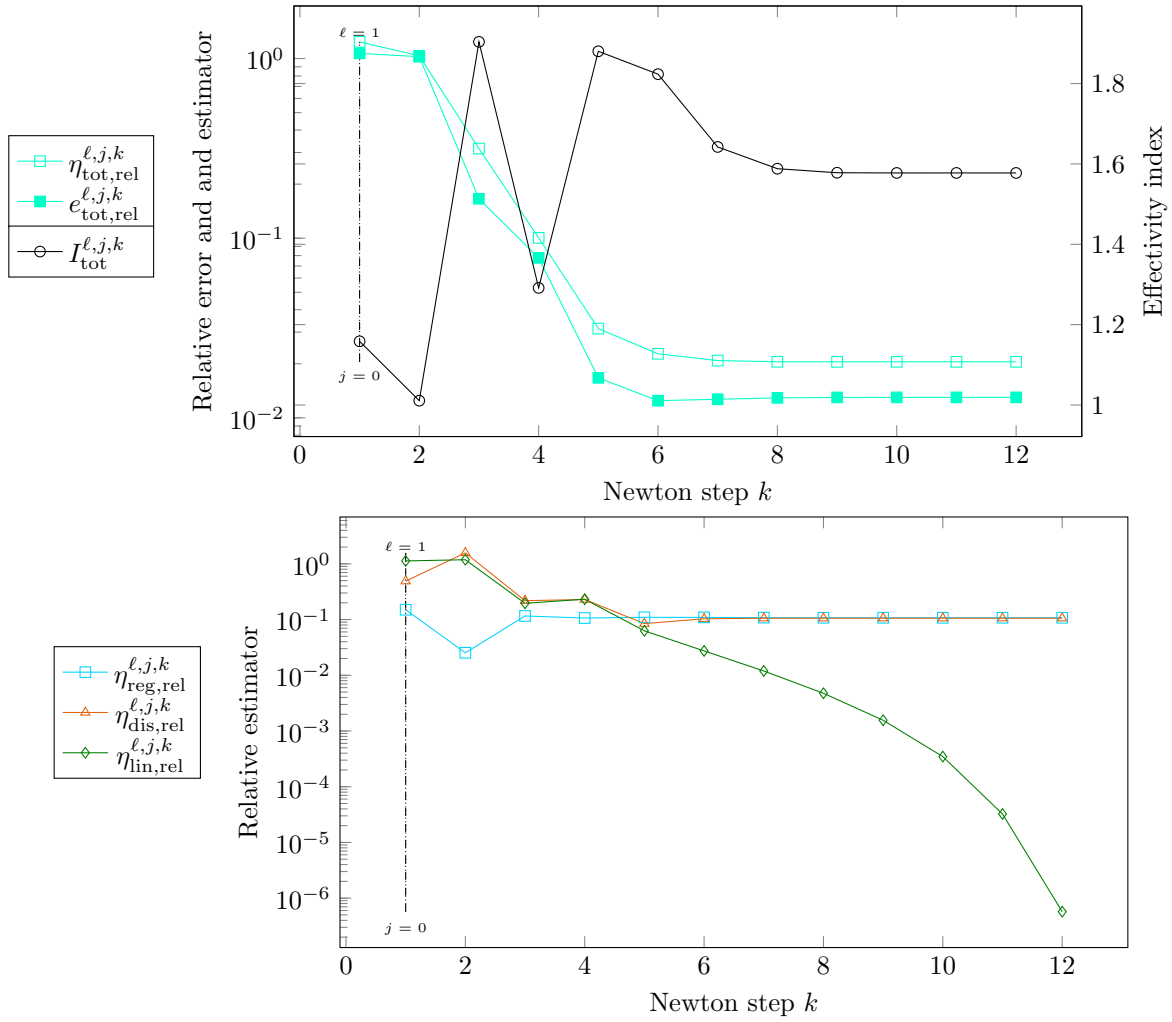


Figure 1.7: [Polynomial solution (1.108), kink nonlinearity (1.19) with $s_0 = 1$ and $m = 10000$, #DOFs = 261121, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 1.0\mathrm{e}16$, $\gamma_{\mathrm{lin}} = 1.0\mathrm{e}\text{-}5$, $C_\epsilon = 1$, polynomial degree $p = 1$, mesh refinement index ℓ, regularization index j, linearization index k] The classical Newton method converges for the regularized problem with $\epsilon = 0.125$ but the errors and estimates other than the linearization stagnate due to the fixed regularization and discretization.

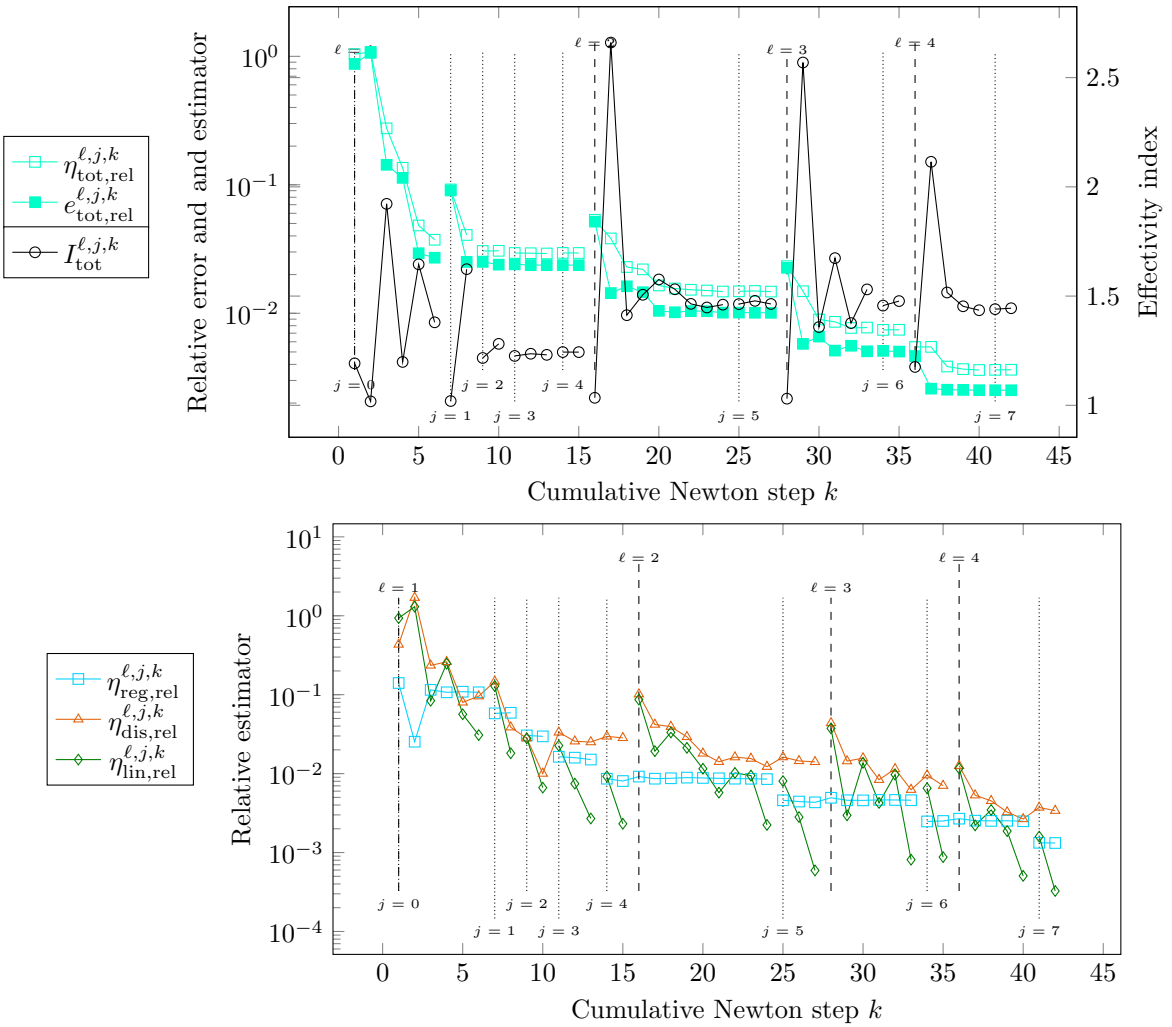


Figure 1.8: [Polynomial solution (1.108), kink nonlinearity (1.19) with $s_0 = 1$ and $m = 10000$, final #DOFs = 261121, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.6$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$, polynomial degree $p = 1$, mesh refinement index ℓ, regularization index j, linearization index k] The adaptive Algorithm 2 applied to a polynomial solution. The final value of the regularization parameter is $\epsilon^6 = 1.95\text{e-}3$.

much less, compared to Figure 1.6, signifying a more stable approximation of the error. It appears further to converge after several iterations to a value near 1.6. We next remark that the Newton linearization exhibits the optimal quadratic convergence according to the values of the linearization estimator $\eta^{\ell,j,k}_{\mathrm{lin}}$. However, the other two estimator components $\eta^{\ell,j,k}_{\mathrm{reg}}$ and $\eta^{\ell,j,k}_{\mathrm{dis}}$ stagnate at similar values: 1.06e-1 and 1.07e-1 respectively. We remark that the reason that these are larger than the total estimator $\eta^{\ell,j,k}_{\mathrm{tot}} = 2.05\text{e-}2$ is due to the insertion of absolute values in the definition (1.94). In any case, the Newton linearization now converges, but the regularization component is much too large to be satisfactory. This motivates the adaptive Algorithm 2 where the regularization estimator is decreased adaptively along the iterations.

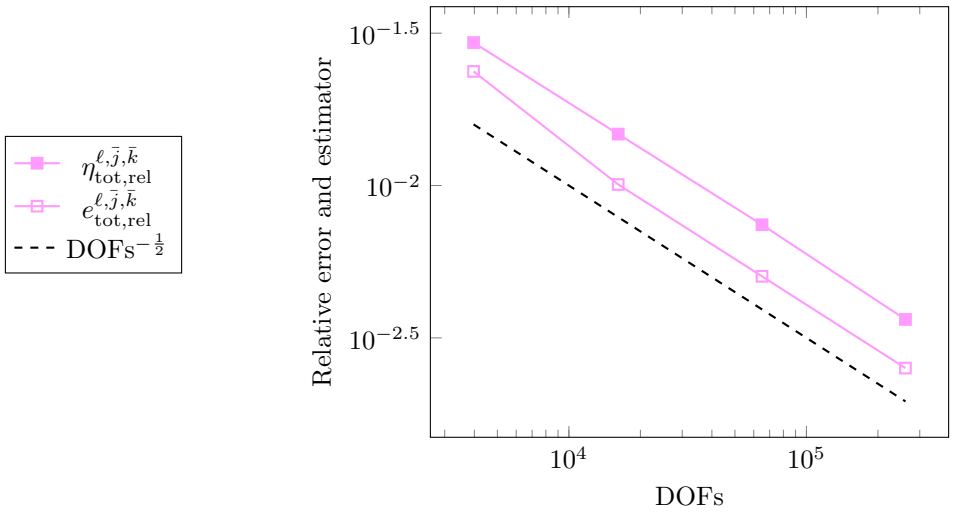


Figure 1.9: [Polynomial solution (1.108), kink nonlinearity (1.19) with $s_0 = 1$ and $m = 10000$, polynomial degree $p = 1$, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.6$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$, #DOFs varies] Achieving the optimal rate of convergence for a polynomial solution with uniform mesh refinement.

1.10.1.3 Adaptive regularization and linearization

We now engage the adaptive regularization and linearization of Algorithm 2 by setting the parameters $\gamma_{\mathrm{reg}} = 0.6$, $\gamma_{\mathrm{lin}} = 0.4$ and $C_\epsilon = 0.5$. We again set $m = 10000$, $\epsilon^0 = 0.125$, and we now start from a uniform coarse mesh with 3969 DOFs for $\ell = 0$ ($64 \times 64 \times 2$ triangles). The results of the adaptive algorithm are presented in Figure 1.8. We first remark that the effectivity index (1.106) stays bounded below 2, and as the Newton solver converges for a fixed (j, ℓ), the effectivity approaches a value near 1.4. Next, we observe that in accordance with the criteria (1.105b), the regularization component estimator is always γ_{reg}-times below the discretization component estimator. Thus, we can guarantee that in this sense the regularization does not pollute the overall error, in constrast to the previous section. Indeed, this is substantiated in Figure 1.9, where the optimal rate of convergence of both the error and the estimator with respect to DOFs is observed, for the stopping indices $\bar{k}$ and $\bar{j}$ satisfying respectively (1.105a) and (1.105b). Finally, we see that the majority of the iterations are spent on the meshes $\ell = 0, 1, 2$. This is another advantage of the adaptive algorithm, where the coarser meshes serve as a way to create a good initial guess for the next mesh. For a smooth problem it is not strictly necessary to begin on a coarse mesh, since the refinement procedure is known a priori, but as we will see in the following section, sometimes starting on a coarse mesh is not just useful to create a sequence of initial guesses, but also to efficiently obtain an optimal mesh family using adaptive mesh refinement. Finally, we remark that the final total error is 2.51e-3 as opposed to 1.03e-2 in the previous case, where no adaptivity was used. This confirms that fixing once and for all the regularization parameter can deteriorate the quality of the final solution, which does not happen for Algorithm 2.

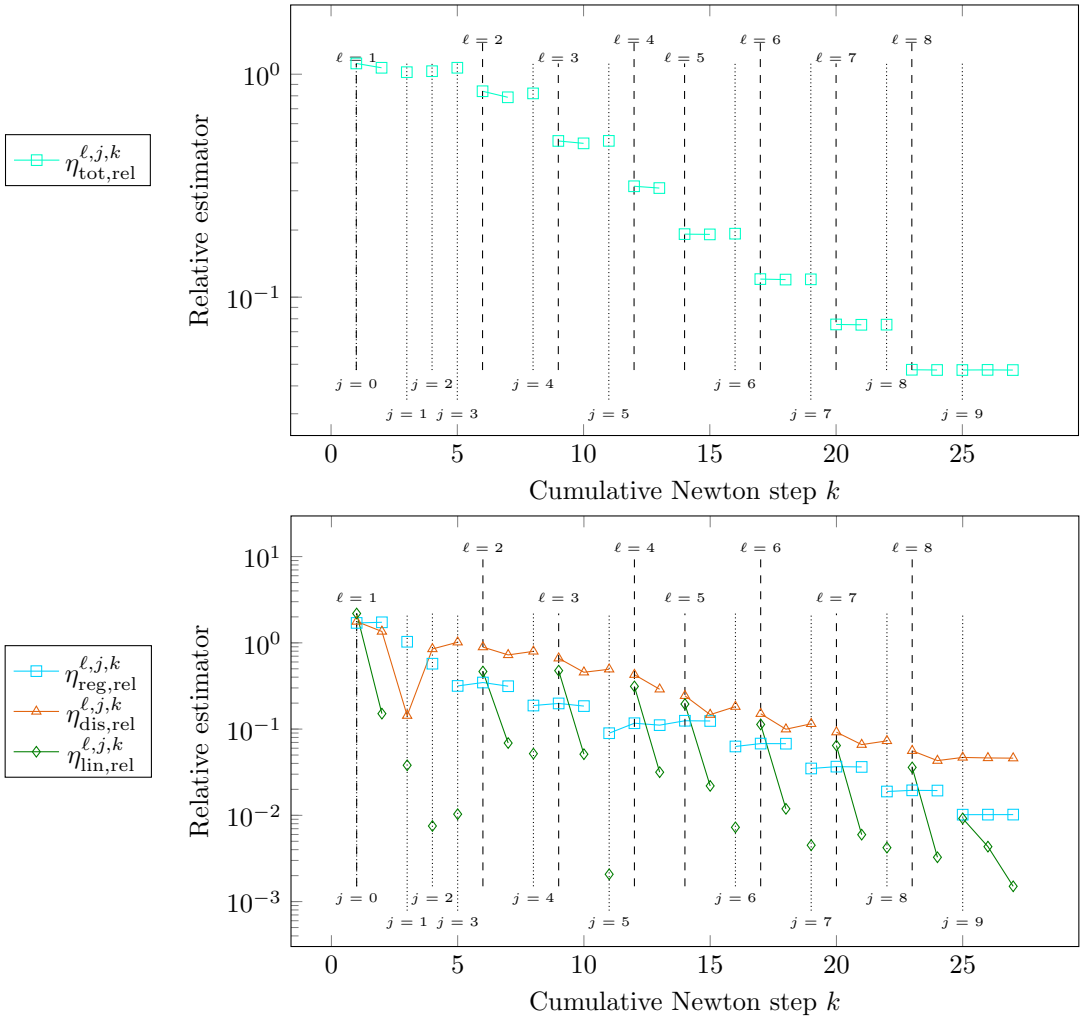


Figure 1.10: [Unknown singular solution with data (1.109), kink nonlinearity (1.19) with $s_0 = 0.75$ and $m = 10000$, final #DOFs = 97793, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.4$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$, polynomial degree $p = 1$, mesh refinement index ℓ, regularization index j, linearization index k] Applying Algorithm 2 using uniform refinement.

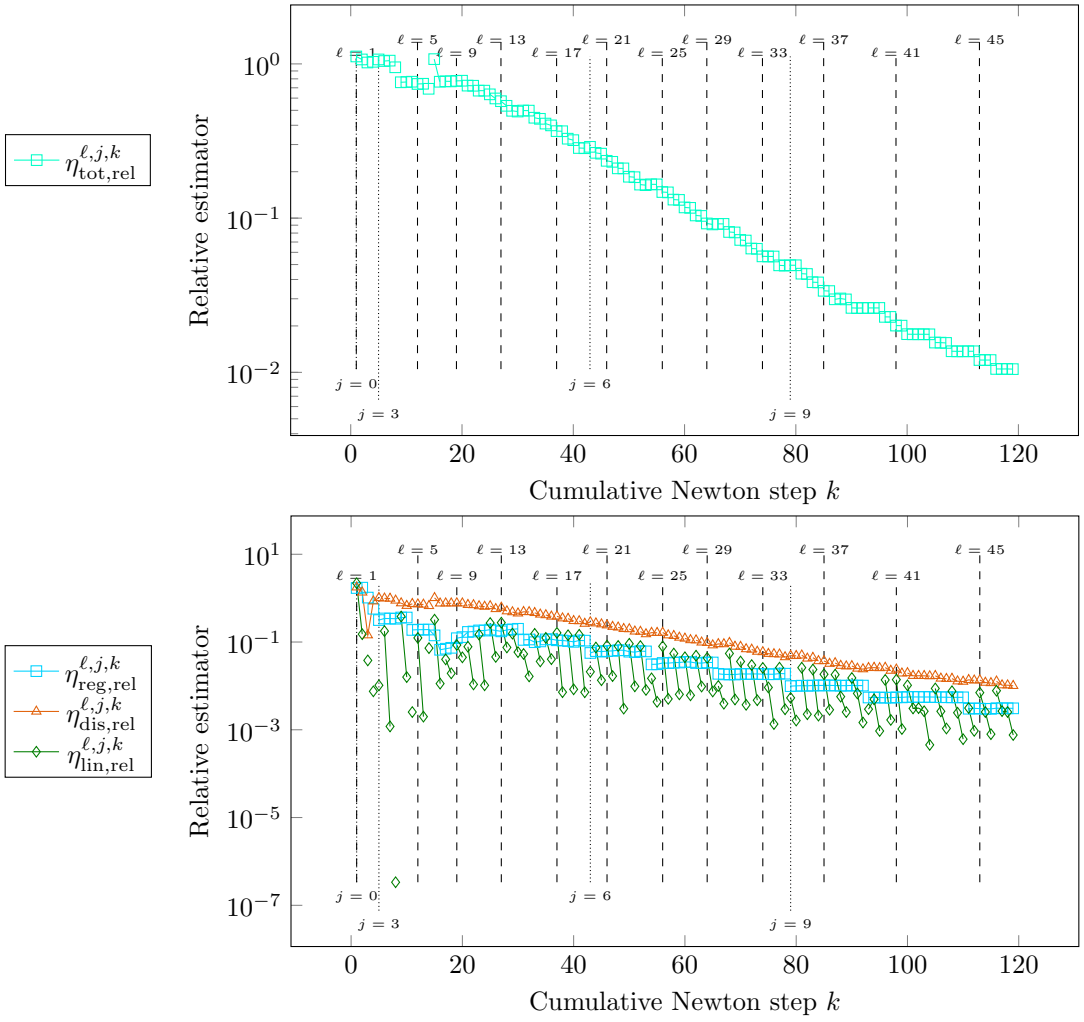


Figure 1.11: [Unknown singular solution with data (1.109), kink nonlinearity (1.19) with $s_0 = 0.75$ and $m = 10000$, final #DOFs = 86973, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.4$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$, polynomial degree $p = 1$, mesh refinement index ℓ, regularization index j, linearization index k] Applying Algorithm 2 using adaptive refinement.

1.10.2 Unknown solution on an L-shaped domain

We now consider an L-shaped domain $\Omega = (-1,1)^2 \setminus ([0,1] \times [-1,0])$ where we impose the boundary condition and right-hand side as

$$u = u_{\mathrm{D}}(r,\theta) := r^{\alpha}\sin(\alpha\theta) \text{ on } \partial\Omega \text{ and } f = 0 \text{ in } \Omega, \tag{1.109}$$

where $\alpha = \frac{2}{3}$. We note that the trace of this function is not a piecewise polynomial at the boundary, but we ignore the error due to interpolation of the boundary condition in this case. This has been rigorously studied in, e.g., [45]. We will consider the parameters $\gamma_{\mathrm{reg}} = 0.4$, $\gamma_{\mathrm{lin}} = 0.4$ and $C_\epsilon = 0.5$. The main difference compared to the polynomial solution of §1.10.1 is that we anticipate that uniform mesh refinement will not achieve the a priori optimal convergence rate due to the re-entrant corner and the nonlinearity that will activate around the curve $|\nabla u| = s_0$. We will first consider uniform refinement of the mesh as in the previous problem, and then adaptive refinement using the estimator $\eta_{\mathrm{tot},K}^{\ell,j,k}$ in (1.92) will be employed on line 19 of Algorithm 2. Note that here, since we do not know the true solution, we do not plot the error and the effectivity index.

In Figure 1.10, we consider the uniform mesh refinement strategy. We see the estimator along with the component estimators. The adaptive algorithm works as before for the smooth case, with the regularization estimator always below the discretization estimator, and for fixed (j,ℓ) the Newton solver converges very quickly. However, if we now consider convergence with respect to DOFs in Figure 1.12, we see that we obtain the suboptimal convergence rate of $\mathrm{DOFs}^{-1/3}$. This is evidence that the true solution is not H^2 regular, and therefore the optimal rate of $\mathrm{DOFs}^{-1/2}$ will not be achieved for uniform mesh refinement. Next, we consider in Figure 1.11 applying Algorithm2 but now with adaptive mesh refinement using Dörfler marking [46]. The elementwise indicators are given by $\eta_{\mathrm{tot},K}^{\ell,j,k}$ from (1.92) and we use the newest vertex bisection algorithm to enforce mesh conformity, i.e., to ensure no hanging nodes are generated. We see first of all that many more iterations are needed to obtain a similar number of final DOFs. However, the total estimator at the end of the iterations is much lower compared to that of the uniform case. This is even more explicit when we plot the number of DOFs versus the total estimator in Figure 1.12. In particular, we see that upon running the same algorithm with adaptive mesh refinement, we recover the optimal rate of $\mathrm{DOFs}^{-1/2}$ for the estimator with respect to DOFs.

One advantage of adaptive mesh refinement is that it allows in general to recover the optimal rate of convergence for arbitrary polynomial degree with respect to DOFs, i.e., $\mathrm{DOFs}^{-p/d}$, even when the solution does not have sufficient regularity for the a priori theory, see e.g. [30]. We now test the convergence rate and the behavior of the adaptive algorithm for higher polynomial degrees. We first show the convergence plots for $2 \leq p \leq 5$ in Figure 1.13. We observe that for $p = 2, 3$ the optimal rate of convergence $\mathrm{DOFs}^{-p/d}$ is again achieved for the adaptive mesh refinement. However, for $p \geq 4$, the convergence rate is suboptimal, it appears to be similar to that of $p = 3$, i.e., $\mathrm{DOFs}^{-3/2}$. We can potentially explain this deterioration by the appearance of one-dimensional curve singularities, see e.g., [29]. Indeed, in Figure 1.14, we notice there is non-trivial refinement along the curves where the norm of the gradient equals s_0. Since the right hand side $f = 0$ and we have chosen a nonsmooth nonlinearity, heuristically the solution must also be nonsmooth along $|\nabla u| = s_0$ to compensate. Finally, we consider comparing the estimator against a notion of cost across

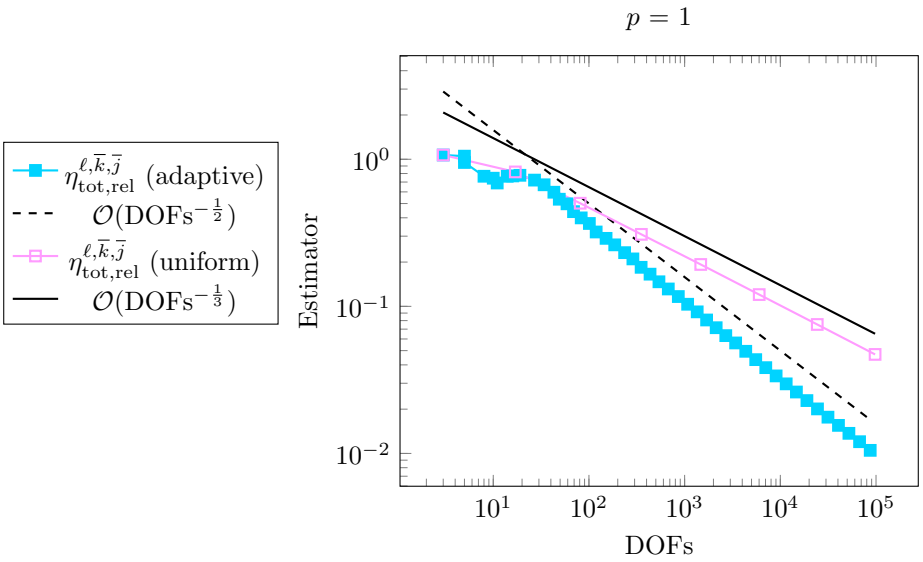


Figure 1.12: [Unknown singular solution with data (1.109), kink nonlinearity (1.19) with $s_0 = 0.75$ and $m = 10000$, polynomial degree $p = 1$, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.4$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$, #DOFs varies] Comparison of the suboptimal convergence for uniform refinement and optimal convergence for adaptive mesh refinement based on the estimator $\eta^{\ell,j,k}_{\mathrm{tot}}$ for the lowest order.

all iterations following [59, 64] We define this cost at each step of the algorithm as

$$\mathrm{Cost} = \sum_{\ell=0}^{\bar{\ell}} \sum_{j=0}^{\bar{j}(\ell)} \sum_{k=1}^{\bar{k}(\ell,j)} (\mathrm{DOFs})_\ell. \tag{1.110}$$

We observe in Figure 1.15 that the rates in this metric are very similar to those observed for the convergence with respect to DOFs of Figure 1.12; in particular, we shall obtain the optimal -1/2 rate in cost for adaptive mesh refinement.

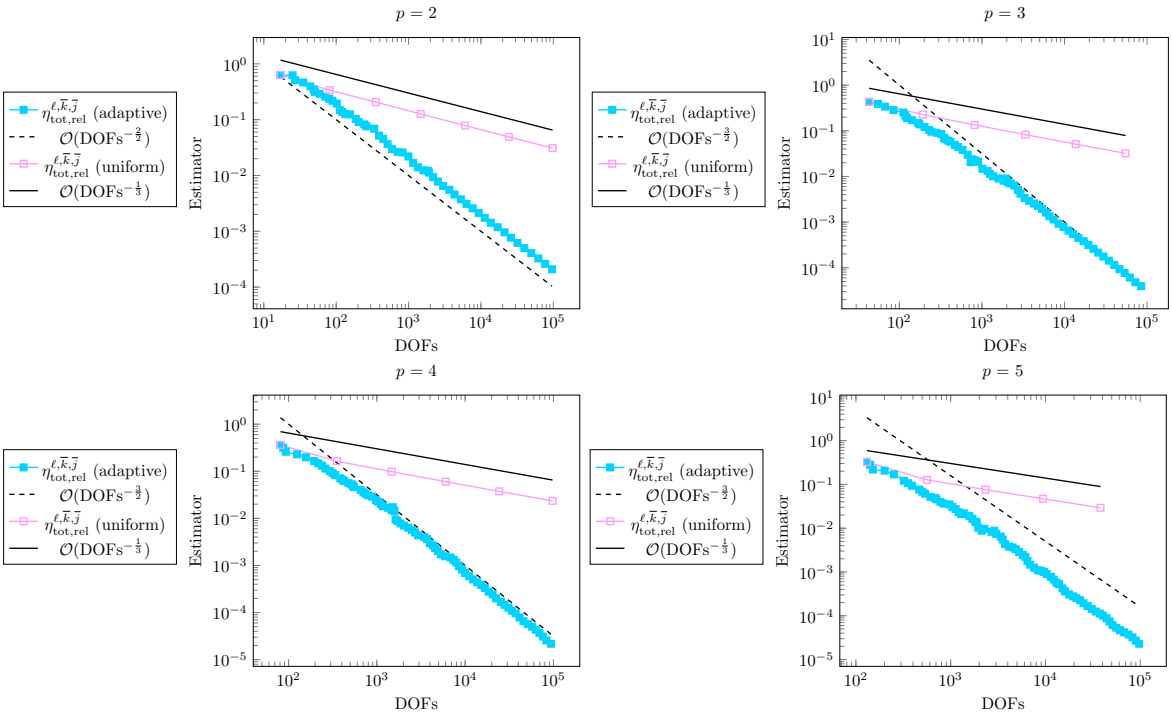


Figure 1.13: [Unknown singular solution with data (1.109), kink nonlinearity (1.19) with $s_0 = 0.75$ and $m = 1000$, different polynomial degrees p, #DOFs varying, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.4$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$] Different polynomial degrees p using adaptive mesh refinement based on $\eta_{\mathrm{tot}}^{\ell,j,k}$. The optimal rate of $\mathrm{DOFs}^{-p/d}$ is obtained for adaptive refinement up to $p = 3$, then only suboptimal convergence of $\mathrm{DOFs}^{-3/d}$ is achieved. We explain this by the appearance of one-dimensional (curve) singularities. In the case of uniform mesh refinement, increasing the polynomial degree does not change the suboptimal rate of $\mathrm{DOFs}^{-1/3}$.

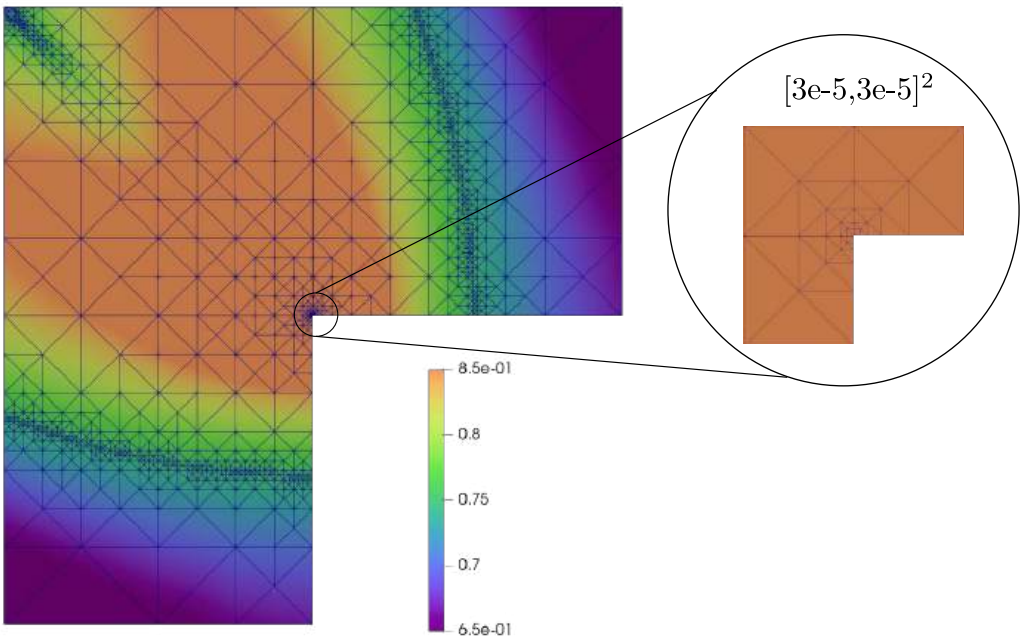


Figure 1.14: [Unknown singular solution with data (1.109), kink nonlinearity (1.19) with $s_0 = 0.75$ and $m = 1000$, polynomial degree $p = 4$, #DOFs=93681, $\epsilon^0 = 0.125$, $\gamma_{\text{reg}} = 0.4$, $\gamma_{\text{lin}} = 0.4$, $C_\epsilon = 0.5$,] Coloring corresponding to the norm of the gradient of the approximate solution at the final iteration, i.e., $|\nabla u_\ell^{\bar{j},\bar{k}}|$. We note the aggressive refinement at the re-entrant corner, and the weaker, but still substantial, refinement along the curves corresponding to $s_0 = |\nabla u_\ell^{\bar{j},\bar{k}}|$, i.e., at the kink.

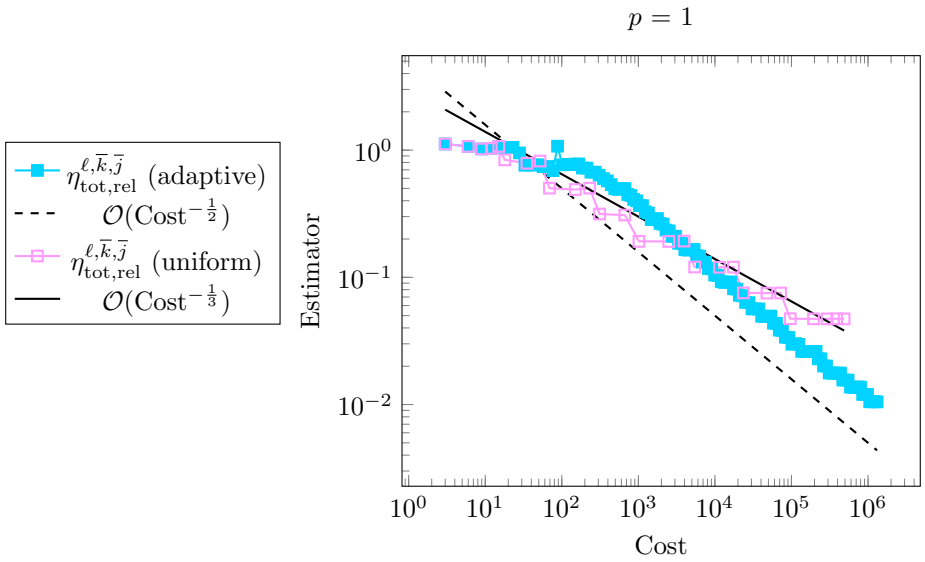


Figure 1.15: [Unknown singular solution with data (1.109), kink nonlinearity (1.19) with $s_0 = 0.75$ and $m = 10000$, polynomial degree $p = 1$, $\epsilon^0 = 0.125$, $\gamma_{\mathrm{reg}} = 0.4$, $\gamma_{\mathrm{lin}} = 0.4$, $C_\epsilon = 0.5$, #DOFs varies] Convergence in costs given by the triple sum (1.110). We observe the same rates in function of the cost as we do for the DOFs which is expected theoretically.

1.11 Conclusions and future work

In this paper, we have considered an adaptive algorithm to iteratively solve energy minimization problems with nonsmooth nonlinearities. Our adaptive algorithm is guided by the so-called primal-dual gap error estimator which provides an upper bound for the difference of energies. We construct the necessary dual object required by the estimator by solving mutually independent, patch-local, minimization problems. We introduce a regularization to allow the use of a standard Newton's method as a nonlinear solver for the nonsmooth system of equations associated to the minimization problem. The algorithm adaptively controls the regularization parameter to reduce the model error incurred by regularizing the problem. We perform a decomposition of the total estimator into component estimators related to regularization, discretization, and linearization. In particular, we prove that these component estimators converge to zero in the limit as the number of associated iterations tends to infinity. These component estimators are used to construct stopping criteria for the various components of the algorithm.

We test our algorithm numerically on two examples. In the first example, we show that the regularization restores the (quadratic) convergence of Newton's method, which without regularization failed to converge. Moreover, the adaptivity in the regularization does not influence the optimal rate of convergence of the error in the energy difference with respect to DOFs. In the second example, we consider an unknown solution on an L-shaped domain. We use adaptive mesh refinement to overcome the geometric singularity generated by the re-entrant corner; there also appears a singularity along a curve arising from the nonsmooth nonlinearity. With the help of adaptive mesh refinement, we again obtain the optimal rate of convergence with respect to DOFs for low order cases. However, for higher orders, suboptimal convergence rates are obtained. We attribute this to the appearance of the above-discussed singularity, which is a well known difficulty for isotropic mesh refinement.

In terms of future work, one possible approach to address the singularity problem in the L-shaped domain case would be to employ an anisotropic refinement strategy. In our work we, however, would be missing a number of theoretical tools. It would also be instructive to prove convergence of the adaptive algorithm, i.e., to show rigorously that by decreasing the regularization, we can obtain the optimal rate of convergence with respect to DOFs and cost, which is what we observe numerically. It may also be possible to extend certain aspects of this algorithm to other energy minimization settings posed in different spaces like the p-Laplace problem or the obstacle problem.

1.A Proofs from §1.5

Proof of Proposition 1.5.1. We begin by proving (1.55a). To simplify notation, define $\xi := \phi'$. We consider the integral in (1.50), with the change of variables $s = \phi'(t) = \xi(t)$

$$\begin{aligned}\int_0^r \xi^{-1}(s)\,\mathrm{d}s &= \int_{\xi^{-1}(0)}^{\xi^{-1}(r)} \xi^{-1}(\xi(t))\xi'(t)\,\mathrm{d}t = \int_0^{\xi^{-1}(r)} t\xi'(t)\,\mathrm{d}t \\ &= t\xi(t)\Big|_0^{\xi^{-1}(r)} - \int_0^{\xi^{-1}(r)} \xi(t)\,\mathrm{d}t = r\xi^{-1}(r) - \phi(\xi^{-1}(r)) + \phi(0),\end{aligned}$$

where we have used our assumptions that $\phi'(0) = 0$ and hence also $(\phi')^{-1}(0) = 0 = \xi^{-1}(0)$. The second equality in (1.55a) follows from the basic fact that for a convex differentiable function, the max is obtained by setting the derivative w.r.t. s in the curly braces equal to zero, and hence $r = \phi'(s)$.

To prove (1.55b), we consider the criterion for convexity. For $r_1, r_2 \in \mathrm{Dom}(\phi^*)$ and $\alpha \in [0,1]$,

$$\begin{aligned}\phi^*(\alpha r_1 + (1-\alpha)r_2) &= \max_s\{s[\alpha r_1 + (1-\alpha)r_2] - \phi(s)\} \\ &= \max_s\{\alpha[sr_1 - \phi(s)] + (1-\alpha)[sr_2 - \phi(s)]\} \\ &\le \alpha \max_s\{sr_1 - \phi(s)\} + (1-\alpha)\max_s\{sr_2 - \phi(s)\} \\ &= \alpha\phi^*(r_1) + (1-\alpha)\phi^*(r_2).\end{aligned}$$

Finally, to prove (1.55c), let now $\zeta(r) := (\phi')^{-1}(r)$, so that

$$\frac{d}{dr}\phi^*(r) = \frac{d}{dr}\big(r\zeta(r) - \phi(\zeta(r))\big) = \zeta(r) + r\zeta'(r) - \phi'(\zeta(r))\zeta'(r) = \zeta(r) = (\phi')^{-1}(r),$$

because $\phi'(\zeta(r)) = r$ by definition, whereas (1.55d) is obvious. □

Proof of Corollary 1.5.2. The inequality (1.56), follows immediately from the max definition of the transform, i.e., the second equality of (1.55a). The maximum in (1.55a) for $r = \phi'(s)$, as discussed above, which leads to the equality. □

Proof of Corollary 1.5.3. Since we know $\phi' : \mathbb{R}^+ \to \mathbb{R}^+$, we have

$$\boldsymbol{A}(\boldsymbol{q}) \cdot \boldsymbol{q} = \frac{\phi'(|\boldsymbol{q}|)}{|\boldsymbol{q}|}\boldsymbol{q} \cdot \boldsymbol{q} = \underbrace{|\boldsymbol{A}(\boldsymbol{q})|}_{r}\underbrace{|\boldsymbol{q}|}_{s}. \tag{1.111}$$

Now take r in s as in the Young inequality (1.56), and note that $r = \phi'(s)$ so equality (1.57b) holds. Unpacking the definitions, we find (1.57a) as

$$\boldsymbol{A}(\boldsymbol{A}^*(\boldsymbol{q})) \overset{(1.51)}{=} \frac{\phi'\big(|\boldsymbol{A}^*(\boldsymbol{q})|\big)}{|\boldsymbol{A}^*(\boldsymbol{q})|}\boldsymbol{A}^*(\boldsymbol{q}) = \phi'\big((\phi^*)'(|\boldsymbol{q}|)\big)\frac{\boldsymbol{A}^*(\boldsymbol{q})}{(\phi^*)'(|\boldsymbol{q}|)} \overset{(1.55c)}{=} \frac{|\boldsymbol{q}|\boldsymbol{A}^*(\boldsymbol{q})}{(\phi^*)'(|\boldsymbol{q}|)} \overset{(1.51)}{=} \boldsymbol{q}.$$

□

Proof of Lemma 1.5.4. Note that for any $x, y \geq 0$,

$$\begin{aligned} 0 \leq \alpha(x-y)^2 &\overset{(1.3b)}{\leq} (x-y)(\phi'(x)-\phi'(y)) \\ \implies \alpha|x-y|^2 &\leq |x-y||\phi'(x)-\phi'(y)| \\ \implies \alpha|x-y| &\leq |\phi'(x)-\phi'(y)|. \end{aligned}$$

Thus, since we assume ϕ' is bijective on $\mathbb{R}^+$, we may take $x = (\phi')^{-1}(r), y = (\phi')^{-1}(s)$, yielding

$$\alpha|(\phi')^{-1}(r) - (\phi')^{-1}(s)| \leq |r-s|.$$

The relationship given by (1.55c) finishes the proof. □

Chapter 2

Robust Energy a Posteriori Estimates For Nonlinear Elliptic Problems

This chapter corresponds to the paper https://inria.hal.science/hal-04033438v2 currently submitted for publication.

2.1 Introduction

Nonlinear elliptic problems are of paramount importance in a broad range of domains such as physics, mechanics, economics, biology, and medicine, see, e.g., [10, 55, 72, 107, 117]. Numerical discretization methods then serve to deliver approximate solutions, upon employing iterative linearizations to resolve the arising discrete nonlinear systems, see, e.g., [20, 42, 49, 67, 80, 129] and the references therein.

Given a numerical approximation, there arises the important question of the error with respect to the exact solution. This is practically handled by the so-called a posteriori estimates. For nonlinear problems, these have been proposed, amongst others, in [11, 13, 32, 35, 47, 53, 63, 66, 67, 74, 81, 83, 95, 107, 129]. In particular, already in, e.g., [129], the concept of a fully computable upper bound on the energy difference while relying on a duality gap has been discussed, see also [11, 13, 35, 63, 95, 107] and the references therein. The crucial question in this context is how to locally construct a suitable equilibrated flux. This has been a subject of research for several decades [84, 41, 85] and has only reached maturity recently [81, 21, 36, 53]. One step further, the estimates can be used to adaptively steer the numerical approximation, and recently, convergence and optimality results have been obtained in [18, 28, 59, 60, 64, 69, 70], see also the references therein.

Two crucial properties of an a posteriori estimate are the efficiency, assessing whether the estimate is not only an upper bound on the error, but also, up to a generic constant, a lower bound, and its robustness, assessing whether the quality of the estimate is independent of the parameters. In the present setting, we specifically use the term robustness if the chosen error measure and the associated estimate are uniformly equivalent, for any strength of the nonlinearity. Namely, the efficiency constant has to be indeed generic, independent of the strength of the nonlinearity, leading to the same overestimation factor (effectivity index) for linear, mildly nonlinear, and highly nonlinear problems. Unfortunately, robustness is typically (theoretically) not achieved; we illustrate this in Figure 2.1. There, we present

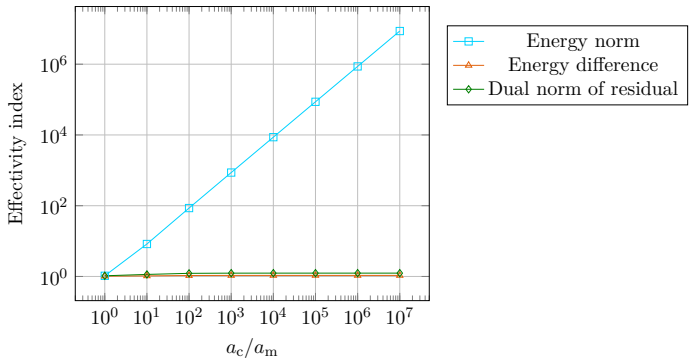


Figure 2.1: [Exponential nonlinearity (2.53), smooth solution (2.51), Newton solver, 25 DOFs] Comparison of the effectivity index (given as the ratio of the estimate over the error) of different error measures and associated a posteriori estimates.

the effectivity indices for three common error measures: the energy norm (L^2 norm of the difference of the weak gradients), the (square root of the) difference of the energies, and the dual norm of the residual (cf., e.g., [58] for their mutual comparisons). We employ guaranteed equilibrated flux estimates following [21, 53], for the problem of Example 2.6.2 below. We can observe that the estimate in the energy norm setting is not robust with respect to the strength of the nonlinearity (the effectivity index explodes as the ratio a_c/a_m from (2.4) below grows). The dual norm of the residual leads to robustness, as proven in [47, 53]. Though the dual norm of the residual is indeed localizable, cf. [19] and the references therein, it may be criticized as it actually does not take into account the nonlinearity (an incorporation has recently been addressed in [93]). The energy difference then numerically shows a robustness, though, to the best of our knowledge, all known theoretical estimates, cf. the references above, depend unfavorably on the ratio a_c/a_m. Our main motivation in this context is to bring a theoretical insight to the robustness in the energy difference setting.

We focus on nonlinear elliptic problems of the form: find $u : \Omega \to \mathbb{R}$ such that

$$-\boldsymbol{\nabla}\cdot(a(\cdot,|\boldsymbol{\nabla}u|)\boldsymbol{\nabla}u) = f \quad \text{in } \Omega, \tag{2.1a}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{2.1b}$$

where $a : \Omega \times [0,\infty) \to (0,\infty)$ is a nonlinear function satisfying assumptions of Lipschitz continuity and strong monotonicity (cf. (2.4) below). We employ a finite element approximation of (2.1) and an iterative linearization, yielding the approximation u_ℓ^k on each mesh $\mathcal{T}_\ell$ and linearization step k. The iterative linearization method needs to satisfy a few clearly identified assumptions. We will show that this is satisfied for usual linearizations such as Picard, Newton, or Zarantonello.

We consider the energy difference $\mathcal{E}_{\mathrm{N},\ell}^k$ of the nonlinear problem (cf. (2.29a)) and its a posteriori estimator $\eta_{\mathrm{N},\ell}^k$ (cf. (2.29b)). We establish that $\mathcal{E}_{\mathrm{N},\ell}^k$ and $\eta_{\mathrm{N},\ell}^k$ are respectively equivalent to L^2 norms of differences of the exact and approximate solutions with pointwise contributions of the nonlinearity (cf. Lemma 2.4.1). We then obtain our first main result, Theorem 2.4.4, which can be summarized as follows. For every iterative linearization index k, neglecting data oscillation, quadrature-type errors, and iterative linearization error terms, we have

$$\mathcal{E}_{\mathrm{N},\ell}^k \le \eta_{\mathrm{N},\ell}^k \lesssim C_\ell^k \mathcal{E}_{\mathrm{N},\ell}^k, \tag{2.2}$$

where the hidden constant depends only on the space dimension, the mesh shape-regularity,

and possibly on the polynomial degree p of the finite element approximation when the spatial dimension is greater than or equal to 4. Here, C_ℓ^k only has a local, through unfortunately not computable, dependence on the nonlinearity; we prove that in any case, $C_\ell^k \le (a_c/a_m)^{1/2}$.

In order to improve the above result and in particular the constant C_ℓ^k, we additionally consider a linearized energy difference $\mathcal{E}_{L,\ell}^k$ (cf. (2.39a)) and the associated estimator $\eta_{L,\ell}^k$ (cf. (2.39b)). We then augment $\mathcal{E}_{N,\ell}^k$ by $\mathcal{E}_{L,\ell}^k$ to form $\mathcal{E}_\ell^k$ and similarly for the estimators. We then obtain our second main result, Theorem 2.5.5, which can be presented as follows. For every iterative linearization index k, neglecting again data oscillation, quadrature-type errors, and iterative linearization error terms, we have

$$\mathcal{E}_\ell^k \le \eta_\ell^k \lesssim \widehat{C}_\ell^k \mathcal{E}_\ell^k, \tag{2.3}$$

with the same dependence as in (2.2) for the hidden constant. Here $\widehat{C}_\ell^k$ only depends on local variations of the linearization matrices and, crucially, is fully computable. Moreover, we show that $\widehat{C}_\ell^k = 1$ in the case of the Zarantonello linearization, making the estimate (2.3) robust with respect to the strength of the nonlinearity. For the other linearizations, the estimate (2.3) is robust if the computed constant $\widehat{C}_\ell^k$ is small, which is an a posteriori verification of robustness for each given setting (nonlinear function a, domain Ω, datum f, mesh $\mathcal{T}_\ell$, linearization step k, polynomial degree p). We also discuss in Remark 2.5.7 why we can expect $\widehat{C}_\ell^k$ tend to 1 in the discretization limit (ℓ large enough).

The rest of the paper is organized as follows. In Section 2.2, we detail the assumptions on the nonlinear function a. We next give the continuous weak formulation with its equivalent energy minimization. Then, we introduce the discrete weak formulation with its associated discrete energy minimization, and finally the iterative linearization. In Section 2.3, we define the convex conjugate, the duality setting, and the equilibrated flux necessary for our a posteriori estimates. In Section 2.4, we study the original energy difference, which leads us to our first main result, Theorem 2.4.4, giving details of (2.2). In Section 2.5, we define the augmented energy difference and estimator and state our second main result, Theorem 2.5.5, giving details of (2.3). In Section 2.6, we present a series of numerical experiments in order to illustrate our theoretical findings, for both settings (2.2) and (2.3) as well as smooth and singular solutions. In Section 2.7, we give a proof of crucial technical result of Lemma 2.4.1, and then in Section 2.8, a proof of Theorem 2.4.4. In Section 2.9, we give a proof of the boundedness of the weight λ_ℓ^k from Lemma 2.5.1 and of the augmented energy difference consistency summarized in Lemma 2.5.3, and then, in Section 2.10, a proof of Theorem 2.5.5. Finally, we summarize, in Appendix 2.A, some useful properties of the nonlinear functions and the assumptions required, and, in Appendix 2.B, we show some technical results to determine the local eigenvalues of the Newton linearization.

2.2 Weak formulation, energy minimization, finite element discretization, and iterative linearization

Let $\Omega \subset \mathbb{R}^d$, $d \ge 1$, be an open polytope with Lipschitz boundary $\partial\Omega$. We consider problem (2.1), where $f \in L^2(\Omega)$ represents a volumetric force term, while a is the diffusion coefficient which depends on the potential $u : \Omega \to \mathbb{R}$ only through the Euclidean norm of its gradient $|\boldsymbol{\nabla} u|$.

We consider the following assumption for the nonlinear function a (see, e.g., [129] for more details).

Assumption 2.2.1 (Nonlinear function a)**.** *We assume that the function $a : \Omega \times [0,\infty) \to (0,\infty)$ is measurable and that there exist constants $a_{\mathrm{m}} \le a_{\mathrm{c}} \in (0,\infty)$ such that, a.e. in Ω and for all $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d$,*

$$|a(\cdot,|\boldsymbol{x}|)\boldsymbol{x} - a(\cdot,|\boldsymbol{y}|)\boldsymbol{y}| \le a_{\mathrm{c}}|\boldsymbol{x}-\boldsymbol{y}| \qquad \textit{(Lipschitz continuity)}, \tag{2.4a}$$

$$(a(\cdot,|\boldsymbol{x}|)\boldsymbol{x} - a(\cdot,|\boldsymbol{y}|)\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y}) \ge a_{\mathrm{m}}|\boldsymbol{x}-\boldsymbol{y}|^2 \qquad \textit{(strong monotonicity)}. \tag{2.4b}$$

2.2.1 Weak formulation and equivalent energy minimization

The weak formulation of problem (2.1) reads: find $u \in H^1_0(\Omega)$ such that

$$(a(\cdot,|\boldsymbol{\nabla} u|)\boldsymbol{\nabla} u, \boldsymbol{\nabla} v) = (f,v) \qquad \forall v \in H^1_0(\Omega), \tag{2.5a}$$

where $(\cdot,\cdot)$ is the inner product of $L^2(\Omega)$.

Referring to [129], the weak formulation (2.5a) is equivalent to the following minimization problem:

$$u = \arg\min_{v\in H^1_0(\Omega)} \mathcal{J}(v), \tag{2.5b}$$

with the energy functional $\mathcal{J} : H^1_0(\Omega) \to \mathbb{R}$ defined as,

$$\mathcal{J}(v) \coloneqq \int_\Omega \phi(\cdot,|\boldsymbol{\nabla} v|) - (f,v), \quad v \in H^1_0(\Omega), \tag{2.6}$$

where the function $\phi : \Omega \times [0,\infty) \to [0,\infty)$ is defined such that, a.e. in Ω and for all $r \in [0,\infty)$,

$$\phi(\cdot,r) \coloneqq \int_0^r a(\cdot,s)s\,\mathrm{d}s. \tag{2.7}$$

It is shown in [129] that, under Assumption 2.2.1, there exists a unique solution to problem (2.5). We refer to Appendix 2.A for more details about equivalent assumptions on the nonlinear functions a and ϕ.

2.2.2 Finite element discretization

Let $\ell \ge 0$ be a mesh level index. We consider simplicial triangulations $\mathcal{T}_\ell$ of the domain Ω satisfying the following shape-regularity property: there exists a constant $\kappa_{\mathcal{T}} > 0$ such that for all $\ell \ge 0$ and all $K \in \mathcal{T}_\ell$, $h_K/\rho_K \le \kappa_{\mathcal{T}}$, where h_K is the diameter of K and ρ_K is the diameter of the largest ball inscribed in K.

For a polynomial degree $p \ge 1$, denoting $\mathbb{P}_p(\mathcal{T}_\ell)$ the space of piecewise polynomials on the mesh $\mathcal{T}_\ell$ of total degree at most p, we define the discrete finite element space $V_\ell^p \coloneqq \mathbb{P}_p(\mathcal{T}_\ell) \cap H^1_0(\Omega)$. The finite element approximation of (2.5a) would be $u_\ell \in V_\ell^p$ such that

$$(a(\cdot,|\boldsymbol{\nabla} u_\ell|)\boldsymbol{\nabla} u_\ell, \boldsymbol{\nabla} v_\ell) = (f,v_\ell) \qquad \forall v_\ell \in V_\ell^p. \tag{2.8a}$$

As in (2.5b), $u_\ell \in V_\ell^p$ solves the minimization problem

$$u_\ell = \arg\min_{v_\ell \in V_\ell^p} \mathcal{J}(v_\ell). \tag{2.8b}$$

Below, we never work with u_ℓ but rather with its approximations coming from iterative linearization.

2.2.3 Iterative linearization

We henceforth consider an iterative linearization of (2.8a), which is anyhow necessary for a practical solution of (2.8a). Let $u_\ell^0 \in V_\ell^p$ be a given initial guess. For an iterative linearization index $k \geq 1$, consider $\boldsymbol{A}_\ell^{k-1} : \Omega \to \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_\ell^{k-1} : \Omega \to \mathbb{R}^d$, arising from a suitable linearization; details and examples are given below. We define the linearized finite element approximation: $u_\ell^k \in V_\ell^p$ to be such that

$$(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k, \boldsymbol{\nabla} v_\ell) = (f, v_\ell) + (\boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla} v_\ell) \qquad \forall v_\ell \in V_\ell^p. \tag{2.9a}$$

As in (2.8b), this is equivalent to the discrete minimization problem

$$u_\ell^k = \arg\min_{v_\ell \in V_\ell^p} \mathcal{J}_\ell^{k-1}(v_\ell) \tag{2.9b}$$

with the linearized energy functional $\mathcal{J}_\ell^{k-1} : H_0^1(\Omega) \to \mathbb{R}$ defined for all $v \in H_0^1(\Omega)$ by

$$\mathcal{J}_\ell^{k-1}(v) \coloneqq \frac{1}{2}\left\|(\boldsymbol{A}_\ell^{k-1})^{\frac{1}{2}}\boldsymbol{\nabla} v\right\|^2 - (f, v) - (\boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla} v), \tag{2.10}$$

where $\|\cdot\|$ is the $L^2(\Omega)$ norm corresponding to the inner product $(\cdot,\cdot)$ of $L^2(\Omega)$.

2.2.3.1 Assumptions on iterative linearization schemes

Let $k \geq 1$. We will suppose that $\boldsymbol{A}_\ell^{k-1} : \Omega \to \mathbb{R}^{d\times d}$ and $\boldsymbol{b}_\ell^{k-1} : \Omega \to \mathbb{R}^d$ from (2.9) satisfy Assumption 2.2.2 below (for the sake of conciseness, we assume that they are well defined everywhere in Ω). We will use the following notation of the derivatives in the second argument of the functions a, ϕ, and others (cf. (2.21)–(2.22)): for all $r \in [0,\infty)$, $a'(\cdot, r) \coloneqq \frac{\partial}{\partial r} a(\boldsymbol{x}, r)$ and $\phi'(\cdot, r) \coloneqq \frac{\partial}{\partial r}\phi(\boldsymbol{x}, r)$.

Assumption 2.2.2 (Iterative linearization)**.** *For all points $\boldsymbol{x} \in \Omega$, we assume that $\boldsymbol{A}_\ell^{k-1}(\boldsymbol{x}) \in \mathbb{R}^{d\times d}$ is a bounded symmetric positive definite matrix. Specifically, denoting by $A_{\mathrm{m},\ell}^{k-1}(\boldsymbol{x})$ and $A_{\mathrm{c},\ell}^{k-1}(\boldsymbol{x})$ respectively its smallest and largest pointwise eigenvalues, we have, for all $\boldsymbol{\xi} \in \mathbb{R}^d$,*

$$|\boldsymbol{A}_\ell^{k-1}(\boldsymbol{x})\boldsymbol{\xi}| \leq A_{\mathrm{c},\ell}^{k-1}(\boldsymbol{x})|\boldsymbol{\xi}| \quad \textit{(boundedness)}, \tag{2.11a}$$

$$(\boldsymbol{A}_\ell^{k-1}(\boldsymbol{x})\boldsymbol{\xi})\cdot\boldsymbol{\xi} \geq A_{\mathrm{m},\ell}^{k-1}(\boldsymbol{x})|\boldsymbol{\xi}|^2 \quad \textit{(positive definiteness)}. \tag{2.11b}$$

Moreover, we suppose uniformity, i.e., that there exist $A_{\mathrm{m}} \leq A_{\mathrm{c}} \in (0,\infty)$ independent of k, ℓ, and $\boldsymbol{x}$ such that

$$A_{\mathrm{m}} \leq A_{\mathrm{m},\ell}^{k-1}(\boldsymbol{x}) \leq A_{\mathrm{c},\ell}^{k-1}(\boldsymbol{x}) \leq A_{\mathrm{c}}. \tag{2.11c}$$

Finally, we explicitly define $\boldsymbol{b}_\ell^{k-1}(\boldsymbol{x}) \in \mathbb{R}^d$ for all $\boldsymbol{x} \in \Omega$ by

$$\boldsymbol{b}_\ell^{k-1}(\boldsymbol{x}) \coloneqq \boldsymbol{A}_\ell^{k-1}(\boldsymbol{x})\boldsymbol{\nabla} u_\ell^{k-1}(\boldsymbol{x}) - a(\boldsymbol{x}, |\boldsymbol{\nabla} u_\ell^{k-1}(\boldsymbol{x})|)\boldsymbol{\nabla} u_\ell^{k-1}(\boldsymbol{x}). \tag{2.12}$$

In the following, we use the boldface font to denote the spaces of multi-dimensional functions, e.g., $\boldsymbol{L}^2(\Omega)$.

Remark 2.2.3 (Assumption 2.2.2)**.** *Equality* (2.12) *implies that* (2.9a) *can be equivalently written as a problem for the increment* $u_\ell^k - u_\ell^{k-1}$ *on the left-hand side and the residual of* u_ℓ^{k-1} *on the right-hand side:*

$$(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_\ell^k - u_\ell^{k-1}), \boldsymbol{\nabla} v_\ell) = (f, v_\ell) - (a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)\boldsymbol{\nabla} u_\ell^{k-1}, \boldsymbol{\nabla} v_\ell) \quad \forall v_\ell \in V_\ell^p, \tag{2.13}$$

which is the form used in, e.g., [93]. Therefore, equality (2.12) *ensures that the discrete problem* (2.9a) *is consistent with the discrete problem* (2.8a) *in that*

$$\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1} \to a(\cdot, |\boldsymbol{\nabla} u_\ell|)\boldsymbol{\nabla} u_\ell \text{ in } \boldsymbol{L}^2(\Omega) \quad \text{when} \quad u_\ell^k \to u_\ell \text{ in } H_0^1(\Omega). \tag{2.14}$$

Indeed, (2.11a) *and* (2.11c) *imply that* $\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_\ell^k - u_\ell^{k-1}) \to \boldsymbol{0}$ *in* $\boldsymbol{L}^2(\Omega)$*, whereas* $a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)\boldsymbol{\nabla} u_\ell^{k-1} \to a(\cdot, |\boldsymbol{\nabla} u_\ell|)\boldsymbol{\nabla} u_\ell$ *in* $\boldsymbol{L}^2(\Omega)$ *thanks to* (2.4a)*. Finally, we recall that the positive definiteness of* $\boldsymbol{A}_\ell^{k-1}$ *implies that* $(\boldsymbol{A}_\ell^{k-1})^{-1}$ *and* $(\boldsymbol{A}_\ell^{k-1})^{\frac{1}{2}}$ *exist, which is used below.*

2.2.3.2 Examples of iterative linearization schemes

We now present some examples of linearization methods satisfying Assumption 2.2.2.

Example 2.2.4 (Picard)**.** *The Picard (fixed point) iteration, see, e.g., [42], is defined as*

$$\boldsymbol{A}_\ell^{k-1} = a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)\boldsymbol{I}_d \quad \text{with} \quad \boldsymbol{b}_\ell^{k-1} = \boldsymbol{0} \quad \text{in } \Omega. \tag{2.15}$$

It satisfies Assumption 2.2.2 with $A_{\mathrm{m},\ell}^{k-1} = A_{\mathrm{c},\ell}^{k-1} = a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)$*, which leads to* $A_{\mathrm{m}} = a_{\mathrm{m}}$ *and* $A_{\mathrm{c}} = a_{\mathrm{c}}$ *thanks to* (2.85)*.*

Example 2.2.5 (Zarantonello)**.** *The Zarantonello iteration, introduced in [127], is defined as*

$$\boldsymbol{A}_\ell^{k-1} = \gamma \boldsymbol{I}_d \quad \text{with} \quad \boldsymbol{b}_\ell^{k-1} = \Big(\gamma - a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)\Big)\boldsymbol{\nabla} u_\ell^{k-1} \quad \text{in } \Omega, \tag{2.16}$$

where $\gamma \in (0, \infty)$ *is a constant parameter. To ensure contraction, one needs to assume that* $\gamma \geq \frac{a_{\mathrm{c}}^2}{a_{\mathrm{m}}}$*. The Zarantonello iteration maintains the same linearization matrix* $\gamma \boldsymbol{I}_d$ *during the iterations and converges linearly, but the convergence is slow as* γ *takes large values. It satisfies Assumption 2.2.2 with* $A_{\mathrm{m},\ell}^{k-1} = A_{\mathrm{c},\ell}^{k-1} = \gamma$*, which leads to* $A_{\mathrm{m}} = A_{\mathrm{c}} = \gamma$*.*

Example 2.2.6 ((Damped) Newton)**.** *The (damped) Newton iteration, see, e.g., [42], is defined as*

$$\begin{aligned} \boldsymbol{A}_\ell^{k-1} &= a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)\boldsymbol{I}_d + \theta \frac{a'(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)}{|\boldsymbol{\nabla} u_\ell^{k-1}|}\boldsymbol{\nabla} u_\ell^{k-1} \otimes \boldsymbol{\nabla} u_\ell^{k-1} \\ \text{with} \quad \boldsymbol{b}_\ell^{k-1} &= \theta a'(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)|\boldsymbol{\nabla} u_\ell^{k-1}|\boldsymbol{\nabla} u_\ell^{k-1} \quad \text{in } \Omega, \end{aligned} \tag{2.17}$$

where $\theta \in [0, 1]$ *is the damping parameter. Observe that* $\theta = 1$ *gives the Newton iteration, whereas* $\theta = 0$ *corresponds to the Picard iteration. If* $\theta = 1$*, the Newton method converges quadratically. However, it might not always converge. The (damped) Newton iteration satisfies Assumption 2.2.2 with, if the space dimension* $d = 1$*,*

$$\begin{aligned} A_{\mathrm{m},\ell}^{k-1} = A_{\mathrm{c},\ell}^{k-1} &= a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|) + \theta a'(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|)|\boldsymbol{\nabla} u_\ell^{k-1}| \\ &\overset{(2.90)}{=} (1-\theta) a(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|) + \theta \phi''(\cdot, |\boldsymbol{\nabla} u_\ell^{k-1}|), \end{aligned} \tag{2.18}$$

and, if $d > 1$,

$$A_{\mathrm{m},\ell}^{k-1} = (1-\theta)a(\cdot,|\nabla u_\ell^{k-1}|) + \theta \min(a(\cdot,|\nabla u_\ell^{k-1}|), \phi''(\cdot,|\nabla u_\ell^{k-1}|)), \tag{2.19a}$$

$$A_{\mathrm{c},\ell}^{k-1} = (1-\theta)a(\cdot,|\nabla u_\ell^{k-1}|) + \theta \max(a(\cdot,|\nabla u_\ell^{k-1}|), \phi''(\cdot,|\nabla u_\ell^{k-1}|)). \tag{2.19b}$$

Indeed, denoting the spectrum of a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ *by* $\mathrm{Spec}(\boldsymbol{A})$*, we infer* (2.19) *by writing,*

$$\begin{aligned}\mathrm{Spec}(\boldsymbol{A}_\ell^{k-1}) &\overset{(2.91)}{=} \{a(\cdot,|\nabla u_\ell^{k-1}|), a(\cdot,|\nabla u_\ell^{k-1}|) + \theta a'(\cdot,|\nabla u_\ell^{k-1}|)|\nabla u_\ell^{k-1}|\}\\ &\overset{(2.90)}{=} \{(1-\theta)a(\cdot,|\nabla u_\ell^{k-1}|) + \theta a(\cdot,|\nabla u_\ell^{k-1}|),\\ &\qquad (1-\theta)a(\cdot,|\nabla u_\ell^{k-1}|) + \theta\phi''(\cdot,|\nabla u_\ell^{k-1}|)\}.\end{aligned}$$

Finally, we can set $A_{\mathrm{m}} = a_{\mathrm{m}}$ *and* $A_{\mathrm{c}} = a_{\mathrm{c}}$ *thanks to* (2.85) *and* (2.87).

2.3 Convex conjugate, dual energy, and flux equilibration

In this section, we define some common tools for the forthcoming developments.

2.3.1 Convex conjugate function and dual energy

Recalling the primal energy $\mathcal{J}$ of (2.6), the corresponding dual energy $\mathcal{J}^* : \boldsymbol{H}(\mathrm{div},\Omega) \to \mathbb{R}$, cf. [10, 72, 107, 129], is defined by

$$\mathcal{J}^*(\boldsymbol{w}) \coloneqq -\int_\Omega \phi^*(\cdot,|\boldsymbol{w}|), \quad \boldsymbol{w} \in \boldsymbol{H}(\mathrm{div},\Omega), \tag{2.20}$$

where $\phi^* : \Omega \times [0,\infty) \to [0,\infty)$ is the convex conjugate of ϕ (also known as the Legendre dual or the Fenchel conjugate), which is defined such that, a.e. in Ω and for all $s \in [0,\infty)$,

$$\phi^*(\cdot,s) \coloneqq \sup_{r\in[0,\infty)} (sr - \phi(\cdot,r)), \tag{2.21a}$$

or equivalently, for all $s \in [0,\infty)$,

$$\phi^*(\cdot,s) = \int_0^s \phi'^{-1}(\cdot,r)\,\mathrm{d}r = s\phi'^{-1}(\cdot,s) - \phi(\cdot,\phi'^{-1}(\cdot,s)). \tag{2.21b}$$

We refer to [72] for more details and recall that the construction of ϕ^* yields

$$\phi^{*\prime} = \phi'^{-1} \quad \text{and} \quad \phi^{*\prime\prime} = \frac{1}{\phi''\circ\phi'^{-1}}. \tag{2.22}$$

Consequently, under Assumption 2.2.1, ϕ^* is convex thanks to Remark 2.A.3 below.

Finally, we define the linearized dual energy functional $\mathcal{J}_\ell^{*,k-1} : \boldsymbol{H}(\mathrm{div},\Omega) \to \mathbb{R}$ such that

$$\mathcal{J}_\ell^{*,k-1}(\boldsymbol{w}) \coloneqq -\frac{1}{2}\left\|(\boldsymbol{A}_\ell^{k-1})^{-\frac{1}{2}}(\boldsymbol{w} - \boldsymbol{b}_\ell^{k-1})\right\|^2, \quad \boldsymbol{w} \in \boldsymbol{H}(\mathrm{div},\Omega). \tag{2.23}$$

2.3.2 Flux equilibration

Let $\mathcal{V}_\ell$ be the set of all mesh vertices and, for each vertex $\boldsymbol{a} \in \mathcal{V}_\ell$, define the patch $\mathcal{T}_\ell^{\boldsymbol{a}}$ as the collection of the simplicies of $\mathcal{T}_\ell$ sharing the vertex $\boldsymbol{a}$, as well as the patch subdomain $\omega_\ell^{\boldsymbol{a}}$ corresponding to $\mathcal{T}_\ell^{\boldsymbol{a}}$. For all $\boldsymbol{a} \in \mathcal{V}_\ell$, we define the space $\boldsymbol{V}_\ell^{\boldsymbol{a}} \coloneqq \boldsymbol{RTN}_p(\mathcal{T}_\ell^{\boldsymbol{a}}) \cap \boldsymbol{H}_0(\mathrm{div}, \omega_\ell^{\boldsymbol{a}})$. Here $\boldsymbol{RTN}_p(\mathcal{T}_\ell^{\boldsymbol{a}})$ denotes the broken space consisting of p-th order Raviart–Thomas–Nédélec space on each simplex, $\boldsymbol{RTN}_p(K) \coloneqq [\mathbb{P}_p(K)]^d + \boldsymbol{x}\mathbb{P}_p(K)$. Moreover, $\boldsymbol{H}_0(\mathrm{div}, \omega_\ell^{\boldsymbol{a}})$ is the subspace of $\boldsymbol{H}(\mathrm{div}, \omega_\ell^{\boldsymbol{a}})$ of functions with vanishing normal trace on $\partial\omega_\ell^{\boldsymbol{a}}$ if $\boldsymbol{a} \in \mathcal{V}_\ell$ is an interior vertex and of functions with vanishing normal trace on $\partial\omega_\ell^{\boldsymbol{a}} \backslash \{\psi_\ell^{\boldsymbol{a}} > 0\}$ if $\boldsymbol{a} \in \mathcal{V}_\ell$ is a boundary vertex. Here, for all $\boldsymbol{a} \in \mathcal{V}_\ell$, the hat function $\psi_\ell^{\boldsymbol{a}}$ is the continuous, piecewise affine function equal to 1 in $\boldsymbol{a}$ and 0 in $\mathcal{V}_\ell \backslash \{\boldsymbol{a}\}$. We recall the partition of unity

$$\sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \psi_\ell^{\boldsymbol{a}}(\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \in \Omega. \tag{2.24}$$

We denote by $\Pi_{\ell,p}$ the L^2-orthogonal projection from $L^2(\Omega)$ to $\mathbb{P}_p(\mathcal{T}_\ell)$ and by $\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}}$ the $\boldsymbol{L}^2$-orthogonal projection from $\boldsymbol{L}^2(\Omega)$ to $\boldsymbol{RTN}_{p-1}(\mathcal{T}_\ell)$; note that both are elementwise. Finally, we consider the equilibrated flux locally reconstructed from u_ℓ^k as

$$\boldsymbol{\sigma}_\ell^k \coloneqq \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}, \tag{2.25a}$$

where, for all vertices $\boldsymbol{a} \in \mathcal{V}_\ell$, following [21, 35, 41, 53, 54],

$$\begin{aligned} \boldsymbol{\sigma}_\ell^{\boldsymbol{a},k} &\coloneqq \arg \min_{\substack{\boldsymbol{w}_\ell \in \boldsymbol{V}_\ell^{\boldsymbol{a}} \\ \boldsymbol{\nabla}\cdot\boldsymbol{w}_\ell = \Pi_{\ell,p}\gamma_\ell^{\boldsymbol{a},k}}} \|(\boldsymbol{\Upsilon}_\ell^k)^{-\frac{1}{2}}(\psi_\ell^{\boldsymbol{a}} \boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}} \boldsymbol{\xi}_\ell^k + \boldsymbol{w}_\ell)\|_{\omega_\ell^{\boldsymbol{a}}}, \\ \text{with } \boldsymbol{\xi}_\ell^k &\coloneqq \boldsymbol{A}_\ell^{k-1} \boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1}, \quad \gamma_\ell^{\boldsymbol{a},k} \coloneqq \psi_\ell^{\boldsymbol{a}} f - \boldsymbol{\nabla}\psi_\ell^{\boldsymbol{a}} \cdot \boldsymbol{\xi}_\ell^k \end{aligned} \tag{2.25b}$$

and where the weight $\boldsymbol{\Upsilon}_\ell^k$ will be chosen according to needs. Specifically, we will set

$$\boldsymbol{\Upsilon}_\ell^k \coloneqq \boldsymbol{I}_d \text{ in Section 2.4}, \tag{2.26a}$$

$$\boldsymbol{\Upsilon}_\ell^k \coloneqq \boldsymbol{A}_\ell^{k-1} \text{ in Section 2.5}. \tag{2.26b}$$

We note that

$$(\gamma_\ell^{\boldsymbol{a},k}, 1)_{\omega_\ell^{\boldsymbol{a}}} = 0$$

for all interior vertices $\boldsymbol{a} \in \mathcal{V}_\ell$, which is an immediate consequence of (2.9a) with the test function $v_\ell = \psi_\ell^{\boldsymbol{a}} \in V_\ell^p$. Consequently, problems (2.25b) are well posed.

Combining (2.25) and (2.24), we infer, as in, e.g., [52],

$$\begin{aligned} \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}_\ell^k &= \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k} = \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \Pi_{\ell,p}(\psi_\ell^{\boldsymbol{a}} f) - \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} \Pi_{\ell,p}(\boldsymbol{\nabla}\psi_\ell^{\boldsymbol{a}} \cdot \boldsymbol{\xi}_\ell^k) \\ &= \Pi_{\ell,p} \sum_{\boldsymbol{a} \in \mathcal{V}_\ell} (\psi_\ell^{\boldsymbol{a}} f) = \Pi_{\ell,p} f. \end{aligned} \tag{2.27}$$

In particular, $\boldsymbol{\sigma}_\ell^k \in \boldsymbol{RTN}_p(\mathcal{T}_\ell) \cap \boldsymbol{H}(\mathrm{div}, \Omega)$ and we have with the Green theorem, since $u_\ell^k \in V_\ell^p = \mathbb{P}_p(\mathcal{T}_\ell) \cap H_0^1(\Omega)$,

$$-(\boldsymbol{\sigma}_\ell^k, \boldsymbol{\nabla} u_\ell^k) = (f, u_\ell^k). \tag{2.28}$$

2.4 A posteriori estimate of the energy difference

This section gives an a posteriori estimate of the energy difference.

2.4.1 Energy difference and the associated estimator

We define the (square root of twice the) energy difference corresponding to the nonlinear problem (2.5) by

$$\mathcal{E}_{\mathrm{N},\ell}^k \coloneqq \Big(2(\mathcal{J}(u_\ell^k) - \mathcal{J}(u))\Big)^{\frac{1}{2}}. \tag{2.29a}$$

Note that $\mathcal{E}_{\mathrm{N},\ell}^k$ is well defined from (2.5b) and the fact that $u_\ell^k \in H_0^1(\Omega)$. Actually, $\mathcal{E}_{\mathrm{N},\ell}^k \geq 0$ and 0 if and only if $u_\ell^k = u$ from the uniqueness of u in (2.5b). We then define the estimator $\eta_{\mathrm{N},\ell}^k$ corresponding to the nonlinear problem (2.5) as in, e.g., [11, 107, 129], employing the dual energy $\mathcal{J}^*$ of (2.20) and the equilibrated flux $\boldsymbol{\sigma}_\ell^k$ of (2.25) with the choice (2.26a),

$$\eta_{\mathrm{N},\ell}^k \coloneqq \Big(2(\mathcal{J}(u_\ell^k) - \mathcal{J}^*(\boldsymbol{\sigma}_\ell^k))\Big)^{\frac{1}{2}}. \tag{2.29b}$$

Note that $\eta_{\mathrm{N},\ell}^k$ is well defined (the argument of the square root is nonnegative), which can be seen from (2.55)–(2.56) below.

2.4.2 Locally-weighted bounds for the energy difference and the associated estimator

Denoting, a.e. in Ω,

$$a_\ell^k \coloneqq a(\cdot, |\boldsymbol{\nabla} u_\ell^k|) \quad \text{and} \quad a_u \coloneqq a(\cdot, |\boldsymbol{\nabla} u|), \tag{2.30}$$

we define, for all $(v, \boldsymbol{w}) \in L^1(\Omega) \times [L^1(\Omega)]^d$, a.e. in Ω,

$$a_{\mathrm{m},\ell}^k(v, \boldsymbol{w}) \coloneqq \min\left(a(\cdot, |\boldsymbol{\nabla} v|), \operatorname*{ess\,inf}_{r \in (|\boldsymbol{w}|, |a_\ell^k \boldsymbol{\nabla} u_\ell^k|)} \phi''(\cdot, \phi'^{-1}(\cdot, r))\right) \overset{(2.85),(2.87)}{\in} [a_{\mathrm{m}}, a_{\mathrm{c}}], \tag{2.31a}$$

$$a_{\mathrm{c},\ell}^k(v, \boldsymbol{w}) \coloneqq \max\left(a(\cdot, |\boldsymbol{\nabla} v|), \operatorname*{ess\,sup}_{r \in (|\boldsymbol{w}|, |a_\ell^k \boldsymbol{\nabla} u_\ell^k|)} \phi''(\cdot, \phi'^{-1}(\cdot, r))\right) \overset{(2.85),(2.87)}{\in} [a_{\mathrm{m}}, a_{\mathrm{c}}], \tag{2.31b}$$

and, for the sake of brevity, we denote for both $\alpha \in \{\mathrm{m}, \mathrm{c}\}$, a.e. in Ω,

$$a_{\alpha,\ell}^{\boldsymbol{\sigma},k} \coloneqq a_{\alpha,\ell}^k(u_\ell^k, \boldsymbol{\sigma}_\ell^k), \quad a_{\alpha,\ell}^{\boldsymbol{\nabla} u,k} \coloneqq a_{\alpha,\ell}^k(u_\ell^k, a_u \boldsymbol{\nabla} u), \quad \text{and} \quad a_{\alpha,\ell}^{u,k} \coloneqq a_{\alpha,\ell}^k(u, a_u \boldsymbol{\nabla} u). \tag{2.31c}$$

Observe that $a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}$ and $a_{\mathrm{c},\ell}^{\boldsymbol{\sigma},k}$ are computable, in contrast to the other terms in (2.31c).

Lemma 2.4.1 (Locally-weighted bounds for the energy difference and estimator)**.** *Recalling* (2.29)–(2.31)*, we have*

$$\|(a_{\mathrm{m},\ell}^{u,k})^{\frac{1}{2}}(\boldsymbol{\nabla} u_\ell^k - \boldsymbol{\nabla} u)\|^2 \leq (\mathcal{E}_{\mathrm{N},\ell}^k)^2 \leq \|(a_{\mathrm{c},\ell}^{u,k})^{\frac{1}{2}}(\boldsymbol{\nabla} u_\ell^k - \boldsymbol{\nabla} u)\|^2, \tag{2.32a}$$

$$\|(a_{\mathrm{c},\ell}^{\boldsymbol{\sigma},k})^{-\frac{1}{2}}(a_\ell^k \boldsymbol{\nabla} u_\ell^k + \boldsymbol{\sigma}_\ell^k)\|^2 \leq (\eta_{\mathrm{N},\ell}^k)^2 \leq \|(a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k})^{-\frac{1}{2}}(a_\ell^k \boldsymbol{\nabla} u_\ell^k + \boldsymbol{\sigma}_\ell^k)\|^2, \tag{2.32b}$$

$$\|(a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k})^{-\frac{1}{2}}(a_\ell^k \boldsymbol{\nabla} u_\ell^k - a_u \boldsymbol{\nabla} u)\|^2 \leq (\mathcal{E}_{\mathrm{N},\ell}^k)^2 \leq 2\|(a_{\mathrm{m},\ell}^{\boldsymbol{\nabla} u,k})^{-\frac{1}{2}}(a_\ell^k \boldsymbol{\nabla} u_\ell^k - a_u \boldsymbol{\nabla} u)\|^2. \tag{2.32c}$$

The proof of this important technical result is postponed to Section 2.7.

2.4.3 Data oscillation, quadrature-type, and iterative linearization estimators

Following, e.g, [52], let

$$(\eta_{\mathrm{osc},\ell}^k)^2 \coloneqq \sum_{K\in\mathcal{T}_\ell}\left[\frac{h_K}{\pi a_{\mathrm{m}}^{\frac12}}\|(I-\Pi_{\ell,p})f\|_K\right]^2, \tag{2.33a}$$

$$(\eta_{\mathrm{osc,q},\ell}^k)^2 \coloneqq \sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k})^2 \quad\text{and}\quad (\eta_{\mathrm{iter},\ell}^k)^2 \coloneqq \sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k})^2, \tag{2.33b}$$

where, for all vertices $\boldsymbol{a}\in\mathcal{V}_\ell$, recalling the notation from (2.25b),

$$(\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k})^2 \coloneqq \frac{1}{\operatorname{ess\,inf}_{\omega_\ell^{\boldsymbol{a}}} a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}\left(\sum_{K\in\mathcal{T}_\ell^{\boldsymbol{a}}}\left[\frac{h_K}{\pi}\|(I-\Pi_{\ell,p})\gamma_\ell^{\boldsymbol{a},k}\|_K\right]^2+\|\psi_\ell^{\boldsymbol{a}}(\boldsymbol{I}-\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}})\boldsymbol{\xi}_\ell^k\|_{\omega_\ell^{\boldsymbol{a}}}^2\right), \tag{2.33c}$$

$$(\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k})^2 \coloneqq \frac{1}{\operatorname{ess\,inf}_{\omega_\ell^{\boldsymbol{a}}} a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}\|a_\ell^k\boldsymbol{\nabla}u_\ell^k-\boldsymbol{\xi}_\ell^k\|_{\omega_\ell^{\boldsymbol{a}}}^2. \tag{2.33d}$$

Remark 2.4.2 (Data oscillation and quadrature-type estimators)**.** *The estimator $\eta_{\mathrm{osc},\ell}^k$ monitors the so-called oscillation in the source datum f: it vanishes if f is piecewise polynomial and is of higher order for piecewise smooth f. The quadrature-type estimators $\eta_{\mathrm{osc,q},\ell}^k$ arise from piecewise polynomial approximation of the (possibly) apolynomial data $\gamma_\ell^{\boldsymbol{a},k}$ and $\boldsymbol{\xi}_\ell^k$ defined in (2.25b), which involve themselves the nonlinear function a through $\boldsymbol{A}_\ell^{k-1}$ and $\boldsymbol{b}_\ell^{k-1}$, cf. Section 2.2.3.2. These terms can completely disappear (for the lowest polynomial degree $p=1$ where $a(\cdot,|\boldsymbol{\nabla}u_\ell^{k-1}|)$ and $a'(\cdot,|\boldsymbol{\nabla}u_\ell^{k-1}|)$ are piecewise constant if the spatial dependence in a is piecewise constant) or be of higher order if the nonlinear function a has the necessary smoothness. One could actually force them to be negligible if a separate, sufficiently increased polynomial degree was chosen for the equilibration in (2.25b).*

Remark 2.4.3 (Iterative linearization estimator)**.** *Congruently with (2.14), if $u_\ell^k\to u_\ell$ in $H_0^1(\Omega)$, then we can have $\eta_{\mathrm{iter},\ell}^k$ as small as we need since, also using the standard coloring bound (2.68),*

$$\begin{aligned}\eta_{\mathrm{iter},\ell}^k &\overset{(2.31a),(2.25b),(2.12)}{\leq} a_{\mathrm{m}}^{-\frac12}(d+1)\|a_\ell^k\boldsymbol{\nabla}u_\ell^k-a_\ell^{k-1}\boldsymbol{\nabla}u_\ell^{k-1}-\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_\ell^k-u_\ell^{k-1})\|\\ &\overset{(2.4a),(2.11)}{\leq} a_{\mathrm{m}}^{-\frac12}(d+1)(a_{\mathrm{c}}+A_{\mathrm{c}})\|\boldsymbol{\nabla}(u_\ell^k-u_\ell^{k-1})\|.\end{aligned} \tag{2.34}$$

2.4.4 A posteriori estimate of the energy difference

We now present our first main result, giving an a posteriori estimate for the energy difference $\mathcal{E}_{\mathrm{N},\ell}^k$ and the estimator $\eta_{\mathrm{N},\ell}^k$ defined in (2.29).

Theorem 2.4.4 (A posteriori estimate of the energy difference)**.** *Suppose Assumption 2.2.1 and let $u\in H_0^1(\Omega)$ be the weak solution of (2.5). Let u_ℓ^k be its finite element approximation given by (2.9) on mesh $\mathcal{T}_\ell$, $\ell\geq 0$, and linearization step $k\geq 1$, for any iterative linearization satisfying Assumption 2.2.2. Let $\boldsymbol{\sigma}_\ell^k$ be the equilibrated flux defined by (2.25) with the choice (2.26a). Then*

$$\mathcal{E}_{\mathrm{N},\ell}^k \leq \eta_{\mathrm{N},\ell}^k + 2\eta_{\mathrm{osc},\ell}^k, \tag{2.35a}$$

$$\eta_{\mathrm{N},\ell}^k \lesssim C_\ell^k \mathcal{E}_{\mathrm{N},\ell}^k + \eta_{\mathrm{osc,q},\ell}^k + \eta_{\mathrm{iter},\ell}^k, \tag{2.35b}$$

where the hidden constant only depends on the space dimension d, the mesh shape-regularity $\kappa_{\mathcal{T}}$, and possibly, when $d \geq 4$, the polynomial degree p, with

$$C_\ell^k \coloneqq \max_{\boldsymbol{a}\in\mathcal{V}_\ell} \left(\frac{\operatorname{ess\,sup}_{\omega_\ell^{\boldsymbol{a}}} a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}}{\operatorname{ess\,inf}_{\omega_\ell^{\boldsymbol{a}}} a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}} \right)^{\frac{1}{2}} \overset{(2.31)}{\leq} \left(\frac{a_{\mathrm{c}}}{a_{\mathrm{m}}} \right)^{\frac{1}{2}}. \tag{2.36}$$

Proof. See Section 2.8. □

Remark 2.4.5 (Structure of C_ℓ^k and robustness)**.** *The constant C_ℓ^k from (2.36) is composed of purely local contributions. Since the ratios of $a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}$ to $a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}$ on each patch $\omega_\ell^{\boldsymbol{a}}$ are smaller than the global ratio $a_{\mathrm{c}}/a_{\mathrm{m}}$ and since the patches $\omega_\ell^{\boldsymbol{a}}$ shrink with mesh refinement, C_ℓ^k may converge to 1. For example, in the case where the problem is smooth in that $a \in C^1(\Omega \times [0,\infty))$, $u \in C^1(\Omega)$, and $\boldsymbol{\nabla} u_\ell^k \to \boldsymbol{\nabla} u$ in $[L^\infty(\Omega)]^d$, then, for ℓ and k large enough,*

$$C_\ell^k \overset{(2.31)}{\lesssim} \widetilde{C}_\ell^k \coloneqq \operatorname*{ess\,sup}_{\boldsymbol{x}\in\Omega} \left(\max\left(\frac{\phi''}{a}, \frac{a}{\phi''} \right)(\boldsymbol{x}, |\boldsymbol{\nabla} u_\ell^k(\boldsymbol{x})|) \right)^{\frac{1}{2}}, \tag{2.37}$$

which is small if the function ϕ''/a is close to 1 over $\Omega\times[0,\infty)$. Unfortunately, C_ℓ^k from (2.36) cannot be computed and its value may possibly explode with the ratio $(a_{\mathrm{c}}/a_{\mathrm{m}})^{1/2}$.

In the next section, by augmenting the energy difference, we will achieve a result similar to (2.35) but with a computable constant C_ℓ^k which moreover takes value 1 for the Zarantonello linearization of Example 2.2.5.

2.5 A posteriori estimate of the augmented energy difference

This section gives an a posteriori estimate of the augmented energy difference defined below.

2.5.1 Energy difference and estimator of the linearized problem

In order to derive robust estimates, we now also crucially consider the linearized problem (2.9). We introduce the abstract linearization on the continuous level: find $u_{\langle\ell\rangle}^k \in H_0^1(\Omega)$ such that

$$(\boldsymbol{A}_\ell^{k-1} \boldsymbol{\nabla} u_{\langle\ell\rangle}^k, \boldsymbol{\nabla} v) = (f, v) + (\boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla} v) \qquad \forall v \in H_0^1(\Omega), \tag{2.38a}$$

which is a linear problem. Note that (2.38a) is equivalent to

$$u_{\langle\ell\rangle}^k \coloneqq \arg \min_{v\in H_0^1(\Omega)} \mathcal{J}_\ell^{k-1}(v), \tag{2.38b}$$

employing the linearized energy (2.10). Analogously to the (nonlinear) energy difference $\mathcal{E}_{\mathrm{N},\ell}^k$ (2.29a), we define the energy difference $\mathcal{E}_{\mathrm{L},\ell}^k$ of the linearized problem (2.38) as

$$\mathcal{E}_{\mathrm{L},\ell}^k \coloneqq \left(2\big(\mathcal{J}_\ell^{k-1}(u_\ell^k) - \mathcal{J}_\ell^{k-1}(u_{\langle\ell\rangle}^k) \big) \right)^{\frac{1}{2}} \overset{(2.77a)}{=} \| (\boldsymbol{A}_\ell^{k-1})^{\frac{1}{2}} \boldsymbol{\nabla}(u_\ell^k - u_{\langle\ell\rangle}^k) \|. \tag{2.39a}$$

Here, there holds trivially $\mathcal{E}_{\mathrm{L},\ell}^k \geq 0$ and $\mathcal{E}_{\mathrm{L},\ell}^k = 0$ if and only if $u_\ell^k = u_{\langle\ell\rangle}^k$. Finally, we define the estimator $\eta_{\mathrm{L},\ell}^k$ of the linearized problem (2.38), employing the linearized dual energy (2.23) and the equilibrated flux $\boldsymbol{\sigma}_\ell^k$ of (2.25) with the choice (2.26b), by

$$\begin{aligned} \eta_{\mathrm{L},\ell}^k &\coloneqq \left(2\big(\mathcal{J}_\ell^{k-1}(u_\ell^k) - \mathcal{J}_\ell^{*,k-1}(\boldsymbol{\sigma}_\ell^k) \big) \right)^{\frac{1}{2}} \\ &\overset{(2.78a)}{=} \| (\boldsymbol{A}_\ell^{k-1})^{-\frac{1}{2}} (\boldsymbol{A}_\ell^{k-1} \boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1} + \boldsymbol{\sigma}_\ell^k) \|. \end{aligned} \tag{2.39b}$$

2.5.2 Augmented energy difference and the associated estimator

Recalling the notation in (2.25b), as in (2.33d) but with the weight (2.26b), we define the iterative linearization estimator $\widehat{\eta}^k_{\mathrm{iter},\ell}$ by

$$\widehat{\eta}^k_{\mathrm{iter},\ell} \coloneqq \|(\boldsymbol{A}^{k-1}_\ell)^{-\frac{1}{2}}(a^k_\ell \boldsymbol{\nabla} u^k_\ell - \boldsymbol{\xi}^k_\ell)\|. \tag{2.40}$$

Observe that with the same reasoning as in Remark 2.4.3, if $(u^k_\ell)_{k\geq 1}$ converges in $H^1_0(\Omega)$, then we can have $\widehat{\eta}^k_{\mathrm{iter},\ell}$ as small as needed. Then, we define a computable weight $\lambda^k_\ell \geq 0$ from the estimators (2.29b) (using $\boldsymbol{\sigma}^k_\ell$ of (2.25) with the choice (2.26b)), (2.39b), and (2.40) by

$$\lambda^k_\ell \coloneqq \frac{\eta^k_{\mathrm{N},\ell}}{\eta^k_{\mathrm{L},\ell} + \widehat{\eta}^k_{\mathrm{iter},\ell}}. \tag{2.41}$$

Finally, we define the augmented energy difference $\mathcal{E}^k_\ell$ and estimator η^k_ℓ by

$$\mathcal{E}^k_\ell \coloneqq \frac{1}{2}\left(\mathcal{E}^k_{\mathrm{N},\ell} + \lambda^k_\ell \mathcal{E}^k_{\mathrm{L},\ell}\right) \quad \text{and} \quad \eta^k_\ell \coloneqq \frac{1}{2}\left(\eta^k_{\mathrm{N},\ell} + \lambda^k_\ell \eta^k_{\mathrm{L},\ell}\right). \tag{2.42}$$

Lemma 2.5.1 (Weight λ^k_ℓ)**.** *There holds*

$$\lambda^k_\ell \leq a_{\mathrm{m}}^{-\frac{1}{2}} A_{\mathrm{c}}^{\frac{1}{2}}. \tag{2.43}$$

Proof. See Section 2.9. □

Remark 2.5.2 (Weight λ^k_ℓ and uniform equivalence of η^k_ℓ and $\eta^k_{\mathrm{N},\ell}$)**.** *The weight λ^k_ℓ from (2.41) ensures a balance between the two components $\eta^k_{\mathrm{N},\ell}$ and $\lambda^k_\ell \eta^k_{\mathrm{L},\ell}$ in (2.42). In particular, $\lambda^k_\ell \eta^k_{\mathrm{L},\ell} = \eta^k_{\mathrm{N},\ell}$ in the limit $\widehat{\eta}^k_{\mathrm{iter},\ell} = 0$, whereby simply $\eta^k_\ell = \eta^k_{\mathrm{N},\ell}$. Moreover, from (2.41)–(2.42), at any linearization iteration $k \geq 1$,*

$$\frac{1}{2}\eta^k_{\mathrm{N},\ell} \leq \eta^k_\ell \leq \eta^k_{\mathrm{N},\ell}, \tag{2.44}$$

so that the augmented estimator η^k_ℓ of (2.42) is uniformly equivalent to the standard primal–dual gap estimator $\eta^k_{\mathrm{N},\ell}$ of (2.29b). This construction is well defined; in particular, the inclusion of the estimator $\widehat{\eta}^k_{\mathrm{iter},\ell}$ ensures that λ^k_ℓ is uniformly bounded for every $k \geq 1$ as per (2.43). This is in turn used in the error consistency of Lemma 2.5.3 and in Theorem 2.5.5 below.

Lemma 2.5.3 (Error consistency)**.** *Let the error $\mathcal{E}^k_\ell$ be given by (2.42). Then there holds*

$$\frac{1}{2}\mathcal{E}^k_{\mathrm{N},\ell} \leq \mathcal{E}^k_\ell \leq \frac{1}{2}\left(\mathcal{E}^k_{\mathrm{N},\ell} + a_{\mathrm{m}}^{-\frac{1}{2}} A_{\mathrm{c}}^{\frac{1}{2}} \mathcal{E}^k_{\mathrm{L},\ell}\right) \lesssim \mathcal{E}^k_{\mathrm{N},\ell} + \|\boldsymbol{\nabla}(u^k_\ell - u^{k-1}_\ell)\|, \tag{2.45}$$

where the last inequality holds up to constants depending on $a_{\mathrm{m}}, a_{\mathrm{c}}, A_{\mathrm{m}}, A_{\mathrm{c}}$. In particular, $\mathcal{E}^k_\ell \to 0$ if and only if $u^k_\ell \to u$ in $H^1_0(\Omega)$.

Proof. See Section 2.9. □

Remark 2.5.4 (Augmented energy difference)**.** *From (2.44), the estimators $\eta^k_{\mathrm{N},\ell}$ and η^k_ℓ are uniformly equivalent. In that respect, Theorem 2.5.5 shows that the standard primal–dual gap estimator $\eta^k_{\mathrm{N},\ell}$ gives a guaranteed and potentially robust bound for the augmented energy difference $\mathcal{E}^k_\ell$ in place of the energy difference $\mathcal{E}^k_{\mathrm{N},\ell}$ of Theorem 2.4.4. As per (2.45), the modification of $\mathcal{E}^k_{\mathrm{N},\ell}$ to $\mathcal{E}^k_\ell$ is a bit more subtle in terms of the equivalence constant. In practice, though, we expect the situation to be better in that the added linearization component $\mathcal{E}^k_{\mathrm{L},\ell}$ multiplied by the weight λ^k_ℓ makes again $\mathcal{E}^k_\ell$ comparable in size to $\mathcal{E}^k_{\mathrm{N},\ell}$.*

2.5.3 Data oscillation and quadrature-type estimators

As in Section 2.4.3, let

$$(\widehat{\eta}^k_{\mathrm{osc},\ell})^2 \coloneqq \sum_{K\in\mathcal{T}_\ell}\left[\frac{h_K}{\pi \inf_K (A^{k-1}_{\mathrm{m},\ell})^{\frac12}}\|(I-\Pi_{\ell,p})f\|_K\right]^2, \tag{2.46a}$$

$$(\widehat{\eta}^k_{\mathrm{osc,q},\ell})^2 \coloneqq \sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\widehat{\eta}^{\boldsymbol{a},k}_{\mathrm{osc,q},\ell})^2 \quad\text{and}\quad (\widehat{\eta}^k_{\mathrm{q},\ell})^2 \coloneqq \sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\widehat{\eta}^{\boldsymbol{a},k}_{\mathrm{q},\ell})^2, \tag{2.46b}$$

where, for all vertices $\boldsymbol{a}\in\mathcal{V}_\ell$, recalling the notation from (2.25b),

$$(\widehat{\eta}^{\boldsymbol{a},k}_{\mathrm{osc,q},\ell})^2 \coloneqq \frac{1}{\inf_{\omega^{\boldsymbol{a}}_\ell} A^{k-1}_{\mathrm{m},\ell}}\sum_{K\in\mathcal{T}^{\boldsymbol{a}}_\ell}\left[\frac{h_K}{\pi}\|(I-\Pi_{\ell,p})\gamma^{\boldsymbol{a},k}_\ell\|_K\right]^2, \tag{2.46c}$$

$$\widehat{\eta}^{\boldsymbol{a},k}_{\mathrm{q},\ell} \coloneqq \|(\boldsymbol{A}^{k-1}_\ell)^{-\frac12}(\psi^{\boldsymbol{a}}_\ell(\boldsymbol{I}-\boldsymbol{\Pi}^{\boldsymbol{RTN}}_{\ell,p-1})\boldsymbol{\xi}^k_\ell)\|_{\omega^{\boldsymbol{a}}_\ell}. \tag{2.46d}$$

2.5.4 A posteriori estimate of the augmented energy difference

We now present our main result giving an a posteriori estimate based on the augmented energy difference and estimator defined in (2.42).

Theorem 2.5.5 (A posteriori estimate of the augmented energy difference)**.** *Suppose Assumption 2.2.1 and let u be the weak solution of (2.5). Let u^k_ℓ be its finite element approximation given by (2.9) on mesh $\mathcal{T}_\ell$, $\ell\ge 0$, and linearization step $k\ge 1$, for any iterative linearization satisfying Assumption 2.2.2. Let $\boldsymbol{\sigma}^k_\ell$ be the equilibrated flux defined by (2.25) with the choice (2.26b). Let $u^k_{\langle\ell\rangle}$ be given by (2.38), and the augmented error $\mathcal{E}^k_\ell$ and estimator η^k_ℓ by (2.42). Then*

$$\mathcal{E}^k_\ell \le \eta^k_\ell + \eta^k_{\mathrm{osc},\ell} + \frac{\lambda^k_\ell}{2}\widehat{\eta}^k_{\mathrm{osc},\ell}, \tag{2.47a}$$

$$\eta^k_\ell \lesssim \widehat{C}^k_\ell\mathcal{E}^k_\ell + \lambda^k_\ell(\widehat{C}^k_\ell\widehat{\eta}^k_{\mathrm{q},\ell} + \widehat{\eta}^k_{\mathrm{osc,q},\ell} + \widehat{\eta}^k_{\mathrm{iter},\ell}), \tag{2.47b}$$

where the hidden constant only depends on the space dimension d, the mesh shape-regularity $\kappa_{\mathcal{T}}$, and possibly, when $d\ge 4$, the polynomial degree p, with

$$\widehat{C}^k_\ell \coloneqq \max_{\boldsymbol{a}\in\mathcal{V}_\ell}\left(\frac{\sup_{\omega^{\boldsymbol{a}}_\ell} A^{k-1}_{\mathrm{c},\ell}}{\inf_{\omega^{\boldsymbol{a}}_\ell} A^{k-1}_{\mathrm{m},\ell}}\right)^{\frac12}\begin{cases} = 1 & \text{for the Zarantonello linearization,}\\ \le \left(\dfrac{A_{\mathrm{c}}}{A_{\mathrm{m}}}\right)^{\frac12} & \text{in general.}\end{cases} \tag{2.48}$$

Proof. See Section 2.10. □

Remark 2.5.6 (Robustness for the Zarantonello linearization)**.** *In the Zarantonello linearization of Example 2.2.5, since $\widehat{C}^k_\ell = 1$, we obtain via (2.47) an estimation of the augmented energy difference $\mathcal{E}^k_\ell$ by the estimator η^k_ℓ (under small oscillation, quadrature-type, and iterative linearization errors) whose quality is independent of the nonlinear function a. As we will see in the proof of Theorem 2.5.5, this relies on the fact that here $\boldsymbol{A}^{k-1}_\ell = \gamma\boldsymbol{I}_d$, the linearized problem (2.38) features a constant diffusion tensor, and consequently the associated error (2.39a) and estimator (2.39b) simplify respectively to $\mathcal{E}^k_{\mathrm{L},\ell} = \gamma^{\frac12}\|\boldsymbol{\nabla}(u^k_\ell - u^k_{\langle\ell\rangle})\|$ and $\eta^k_{\mathrm{L},\ell} = \|\gamma^{-\frac12}(\gamma\boldsymbol{\nabla}u^k_\ell - \boldsymbol{b}^{k-1}_\ell + \boldsymbol{\sigma}^k_\ell)\|$.*

Remark 2.5.7 (Robustness in the general case)**.** *The constant $\widehat{C}_\ell^k$ can be easily calculated in practice, without knowing the continuous solution u of (2.5). It is in particular defined from merely patchwise variations of the linearization matrix $\boldsymbol{A}_\ell^{k-1}$ (recall (2.11)). For the Picard and damped Newton iteration cases of Examples 2.2.4 and 2.2.6, in particular, the local ratio of the functions $A_{\mathrm{c},\ell}^{k-1}$ and $A_{\mathrm{m},\ell}^{k-1}$ is a lower bound for that of the global constants A_c and A_m given in (2.11c), typically bringing $\widehat{C}_\ell^k$ close to 1 as in the Zarantonello case. This is indeed observed in the numerical experiments of Section 2.6. Importantly, $\widehat{C}_\ell^k$ allows us to quantify the quality of the estimates in any situation: whenever it is small, we can affirm robustness a posteriori.*

Remark 2.5.8 (Theorem 2.5.5 at the convergence of iterative linearization / impact of the chosen iterative linearization)**.** *Suppose that $u_\ell^k \to u_\ell$ in $H_0^1(\Omega)$, i.e., the solution u_ℓ^k of the iterative linearization (2.9) converges to the solution u_ℓ of the discrete nonlinear problem (2.8a). In the limit, $\widehat{\eta}_{\mathrm{iter},\ell}^k$ from (2.40) vanishes and all the different iterative linearizations, cf. the examples of Section 2.2.3.2, yield the unique finite element approximation u_ℓ. Nevertheless, in Theorem 2.5.5, the augmented energy difference $\mathcal{E}_\ell^k$ from (2.42) is still (possibly) influenced by the chosen iterative linearization, in particular by the (limit) values of $\boldsymbol{A}_\ell^{k-1}$ and $\boldsymbol{b}_\ell^{k-1}$ as per (2.39), cf. also Remarks 2.5.2 and 2.5.3.*

2.6 Numerical results

In this section, we present numerical experiments that serve to illustrate Theorems 2.4.4 and 2.5.5. Thus, we will primarily be interested in the effectivity indices

$$I_{\mathrm{N},\ell}^k \coloneqq \frac{\eta_{\mathrm{N},\ell}^k}{\mathcal{E}_{\mathrm{N},\ell}^k}, \quad I_\ell^k \coloneqq \frac{\eta_\ell^k}{\mathcal{E}_\ell^k}, \quad I_{\mathrm{L},\ell}^k \coloneqq \frac{\eta_{\mathrm{L},\ell}^k}{\mathcal{E}_{\mathrm{L},\ell}^k}, \tag{2.49}$$

where the last one brings insight into the augmentation of Section 2.5.1. In particular, our numerical experiments study the robustness of our estimates with respect to the ratio $a_\mathrm{c}/a_\mathrm{m}$ from (2.4) where we consider $a_\mathrm{c}/a_\mathrm{m} = 10^i$, $i \in \{0, \dots, 7\}$. We present results for the three linearization methods of Examples 2.2.4, 2.2.5, and 2.2.6 after the convergence criterion

$$\|\boldsymbol{\nabla}(u_\ell^{\overline{k}-1} - u_\ell^{\overline{k}})\| < 10^{-6} \tag{2.50}$$

has been reached.

In addition to V_ℓ^p as defined in Section 2.2.2, we also consider a richer discrete space $V_{\tilde{\ell}}^{\tilde{p}}$ obtained by refining the mesh $\mathcal{T}_\ell$ and using higher-order polynomials, $\ell < \tilde{\ell}$ and $p < \tilde{p}$. This space serves as an approximation to $H_0^1(\Omega)$, so that we can approximately compute $u_{\langle\ell\rangle}^k$ defined in (2.38b) at each iteration k of the linearization method. This only serves here for the evaluation of the error $\mathcal{E}_{\mathrm{L},\ell}^k$; it is not needed to evaluate our estimators $\eta_{\mathrm{N},\ell}^k$, $\eta_{\mathrm{L},\ell}^k$, and η_ℓ^k.

For all examples, we use the method of manufactured solutions, i.e. we choose a solution u and construct f through (2.1). The boundary conditions are then enforced by the true solution. All experiments were conducted using the `Gridap.jl` finite element software package [7, 118].

2.6.1 Smooth solution

We consider a unit square domain $\Omega = (0,1)^2$. We set for all $(x,y) \in \Omega$,

$$u(x,y) \coloneqq 10\,x(x-1)y(y-1). \tag{2.51}$$

For the space V_ℓ^p, we use a polynomial degree $p = 1$ and a uniform triangular mesh consisting of 8192 elements for a total of 3969 DOFs. We consider three different nonlinear functions $a : [0,\infty) \to (0,\infty)$ (independent of the spatial coordinate $\boldsymbol{x} \in \Omega$) satisfying Assumption 2.2.1. We first consider the following example, in which the function a is monotone (decreasing).

Example 2.6.1 (Mean curvature nonlinearity)**.** *The mean curvature nonlinearity (cf. [40]) is defined such that for all $r \in [0,\infty)$,*

$$a(r) \coloneqq a_{\mathrm{m}} + \frac{a_{\mathrm{c}} - a_{\mathrm{m}}}{\sqrt{1+r^2}}, \tag{2.52}$$

where $a_{\mathrm{m}}, a_{\mathrm{c}} \in (0,\infty)$ with $a_{\mathrm{m}} \le a_{\mathrm{c}}$. Observe that Assumption 2.2.1 holds for the mean curvature nonlinearity. Indeed, we use Proposition 2.A.2 observing that, for all $r \in [0,\infty)$,

$$\phi''(r) = a_{\mathrm{m}} + \frac{a_{\mathrm{c}} - a_{\mathrm{m}}}{(1+r^2)^{\frac{3}{2}}} \in [a_{\mathrm{m}}, a_{\mathrm{c}}].$$

The results for the effectivity indices (2.49) are presented in Figure 2.2, taking $a_{\mathrm{m}} = 1$. We see that they vary only very mildly with respect to the ratio $a_{\mathrm{m}}/a_{\mathrm{c}}$ and that all values are below 1.2. This illustrates the robustness for the Zarantonello linearization that has been proven in Theorem 2.5.5. For the Newton and Picard linearizations, we also compute and display the constant $\widehat{C}_\ell^{\overline{k}}$ of (2.48). Since we see that $\widehat{C}_\ell^{\overline{k}}$ is uniformly close to 1 (bounded by 3), we are sure that the effectivity index I_ℓ^k will be small and uniformly bounded even prior to computing it. This is the meaning of "affirm robustness a posteriori" of Remark 2.5.7.

We now consider the following nonmonotone nonlinear function a similar to the example given in [70, Section 5.3.2].

Example 2.6.2 (Exponential nonlinearity)**.** *The exponential nonlinearity is defined such that for all $r \in [0,\infty)$,*

$$a(r) \coloneqq a_{\mathrm{m}} + (a_{\mathrm{c}} - a_{\mathrm{m}})\frac{1 - e^{-\frac{3}{2}r^2}}{1 + 2e^{-\frac{3}{2}}}, \tag{2.53}$$

where again $a_{\mathrm{m}}, a_{\mathrm{c}} \in (0,\infty)$ with $a_{\mathrm{m}} \le a_{\mathrm{c}}$. Observe that Assumption 2.2.1 holds for the exponential nonlinearity. Indeed, we again use Proposition 2.A.2 observing that, for all $r \in [0,\infty)$,

$$\phi''(r) = a_{\mathrm{m}} + (a_{\mathrm{c}} - a_{\mathrm{m}})\frac{1 + (3r^2 - 1)e^{-\frac{3}{2}r^2}}{1 + 2e^{-\frac{3}{2}}} \in [a_{\mathrm{m}}, a_{\mathrm{c}}].$$

The results, again with $a_{\mathrm{m}} = 1$, are presented in Figure 2.3. The Picard linearization is not included because the solver does not converge for large values of the ratio $a_{\mathrm{c}}/a_{\mathrm{m}}$. We observe that the results for the Zarantonello iteration are similar to those of Figure 2.2. However, the effectivity indices of the Newton iteration seem to start to deteriorate for large values of the ratio $a_{\mathrm{c}}/a_{\mathrm{m}}$. We can see that the reason is that the constant $\widehat{C}_\ell^{\overline{k}}$ is becoming very large. This constant is thus here a good indicator that the robustness may not be obtained; in this case, we cannot "affirm robustness a posteriori" and it indeed seems not to hold. For this example, we also present, in Figure 2.4, the component errors $\mathcal{E}_{\mathrm{N},\ell}^{\overline{k}}$ and $\lambda_\ell^{\overline{k}}\mathcal{E}_{\mathrm{L},\ell}^{\overline{k}}$, as well as the factor $\lambda_\ell^{\overline{k}}$, see (2.41)–(2.42). We observe that $\mathcal{E}_{\mathrm{N},\ell}^{\overline{k}}$ and $\lambda_\ell^{\overline{k}}\mathcal{E}_{\mathrm{L},\ell}^{\overline{k}}$ stay very close, independently of the ratio $a_{\mathrm{c}}/a_{\mathrm{m}}$, which was our design. Remark, though, that $\lambda_\ell^{\overline{k}} \simeq 1$ for Newton, whereas $\lambda_\ell^{\overline{k}} \simeq (a_{\mathrm{c}}/a_{\mathrm{m}})^{\frac{1}{2}}$ for Zarantonello.

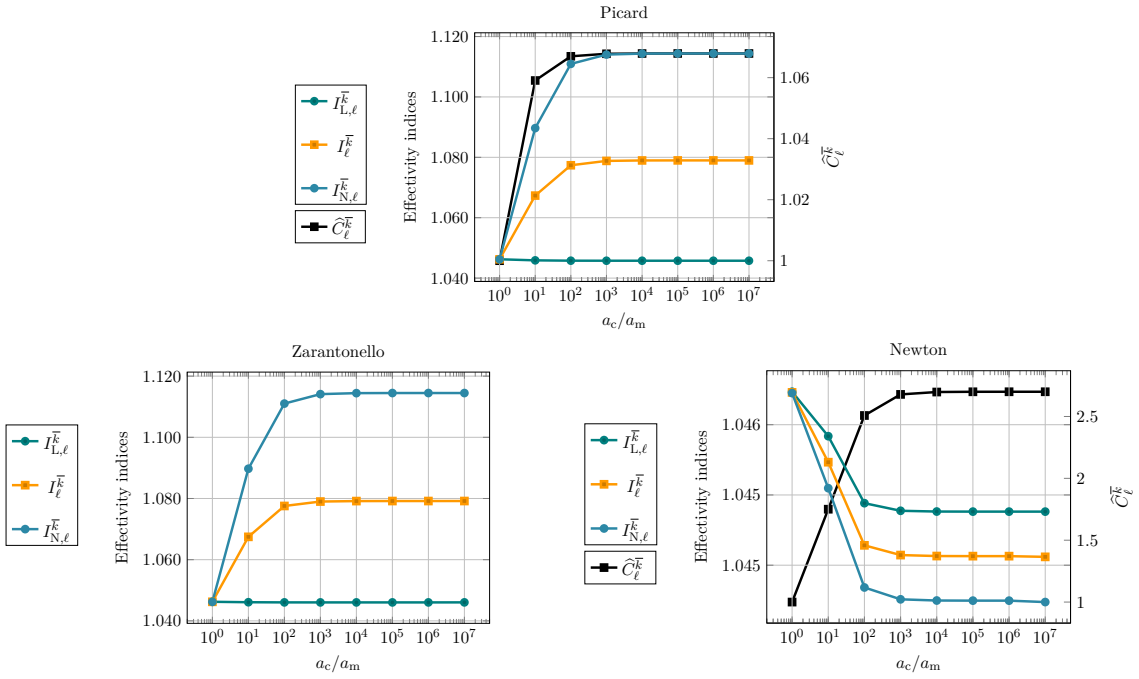


Figure 2.2: [Mean curvature nonlinearity (2.52), smooth solution (2.51), unit square domain, 3969 DOFs] Effectivity indices from (2.49) and the computable constant $\widehat{C}^{\overline{k}}_{\ell}$ from (2.48).

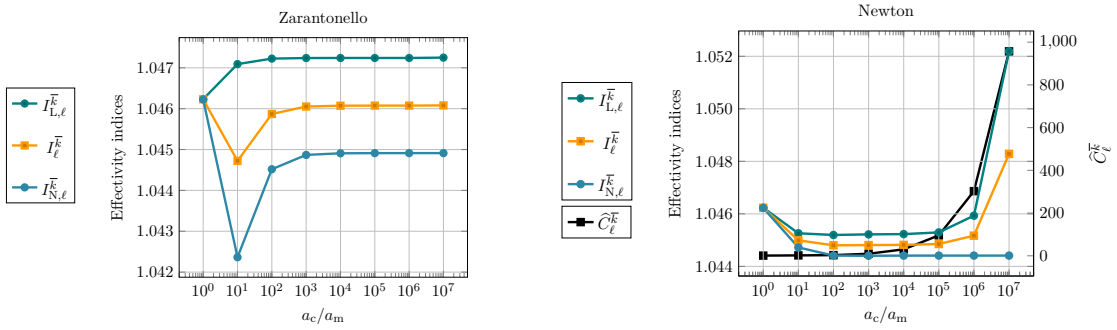


Figure 2.3: [Exponential nonlinearity (2.53), smooth solution (2.51), unit square domain, 3969 DOFs] Effectivity indices from (2.49) and the computable constant $\widehat{C}^{\overline{k}}_{\ell}$ from (2.48).

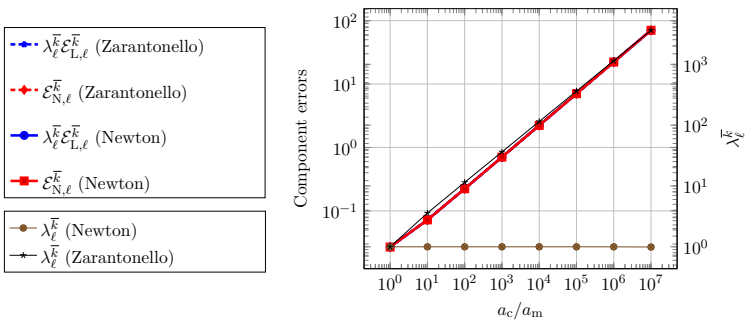
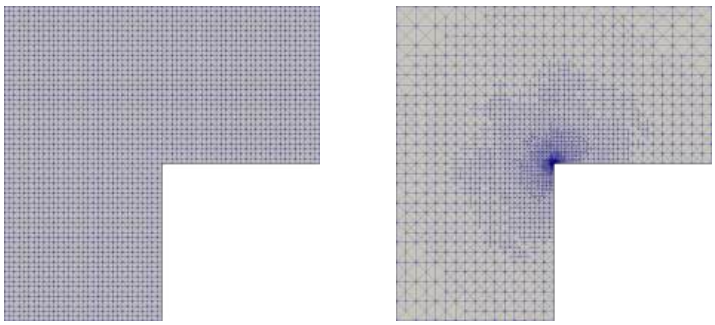


Figure 2.4: [Exponential nonlinearity (2.53), smooth solution (2.51), unit square domain, 3969 DOFs] Components $\mathcal{E}^{\overline{k}}_{N,\ell}$ and $\lambda^{\overline{k}}_{\ell}\mathcal{E}^{\overline{k}}_{L,\ell}$ from (2.42) together with the weight $\lambda^{\overline{k}}_{\ell}$ from (2.41).



(a) 2945 DOFs (b) 2417 DOFs

Figure 2.5: Uniformly (left) and adaptively (right) refined meshes for the L-shaped domain with the singular solution (2.54). The adaptive mesh corresponds to the 28th iteration of Algorithm 3.

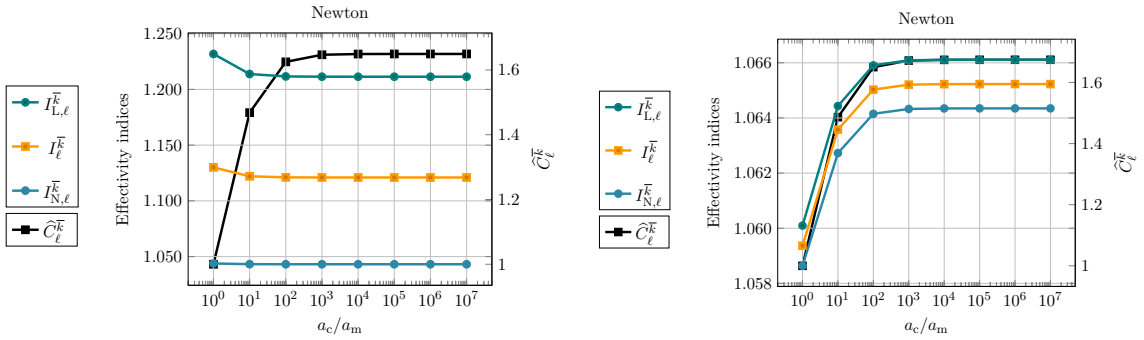


Figure 2.6: [Exponential nonlinearity (2.53), singular solution (2.54), L-shaped domain] Effectivity indices from (2.49) and the computable constant $\widehat{C}_\ell^k$ from (2.48), for the uniform mesh (left) and the adaptive mesh (right) shown in Figure 2.5.

2.6.2 Singular solution

We consider the L-shaped domain $\Omega = (-1,1)^2 \setminus ([0,1) \times (-1,0])$ and the singular solution u in polar coordinates $(\rho,\theta) \in [0,\infty) \times [0,2\pi)$

$$u(\rho,\theta) = \rho^\alpha \sin(\alpha\theta) \tag{2.54}$$

with $\alpha := \frac{2}{3}$, so that $u \in H^{1+\frac{2}{3}-\varepsilon}(\Omega)$ for all $\varepsilon > 0$. We consider the exponential nonlinearity of Example 2.6.2 again with $a_m = 1$; this choice of solution ensures that the right-hand side f belongs to $L^2(\Omega)$ despite the singularity in the norm of the gradient for the L-shaped solution (2.54).

We consider two different meshes to analyze the results, see Figure 2.5. One mesh is obtained by taking an initial uniform triangulation of Ω, while the other one is adaptive following Algorithm 3.

The results are presented in Figure 2.6. The Newton iteration showcases, for both meshes, effectivity indices close to 1, which stabilize for large enough values of the ratio a_m/a_c. Moreover, the effectivity indices corresponding to the adaptive meshes are closer to 1 than those corresponding to the uniform meshes. Since $\widehat{C}_\ell^k$ takes small values below 2, we can again claim robustness a posteriori, see Remark 2.5.7.

2.6.3 Convergence on a sequence of adaptively refined meshes

Due to the singularity in the solution of the previous section, it is of interest to consider a local adaptive mesh refinement strategy. We thus need to compute the contribution of the estimator $\eta^k_{\mathrm{N},\ell}$ to each mesh element which is nonnegative. In order to do that, cf. the discussion in, e.g., [13, Proposition 4.9], we use (2.6), (2.20), and (2.28) to rewrite the estimator $\eta^k_{\mathrm{N},\ell}$ of (2.29b) as

$$\eta^k_{\mathrm{N},\ell} = \left(2\int_\Omega \left(\phi^*(\cdot,|\boldsymbol{\sigma}^k_\ell|)+\phi(\cdot,|\boldsymbol{\nabla} u^k_\ell|)+\boldsymbol{\sigma}^k_\ell\cdot\boldsymbol{\nabla} u^k_\ell\right)\right)^{\frac{1}{2}} = \left(\sum_{K\in\mathcal{T}_\ell}(\eta^k_{\mathrm{N},K})^2\right)^{\frac{1}{2}}, \tag{2.55a}$$

$$(\eta^k_{\mathrm{N},K})^2 := 2\int_K \left(\phi^*(\cdot,|\boldsymbol{\sigma}^k_\ell|)+\phi(\cdot,|\boldsymbol{\nabla} u^k_\ell|)+\boldsymbol{\sigma}^k_\ell\cdot\boldsymbol{\nabla} u^k_\ell\right), \quad K\in\mathcal{T}_\ell. \tag{2.55b}$$

Here, recalling the generalized Young inequality, cf. [72],

$$\phi(\cdot,|\boldsymbol{x}|)+\phi^*(\cdot,|\boldsymbol{y}|)+\boldsymbol{x}\cdot\boldsymbol{y}\geq 0 \qquad \forall \boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d, \tag{2.56}$$

it follows that for all $K\in\mathcal{T}_\ell$, indeed $\eta^k_{\mathrm{N},K}\geq 0$. We then use the standard newest vertex bisection algorithm, see [28] and the references therein, as follows.

Algorithm 3: Adaptive refinement

1 Let $\varepsilon_{\mathrm{STOP}}$ and $\theta\in(0,1)$ be parameters, and let $\mathcal{T}_0$ be a conforming initial triangulation of Ω. Let $u^0_0\in V^1_0$ be an initial linearization guess. For $\ell\geq 0$:

1. **Solve**: Starting from u^0_ℓ, solve the linearized problems (2.9) until the convergence criterion (2.50) is satisfied.
2. **Estimate**: Compute the elementwise estimators $(\eta^{\bar{k}}_{\mathrm{N},K})_{K\in\mathcal{T}_\ell}$ of (2.55b). If $\eta^{\bar{k}}_{\mathrm{N},\ell}<\varepsilon_{\mathrm{STOP}}$, then stop.
3. **Mark**: Choose a set $\mathcal{M}_\ell\subset\mathcal{T}_\ell$ with minimal cardinality such that
$$\sum_{K\in\mathcal{M}_\ell}(\eta^{\bar{k}}_{\mathrm{N},K})^2\geq\theta^2\sum_{K\in\mathcal{T}_\ell}(\eta^{\bar{k}}_{\mathrm{N},K})^2. \tag{2.57}$$
4. **Refine**: Perform the newest vertex bisection on the set $\mathcal{M}_\ell$. Set $\ell:=\ell+1$, $u^0_\ell:=u^{\bar{k}}_{\ell-1}$, and go to step 1.

The results of the refinement study are displayed in Figure 2.7, for the exponential nonlinearity (2.53) and the singular solution (2.54). We consider two values of the parameter $a_{\mathrm{c}}/a_{\mathrm{m}}$, namely 10^3 and 10^6. We note that for both values of the ratio, the asymptotic rates for the estimator and error agree with the theoretical optimal rate of $(\mathrm{DOFs})^{-1/2}$ for the adaptive algorithm; we observe no distinguishable difference in this graphic representation between $a_{\mathrm{c}}/a_{\mathrm{m}}=10^3$ and 10^6.

We also made an analogous study on the augmented error $\mathcal{E}^{\bar{k}}_\ell$ and the estimator $\eta^{\bar{k}}_\ell$, with the same strategy of refinement, using the local version $\eta^{\bar{k}}_K:=\eta^{\bar{k}}_{\mathrm{N},K}+\lambda^{\bar{k}}_\ell\eta^{\bar{k}}_{\mathrm{L},K}$, for all $K\in\mathcal{T}_\ell$, of the estimator $\eta^{\bar{k}}_\ell$. The results are displayed in Figure 2.8. We observe a similar behavior of the asymptotic rates as in Figure 2.7.

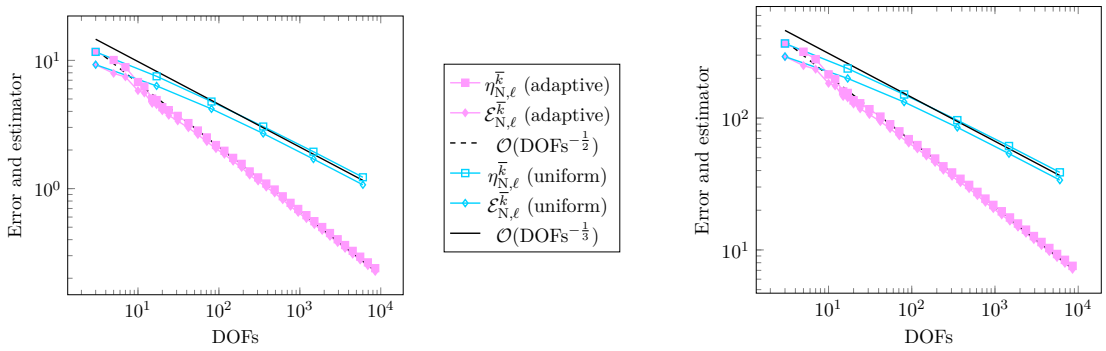


Figure 2.7: [Exponential nonlinearity (2.53), singular solution (2.54), L-shaped domain, uniform vs adaptive mesh refinement] Convergence rates of $\mathcal{E}^{\overline{k}}_{\mathrm{N},\ell}$ and $\eta^{\overline{k}}_{\mathrm{N},\ell}$ for uniform and adaptive mesh refinement for $a_{\mathrm{c}}/a_{\mathrm{m}}$ equal to 10^3 (left) and 10^6 (right).

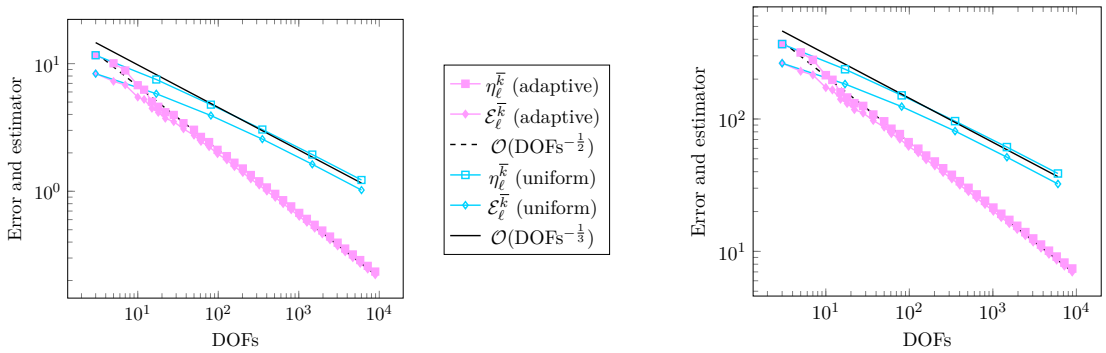


Figure 2.8: [Exponential nonlinearity (2.53), singular solution (2.54), L-shaped domain, uniform vs adaptive mesh refinement] Convergence rates of $\mathcal{E}^{\overline{k}}_{\ell}$ and $\eta^{\overline{k}}_{\ell}$ for uniform and adaptive mesh refinement for $a_{\mathrm{c}}/a_{\mathrm{m}}$ equal to 10^3 (left) and 10^6 (right).

In conclusion, the adaptive mesh refinement is more efficient than the uniform mesh refinement since it requires a smaller number of DOFs for the same precision. The behavior seems to be independent of the strength of the nonlinearity $a_{\mathrm{c}}/a_{\mathrm{m}}$.

2.7 Proof of Lemma 2.4.1

Proof of Lemma 2.4.1. Observing that by integration by parts (IBP), $\frac{\partial}{\partial r}((|\boldsymbol{\sigma}^k_\ell|-r)\phi^{*\prime}(\cdot,r)) = (|\boldsymbol{\sigma}^k_\ell|-r)\phi^{*\prime\prime}(\cdot,r) - \phi^{*\prime}(\cdot,r)$, we obtain, a.e. in Ω,

$$\begin{aligned}
\phi^*(\cdot,|\boldsymbol{\sigma}^k_\ell|) - \phi^*(\cdot,a^k_\ell|\boldsymbol{\nabla} u^k_\ell|) &= \int_{a^k_\ell|\boldsymbol{\nabla} u^k_\ell|}^{|\boldsymbol{\sigma}^k_\ell|} \phi^{*\prime}(\cdot,r)\,\mathrm{d}r \\
&\overset{\text{(IBP)}}{=} \int_{a^k_\ell|\boldsymbol{\nabla} u^k_\ell|}^{|\boldsymbol{\sigma}^k_\ell|} (|\boldsymbol{\sigma}^k_\ell|-r)\phi^{*\prime\prime}(\cdot,r)\,\mathrm{d}r - \Big[(|\boldsymbol{\sigma}^k_\ell|-r)\phi^{*\prime}(\cdot,r)\Big]_{a^k_\ell|\boldsymbol{\nabla} u^k_\ell|}^{|\boldsymbol{\sigma}^k_\ell|} \\
&\overset{(2.22)}{=} \int_{a^k_\ell|\boldsymbol{\nabla} u^k_\ell|}^{|\boldsymbol{\sigma}^k_\ell|} \frac{|\boldsymbol{\sigma}^k_\ell|-r}{\phi''(\cdot,\phi'^{-1}(\cdot,r))}\,\mathrm{d}r + (|\boldsymbol{\sigma}^k_\ell| - a^k_\ell|\boldsymbol{\nabla} u^k_\ell|)|\boldsymbol{\nabla} u^k_\ell|,
\end{aligned} \tag{2.58}$$

where we used the fact that $\phi'^{-1}(\cdot,a^k_\ell|\boldsymbol{\nabla} u^k_\ell|) = \phi'^{-1}(\cdot,\phi'(\cdot,|\boldsymbol{\nabla} u^k_\ell|)) = |\boldsymbol{\nabla} u^k_\ell|$ thanks to (2.84). Reusing this relation together with the definitions (2.6) of $\mathcal{J}$ and (2.20) of $\mathcal{J}^*$, we obtain

the right-hand side of (2.32b) by writing

$$
\begin{aligned}
(\eta_{\mathrm{N},\ell}^k)^2 &\overset{(2.29b)}{=} 2\int_\Omega \Big(\phi^*(\cdot,|\boldsymbol{\sigma}_\ell^k|)+\phi(\cdot,|\boldsymbol{\nabla} u_\ell^k|)\Big) - (f,u_\ell^k)\\
&\overset{(2.28)}{=} 2\int_\Omega \Big(\phi^*(\cdot,|\boldsymbol{\sigma}_\ell^k|)+\phi(\cdot,|\boldsymbol{\nabla} u_\ell^k|)+\boldsymbol{\sigma}_\ell^k\cdot\boldsymbol{\nabla} u_\ell^k\Big)\\
&\overset{(2.21b)}{=} 2\int_\Omega \Big(\phi^*(\cdot,|\boldsymbol{\sigma}_\ell^k|)-\phi^*(\cdot,a_\ell^k|\boldsymbol{\nabla} u_\ell^k|)+a_\ell^k|\boldsymbol{\nabla} u_\ell^k|^2+\boldsymbol{\sigma}_\ell^k\cdot\boldsymbol{\nabla} u_\ell^k\Big)\\
&\overset{(2.58)}{=} 2\int_\Omega \left[\int_{a_\ell^k|\boldsymbol{\nabla} u_\ell^k|}^{|\boldsymbol{\sigma}_\ell^k|}\frac{|\boldsymbol{\sigma}_\ell^k|-r}{\phi''(\cdot,\phi'^{-1}(\cdot,r))}\,\mathrm{d}r+\frac{1}{a_\ell^k}(|\boldsymbol{\sigma}_\ell^k||a_\ell^k\boldsymbol{\nabla} u_\ell^k|+\boldsymbol{\sigma}_\ell^k\cdot(a_\ell^k\boldsymbol{\nabla} u_\ell^k))\right]\\
&\overset{(2.31c)}{\le} \int_\Omega \frac{2}{a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}\left(\left[-\frac{1}{2}(|\boldsymbol{\sigma}_\ell^k|-r)^2\right]_{a_\ell^k|\boldsymbol{\nabla} u_\ell^k|}^{|\boldsymbol{\sigma}_\ell^k|}+|\boldsymbol{\sigma}_\ell^k||a_\ell^k\boldsymbol{\nabla} u_\ell^k|+\boldsymbol{\sigma}_\ell^k\cdot(a_\ell^k\boldsymbol{\nabla} u_\ell^k)\right)\\
&= \int_\Omega \frac{1}{a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}\left[(|\boldsymbol{\sigma}_\ell^k|-a_\ell^k|\boldsymbol{\nabla} u_\ell^k|)^2+2(|\boldsymbol{\sigma}_\ell^k||a_\ell^k\boldsymbol{\nabla} u_\ell^k|+\boldsymbol{\sigma}_\ell^k\cdot(a_\ell^k\boldsymbol{\nabla} u_\ell^k))\right]\\
&\overset{(2.86a)}{=} \int_\Omega \frac{1}{a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}|a_\ell^k\boldsymbol{\nabla} u_\ell^k+\boldsymbol{\sigma}_\ell^k|^2,
\end{aligned}
\tag{2.59}
$$

where, in using (2.21b), we have set $s = a_\ell^k|\boldsymbol{\nabla} u_\ell^k|$, and where we used (2.31c) noticing that $|\boldsymbol{\sigma}_\ell^k||a_\ell^k\boldsymbol{\nabla} u_\ell^k| + \boldsymbol{\sigma}_\ell^k\cdot(a_\ell^k\boldsymbol{\nabla} u_\ell^k) \ge 0$ thanks to the Cauchy–Schwarz inequality, together with the fact that $|\boldsymbol{\sigma}_\ell^k| - r$ is of the same sign as $|\boldsymbol{\sigma}_\ell^k| - a_\ell^k|\boldsymbol{\nabla} u_\ell^k|$ for all $r \in (a_\ell^k|\boldsymbol{\nabla} u_\ell^k|, |\boldsymbol{\sigma}_\ell^k|)$. The left-hand side of (2.32b) is obtained with the same reasoning.

Now, observing that $\frac{\partial}{\partial r}((|\boldsymbol{\nabla} u_\ell^k| - r)\phi'(\cdot,r)) = (|\boldsymbol{\nabla} u_\ell^k| - r)\phi''(\cdot,r) - \phi'(\cdot,r)$ by integration by parts (IBP), we get

$$
\begin{aligned}
(\mathcal{E}_{\mathrm{N},\ell}^k)^2 &\overset{(2.29a),(2.6)}{=} 2\int_\Omega (\phi(\cdot,|\boldsymbol{\nabla} u_\ell^k|)-\phi(\cdot,|\boldsymbol{\nabla} u|)) - 2(f,u_\ell^k-u)\\
&\overset{(2.5a)}{=} 2\int_\Omega\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}\phi'(\cdot,r)\,\mathrm{d}r - 2(a_u\boldsymbol{\nabla} u,\boldsymbol{\nabla}(u_\ell^k-u))\\
&\overset{(\mathrm{IBP})}{=} 2\int_\Omega\left(\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}(|\boldsymbol{\nabla} u_\ell^k|-r)\phi''(\cdot,r)\,\mathrm{d}r-\Big[(|\boldsymbol{\nabla} u_\ell^k|-r)\phi'(\cdot,r)\Big]_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}\right)\\
&\qquad - 2(a_u\boldsymbol{\nabla} u,\boldsymbol{\nabla}(u_\ell^k-u))\\
&\overset{(2.84)}{=} 2\int_\Omega\left(\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}(|\boldsymbol{\nabla} u_\ell^k|-r)\phi''(\cdot,r)\,\mathrm{d}r+a_u(|\boldsymbol{\nabla} u_\ell^k||\boldsymbol{\nabla} u|-\boldsymbol{\nabla} u_\ell^k\cdot\boldsymbol{\nabla} u)\right).
\end{aligned}
\tag{2.60}
$$

From there, with the same reasoning as for (2.32b), we obtain the right-hand side of (2.32a), by writing

$$
\begin{aligned}
(\mathcal{E}_{\mathrm{N},\ell}^k)^2 &\overset{(2.60),(2.31c)}{\le} \int_\Omega a_{\mathrm{c},\ell}^{u,k}\Big((|\boldsymbol{\nabla} u_\ell^k|-|\boldsymbol{\nabla} u|)^2+2(|\boldsymbol{\nabla} u_\ell^k||\boldsymbol{\nabla} u|-\boldsymbol{\nabla} u_\ell^k\cdot\boldsymbol{\nabla} u)\Big)\\
&\overset{(2.86a)}{=} \int_\Omega a_{\mathrm{c},\ell}^{u,k}|\boldsymbol{\nabla} u_\ell^k-\boldsymbol{\nabla} u|^2.
\end{aligned}
\tag{2.61}
$$

The left-hand side of (2.32a) is obtained with the same reasoning. Moreover, since ϕ' is nondecreasing and $|\boldsymbol{\nabla} u_\ell^k| - r$ is of the same sign as $|\boldsymbol{\nabla} u_\ell^k| - |\boldsymbol{\nabla} u|$ for all $r \in (|\boldsymbol{\nabla} u|, |\boldsymbol{\nabla} u_\ell^k|)$,

using the mean value inequality, we have a.e. in Ω,

$$\begin{aligned}\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}(|\boldsymbol{\nabla} u_\ell^k|-r)\phi''(\cdot,r)\,\mathrm{d}r &\geq \int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}\frac{\phi'(\cdot,|\boldsymbol{\nabla} u_\ell^k|)-\phi'(\cdot,r)}{\operatorname*{ess\,sup}_{s\in(r,|\boldsymbol{\nabla} u_\ell^k|)}\phi''(\cdot,s)}\phi''(\cdot,r)\,\mathrm{d}r\\ &\overset{(2.31c)}{\geq} \frac{1}{a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}}\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}(\phi'(\cdot,|\boldsymbol{\nabla} u_\ell^k|)-\phi'(\cdot,r))\phi''(\cdot,r)\,\mathrm{d}r\\ &= \frac{1}{a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}}\left[\phi'(\cdot,|\boldsymbol{\nabla} u_\ell^k|)\phi'(\cdot,r)-\frac{1}{2}\phi'(\cdot,r)^2\right]_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}\\ &\overset{(2.84)}{=} \frac{1}{a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}}\left(\frac{1}{2}\left(|a_\ell^k\boldsymbol{\nabla} u_\ell^k|^2+|a_u\boldsymbol{\nabla} u|^2\right)-a_\ell^k a_u|\boldsymbol{\nabla} u_\ell^k||\boldsymbol{\nabla} u|\right),\end{aligned} \tag{2.62}$$

where we used (2.31c) knowing that $|a_u\boldsymbol{\nabla} u|=\phi'(\cdot,|\boldsymbol{\nabla} u|)$ and $|a_\ell^k\boldsymbol{\nabla} u_\ell^k|=\phi'(\cdot,|\boldsymbol{\nabla} u_\ell^k|)$. Thus, observing that $|\boldsymbol{\nabla} u_\ell^k||\boldsymbol{\nabla} u|-\boldsymbol{\nabla} u_\ell^k\cdot\boldsymbol{\nabla} u\geq 0$ thanks to the Cauchy–Schwarz inequality, we obtain the left-hand side of (2.32c) by writing

$$\begin{aligned}(\mathcal{E}_{\mathrm{N},\ell}^k)^2 &\overset{(2.60),(2.62),(2.31c)}{\geq} \int_\Omega \frac{2}{a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}}\left(\frac{1}{2}\left(|a_\ell^k\boldsymbol{\nabla} u_\ell^k|^2+|a_u\boldsymbol{\nabla} u|^2\right)-(a_\ell^k\boldsymbol{\nabla} u_\ell^k)\cdot(a_u\boldsymbol{\nabla} u)\right)\\ &= \int_\Omega \frac{1}{a_{\mathrm{c},\ell}^{\boldsymbol{\nabla} u,k}}|a_\ell^k\boldsymbol{\nabla} u_\ell^k-a_u\boldsymbol{\nabla} u|^2.\end{aligned} \tag{2.63}$$

Finally, since $|\boldsymbol{\nabla} u_\ell^k|-r$ is of the same sign as $|\boldsymbol{\nabla} u_\ell^k|-|\boldsymbol{\nabla} u|$ for all $r\in(|\boldsymbol{\nabla} u|,|\boldsymbol{\nabla} u_\ell^k|)$, we have

$$\begin{aligned}\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}(|\boldsymbol{\nabla} u_\ell^k|-r)\phi''(\cdot,r)\,\mathrm{d}r &\leq (|\boldsymbol{\nabla} u_\ell^k|-|\boldsymbol{\nabla} u|)\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}\phi''(\cdot,r)\,\mathrm{d}r\\ &\overset{(2.31c)}{\leq} \frac{1}{a_{\mathrm{m},\ell}^{\boldsymbol{\nabla} u,k}}\left(\int_{|\boldsymbol{\nabla} u|}^{|\boldsymbol{\nabla} u_\ell^k|}\phi''(\cdot,r)\,\mathrm{d}r\right)^2 \overset{(2.84)}{=} \frac{1}{a_{\mathrm{m},\ell}^{\boldsymbol{\nabla} u,k}}\left(|a_\ell^k\boldsymbol{\nabla} u_\ell^k|-|a_u\boldsymbol{\nabla} u|\right)^2.\end{aligned} \tag{2.64}$$

Hence, observing again that $|\boldsymbol{\nabla} u_\ell^k||\boldsymbol{\nabla} u|-\boldsymbol{\nabla} u_\ell^k\cdot\boldsymbol{\nabla} u\geq 0$ thanks to the Cauchy–Schwarz inequality, we obtain the right-hand side of (2.32c) by writing

$$\begin{aligned}(\mathcal{E}_{\mathrm{N},\ell}^k)^2 &\overset{(2.60),(2.64),(2.31c)}{\leq} \int_\Omega \frac{2}{a_{\mathrm{m},\ell}^{\boldsymbol{\nabla} u,k}}\Big(\left(|a_\ell^k\boldsymbol{\nabla} u_\ell^k|-|a_u\boldsymbol{\nabla} u|\right)^2+2(|a_\ell^k\boldsymbol{\nabla} u_\ell^k||a_u\boldsymbol{\nabla} u|\\ &\quad-(a_\ell^k\boldsymbol{\nabla} u_\ell^k)\cdot(a_u\boldsymbol{\nabla} u))\Big) \overset{(2.86a)}{=} \int_\Omega \frac{2}{a_{\mathrm{m},\ell}^{\boldsymbol{\nabla} u,k}}|a_\ell^k\boldsymbol{\nabla} u_\ell^k-a_u\boldsymbol{\nabla} u|^2.\end{aligned} \tag{2.65}$$

□

2.8 Proof of Theorem 2.4.4

For all vertices $\boldsymbol{a}\in\mathcal{V}_\ell$, we introduce the space $H_*^1(\omega_\ell^{\boldsymbol{a}})$ such that

$$H_*^1(\omega_\ell^{\boldsymbol{a}}) := \begin{cases}\{\varphi\in H^1(\omega_\ell^{\boldsymbol{a}}) : (\varphi,1)_{\omega_\ell^{\boldsymbol{a}}}=0\} & \text{if } \boldsymbol{a}\in\mathcal{V}_\ell^{\mathrm{int}},\\ \{\varphi\in H^1(\omega_\ell^{\boldsymbol{a}}) : \varphi_{|\partial\omega_\ell^{\boldsymbol{a}}\cap\{\psi_\ell^{\boldsymbol{a}}>0\}}=0\} & \text{if } \boldsymbol{a}\notin\mathcal{V}_\ell^{\mathrm{int}},\end{cases} \tag{2.66}$$

where $\mathcal{V}_\ell^{\mathrm{int}}$ is the set of the vertices of $\mathcal{T}_\ell$ lying inside Ω. We will need:

Lemma 2.8.1 (Patchwise flux equilibration stability)**.** *Let $\boldsymbol{a} \in \mathcal{V}_\ell$ and let $\boldsymbol{\tau}_\ell^{\boldsymbol{a}} \in \boldsymbol{RTN}_p(\mathcal{T}_\ell^{\boldsymbol{a}})$ and $g_\ell^{\boldsymbol{a}} \in \mathbb{P}_p(\mathcal{T}_\ell^{\boldsymbol{a}})$ be such that $(g_\ell^{\boldsymbol{a}}, 1)_{\omega_\ell^{\boldsymbol{a}}} = 0$ if $\boldsymbol{a} \in \mathcal{V}_\ell^{\mathrm{int}}$. Then,*

$$\min_{\substack{\boldsymbol{v}_\ell^{\boldsymbol{a}} \in \boldsymbol{V}_\ell^{\boldsymbol{a}} \\ \boldsymbol{\nabla}\cdot\boldsymbol{v}_\ell^{\boldsymbol{a}} = g_\ell^{\boldsymbol{a}}}} \|\boldsymbol{v}_\ell^{\boldsymbol{a}} + \boldsymbol{\tau}_\ell^{\boldsymbol{a}}\|_{\omega_\ell^{\boldsymbol{a}}} \lesssim \sup_{\substack{\varphi \in H^1_*(\omega_\ell^{\boldsymbol{a}}) \\ \|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}} = 1}} \left[(g_\ell^{\boldsymbol{a}}, \varphi)_{\omega_\ell^{\boldsymbol{a}}} - (\boldsymbol{\tau}_\ell^{\boldsymbol{a}}, \boldsymbol{\nabla}\varphi)_{\omega_\ell^{\boldsymbol{a}}} \right], \tag{2.67}$$

where the hidden constant only depends on the space dimension d, the mesh shape-regularity $\kappa_{\mathcal{T}}$, and possibly, when $d \geq 4$, the polynomial degree p.

Proof. See [54, Corollary 3.3] or [52, Lemma 3.2]. □

We will use next the fact that, for all $\boldsymbol{v} \in \boldsymbol{L}^2(\Omega)$ and $\boldsymbol{W} \in [L^2(\Omega)]^{d\times d}$,

$$\sum_{a\in\mathcal{V}_\ell} \|\boldsymbol{v}\|^2_{\omega_\ell^{\boldsymbol{a}}} = \sum_{K\in\mathcal{T}_\ell} \sum_{a\in\mathcal{V}_K} \|\boldsymbol{v}\|_K^2 = (d+1) \sum_{K\in\mathcal{T}_\ell} \|\boldsymbol{v}\|_K^2 = (d+1)\|\boldsymbol{v}\|^2, \tag{2.68}$$

since every simplex $K \in \mathcal{T}_\ell$ has $(d+1)$ vertices, collected in the set $\mathcal{V}_K$. Also,

$$\begin{aligned} \|\boldsymbol{v} + \boldsymbol{W}\boldsymbol{\sigma}_\ell^k\|^2 &= \sum_{K\in\mathcal{T}_\ell} \|\boldsymbol{v} + \boldsymbol{W}\boldsymbol{\sigma}_\ell^k\|_K^2 \overset{(2.24),(2.25a)}{=} \sum_{K\in\mathcal{T}_\ell} \Big\| \sum_{a\in\mathcal{V}_K} (\psi_\ell^{\boldsymbol{a}}\boldsymbol{v} + \boldsymbol{W}\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}) \Big\|_K^2 \\ &\leq (d+1) \sum_{K\in\mathcal{T}_\ell} \sum_{a\in\mathcal{V}_K} \|\psi_\ell^{\boldsymbol{a}}\boldsymbol{v} + \boldsymbol{W}\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}\|_K^2 = (d+1) \sum_{a\in\mathcal{V}_\ell} \|\psi_\ell^{\boldsymbol{a}}\boldsymbol{v} + \boldsymbol{W}\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}\|^2_{\omega_\ell^{\boldsymbol{a}}}. \end{aligned} \tag{2.69}$$

In the following, for any $x, y \in [0,\infty)$, we use the notation $x \lesssim y$ when $x \leq Cy$ with $C \geq 0$ only depending on the space dimension d, the mesh shape-regularity $\kappa_{\mathcal{T}}$, and possibly, when $d \geq 4$, the polynomial degree p.

Proof of Theorem 2.4.4. As in [10, 107, 129] but including data oscillations, we have

$$\begin{aligned} (\mathcal{E}_{\mathrm{N},\ell}^k)^2 &\overset{(2.29)}{=} (\eta_{\mathrm{N},\ell}^k)^2 - 2\int_\Omega (\phi^*(|\boldsymbol{\sigma}_\ell^k|) + \phi(|\boldsymbol{\nabla}u|) - fu) \\ &\overset{(2.21a)}{\leq} (\eta_{\mathrm{N},\ell}^k)^2 - 2\int_\Omega (|\boldsymbol{\sigma}_\ell^k||\boldsymbol{\nabla}u| - fu) \leq (\eta_{\mathrm{N},\ell}^k)^2 + 2((f,u) + (\boldsymbol{\sigma}_\ell^k, \boldsymbol{\nabla}u)) \\ &\overset{(2.28)}{=} (\eta_{\mathrm{N},\ell}^k)^2 + 2(f - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}_\ell^k, u - u_\ell^k) \\ &\overset{(2.27)}{\leq} (\eta_{\mathrm{N},\ell}^k)^2 + 2\left(\sum_{K\in\mathcal{T}_\ell} \left[\frac{h_K}{\pi} \|f - \Pi_{\ell,p} f\|_K \right]^2 \right)^{\frac12} \|\boldsymbol{\nabla}(u - u_\ell^k)\| \\ &\overset{(2.76),(2.33a)}{\leq} (\eta_{\mathrm{N},\ell}^k)^2 + 2\eta_{\mathrm{osc},\ell}^k \mathcal{E}_{\mathrm{N},\ell}^k. \end{aligned} \tag{2.70}$$

By the quadratic formula, this gives

$$\mathcal{E}_{\mathrm{N},\ell}^k \leq \frac12 \left(2\eta_{\mathrm{osc},\ell}^k + (4(\eta_{\mathrm{osc},\ell}^k)^2 + 4(\eta_{\mathrm{N},\ell}^k)^2)^{\frac12} \right) \leq \eta_{\mathrm{N},\ell}^k + 2\eta_{\mathrm{osc},\ell}^k, \tag{2.71}$$

hence (2.35a).

It remains to prove (2.35b). Let us temporarily denote $\varsigma := 1/(\operatorname{ess\,inf}_{\omega_\ell^{\boldsymbol{a}}} a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k})^{\frac12}$ the weight appearing in (2.33c)–(2.33d). We apply the triangle inequality, the definitions (2.33c)–(2.33d), the fact that $0 \leq \psi_\ell^{\boldsymbol{a}} \leq 1$, the definition (2.25b) with the choice (2.26a), and Lemma 2.8.1 with $\boldsymbol{\tau}_\ell^{\boldsymbol{a}} = \psi_\ell^{\boldsymbol{a}} \boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}} \boldsymbol{\xi}_\ell^k \in \boldsymbol{RTN}_p(\mathcal{T}_\ell^{\boldsymbol{a}})$ and $g_\ell^{\boldsymbol{a}} = \Pi_{\ell,p}\gamma_\ell^{\boldsymbol{a},k} \in \mathbb{P}_p(\mathcal{T}_\ell^{\boldsymbol{a}})$

with $(g_\ell^{\boldsymbol{a}},1)_{\omega_\ell^{\boldsymbol{a}}}=0$ if $\boldsymbol{a}\in\mathcal{V}_\ell^{\mathrm{int}}$. Also using the fact that $\psi_\ell^{\boldsymbol{a}}\varphi\in H_0^1(\omega_\ell^{\boldsymbol{a}})$ and $\boldsymbol{\nabla}(\psi_\ell^{\boldsymbol{a}}\varphi)=\psi_\ell^{\boldsymbol{a}}\boldsymbol{\nabla}\varphi+\varphi\boldsymbol{\nabla}\psi_\ell^{\boldsymbol{a}}$ for all $\varphi\in H_*^1(\omega_\ell^{\boldsymbol{a}})$, we get

$$
\begin{aligned}
\varsigma\|\psi_\ell^{\boldsymbol{a}}a_\ell^k\boldsymbol{\nabla}u_\ell^k+\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}\|_{\omega_\ell^{\boldsymbol{a}}} &\overset{(2.33)}{\leq} \varsigma\|\psi_\ell^{\boldsymbol{a}}\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}}\boldsymbol{\xi}_\ell^k+\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}\|_{\omega_\ell^{\boldsymbol{a}}}+\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}+\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k}\\
&\overset{(2.25b),(2.67)}{\lesssim} \varsigma\sup_{\substack{\varphi\in H_*^1(\omega_\ell^{\boldsymbol{a}})\\ \|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}}=1}}\Big[(\Pi_{\ell,p}\gamma_\ell^{\boldsymbol{a},k},\varphi)_{\omega_\ell^{\boldsymbol{a}}}-(\psi_\ell^{\boldsymbol{a}}\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}}\boldsymbol{\xi}_\ell^k,\boldsymbol{\nabla}\varphi)_{\omega_\ell^{\boldsymbol{a}}}\Big]+\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}+\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k}\\
&\overset{(2.25b),(2.33c)}{\lesssim} \varsigma\sup_{\substack{\varphi\in H_*^1(\omega_\ell^{\boldsymbol{a}})\\ \|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}}=1}}\Big[(f,\psi_\ell^{\boldsymbol{a}}\varphi)_{\omega_\ell^{\boldsymbol{a}}}-(\boldsymbol{\xi}_\ell^k,\boldsymbol{\nabla}(\psi_\ell^{\boldsymbol{a}}\varphi))_{\omega_\ell^{\boldsymbol{a}}}\Big]+\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}+\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k}\\
&\overset{(2.33d)}{\lesssim} \varsigma\sup_{\substack{\varphi\in H_*^1(\omega_\ell^{\boldsymbol{a}})\\ \|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}}=1}}\Big[(f,\psi_\ell^{\boldsymbol{a}}\varphi)_{\omega_\ell^{\boldsymbol{a}}}-(a_\ell^k\boldsymbol{\nabla}u_\ell^k,\boldsymbol{\nabla}(\psi_\ell^{\boldsymbol{a}}\varphi))_{\omega_\ell^{\boldsymbol{a}}}\Big]+\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}+\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k}\\
&\overset{(2.5a)}{=} \varsigma\sup_{\substack{\varphi\in H_*^1(\omega_\ell^{\boldsymbol{a}})\\ \|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}}=1}}(a_u\boldsymbol{\nabla}u-a_\ell^k\boldsymbol{\nabla}u_\ell^k,\boldsymbol{\nabla}(\psi_\ell^{\boldsymbol{a}}\varphi))_{\omega_\ell^{\boldsymbol{a}}}+\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}+\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k}\\
&\lesssim \varsigma\|a_\ell^k\boldsymbol{\nabla}u_\ell^k-a_u\boldsymbol{\nabla}u\|_{\omega_\ell^{\boldsymbol{a}}}+\eta_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}+\eta_{\mathrm{iter},\ell}^{\boldsymbol{a},k},
\end{aligned}
\tag{2.72}
$$

where we have used $\|\boldsymbol{\nabla}(\psi_\ell^{\boldsymbol{a}}\varphi)\|_{\omega_\ell^{\boldsymbol{a}}}\lesssim\|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}}$ as in [21, 52]. In conclusion, we obtain

$$
\begin{aligned}
\eta_{\mathrm{N},\ell}^k &\overset{(2.32b),(2.69)}{\lesssim} \left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}\frac{1}{\operatorname{ess\,inf}_{\omega_\ell^{\boldsymbol{a}}}a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}\|\psi_\ell^{\boldsymbol{a}}a_\ell^k\boldsymbol{\nabla}u_\ell^k+\boldsymbol{\sigma}_\ell^{\boldsymbol{a},k}\|_{\omega_\ell^{\boldsymbol{a}}}^2\right)^{\frac12}\\
&\overset{(2.72),(2.33)}{\lesssim} \left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}\frac{1}{\operatorname{ess\,inf}_{\omega_\ell^{\boldsymbol{a}}}a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k}}\|a_\ell^k\boldsymbol{\nabla}u_\ell^k-a_u\boldsymbol{\nabla}u\|_{\omega_\ell^{\boldsymbol{a}}}^2\right)^{\frac12}+\eta_{\mathrm{osc,q},\ell}^k+\eta_{\mathrm{iter},\ell}^k\\
&\overset{(2.36)}{\leq} C_\ell^k\left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}\frac{1}{\operatorname{ess\,sup}_{\omega_\ell^{\boldsymbol{a}}}a_{\mathrm{c},\ell}^{\boldsymbol{\nabla}u,k}}\|a_\ell^k\boldsymbol{\nabla}u_\ell^k-a_u\boldsymbol{\nabla}u\|_{\omega_\ell^{\boldsymbol{a}}}^2\right)^{\frac12}+\eta_{\mathrm{osc,q},\ell}^k+\eta_{\mathrm{iter},\ell}^k\\
&\overset{(2.68),(2.32c)}{\lesssim} C_\ell^k\mathcal{E}_{\mathrm{N},\ell}^k+\eta_{\mathrm{osc,q},\ell}^k+\eta_{\mathrm{iter},\ell}^k,
\end{aligned}
\tag{2.73}
$$

where we have used the triangle inequality on $\ell_2(\mathbb{R}^{|\mathcal{V}_\ell|})$. □

2.9 Proof of Lemmas 2.5.1 and 2.5.3

Proof of Lemma 2.5.1. We write with the triangle inequality

$$
\begin{aligned}
\eta_{\mathrm{N},\ell}^k &\overset{(2.59)}{\leq} \|(a_{\mathrm{m},\ell}^{\boldsymbol{\sigma},k})^{-\frac12}(a_\ell^k\boldsymbol{\nabla}u_\ell^k+\boldsymbol{\sigma}_\ell^k)\| \overset{(2.31),(2.11)}{\leq} a_{\mathrm{m}}^{-\frac12}A_{\mathrm{c}}^{\frac12}\|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(a_\ell^k\boldsymbol{\nabla}u_\ell^k+\boldsymbol{\sigma}_\ell^k)\|\\
&\leq a_{\mathrm{m}}^{-\frac12}A_{\mathrm{c}}^{\frac12}\Big(\|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_\ell^k-u_\ell^{k-1})+a_\ell^{k-1}\boldsymbol{\nabla}u_\ell^{k-1}+\boldsymbol{\sigma}_\ell^k)\|\\
&\qquad +\|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(a_\ell^k\boldsymbol{\nabla}u_\ell^k-a_\ell^{k-1}\boldsymbol{\nabla}u_\ell^{k-1}-\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_\ell^k-u_\ell^{k-1}))\|\Big)\\
&\overset{(2.39b),(2.40)}{=} a_{\mathrm{m}}^{-\frac12}A_{\mathrm{c}}^{\frac12}(\eta_{\mathrm{L},\ell}^k+\widehat{\eta}_{\mathrm{iter},\ell}^k),
\end{aligned}
\tag{2.74}
$$

where we have also employed the definitions (2.12) and (2.25b). Thus, by virtue of the definition (2.41), λ_ℓ^k is uniformly bounded as per (2.43). □

Proof of Lemma 2.5.3. The first two inequalities in (2.45) are immediate from (2.42) and (2.43). As for the last one, for all $k \geq 1$,

$$\begin{aligned}
A_{\mathrm{m}}\|\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})\|^2 &\overset{(2.11)}{\leq} (\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1}), \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})) \\
&\overset{(2.12)}{=} (\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}u_{\langle\ell\rangle}^k - \boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})) - (a_\ell^{k-1}\boldsymbol{\nabla}u_\ell^{k-1}, \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})) \\
&\overset{(2.38a)}{=} (f, u_{\langle\ell\rangle}^k - u_\ell^{k-1}) - (a_\ell^{k-1}\boldsymbol{\nabla}u_\ell^{k-1}, \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})) \\
&\overset{(2.5a)}{=} (a_u\boldsymbol{\nabla}u - a_\ell^{k-1}\boldsymbol{\nabla}u_\ell^{k-1}, \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})) \\
&\overset{(2.4a)}{\leq} a_{\mathrm{c}}\|\boldsymbol{\nabla}(u - u_\ell^{k-1})\|\|\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})\|.
\end{aligned} \tag{2.75}$$

Hence, dividing (2.75) by $\|\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})\|$, we infer

$$A_{\mathrm{m}}a_{\mathrm{c}}^{-1}\|\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^{k-1})\| \leq \|\boldsymbol{\nabla}(u - u_\ell^{k-1})\|,$$

and consequently, by the triangle inequality,

$$A_{\mathrm{m}}a_{\mathrm{c}}^{-1}\|\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^k)\| \leq \|\boldsymbol{\nabla}(u - u_\ell^k)\| + \|\boldsymbol{\nabla}(u_\ell^k - u_\ell^{k-1})\| + A_{\mathrm{m}}a_{\mathrm{c}}^{-1}\|\boldsymbol{\nabla}(u_\ell^k - u_\ell^{k-1})\|.$$

Thus, $\mathcal{E}_{\mathrm{L},\ell}^k \lesssim \mathcal{E}_{\mathrm{N},\ell}^k + \|\boldsymbol{\nabla}(u_\ell^k - u_\ell^{k-1})\|$ thanks to definition (2.39a) (employing the relation (2.77a) below) and

$$a_{\mathrm{m}}\|\boldsymbol{\nabla}(u_\ell^k - u)\|^2 \leq (\mathcal{E}_{\mathrm{N},\ell}^k)^2 \leq a_{\mathrm{c}}\|\boldsymbol{\nabla}(u_\ell^k - u)\|^2, \tag{2.76}$$

which is a consequence of inequality (2.32a) together with the bounds in (2.31).

The convergence statement is then easy. Assuming $\mathcal{E}_\ell^k \to 0$ implies $\mathcal{E}_{\mathrm{N},\ell}^k \to 0$ and then the $H_0^1(\Omega)$ convergence follows from (2.76). Reciprocally, when $u_\ell^k \to u$ in $H_0^1(\Omega)$, the last term in (2.45) goes to 0, as well as $\mathcal{E}_{\mathrm{N},\ell}^k$ thanks to (2.76), so that $\mathcal{E}_\ell^k \to 0$. □

2.10 Proof of Theorem 2.5.5

Recalling the definition (2.10) of $\mathcal{J}_\ell^{k-1}$, we can rewrite $\mathcal{E}_{\mathrm{L},\ell}^k$ given by (2.39a) as

$$\begin{aligned}
(\mathcal{E}_{\mathrm{L},\ell}^k)^2 &\overset{(2.10)}{=} \left\|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}u_\ell^k\right\|^2 - \left\|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}u_{\langle\ell\rangle}^k\right\|^2 \\
&\qquad - 2(\boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla}(u_\ell^k - u_{\langle\ell\rangle}^k)) - 2(f, u_\ell^k - u_{\langle\ell\rangle}^k) \\
&\overset{(2.38a)}{=} \left\|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}u_\ell^k\right\|^2 - \left\|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}u_{\langle\ell\rangle}^k\right\|^2 - 2(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}u_{\langle\ell\rangle}^k, \boldsymbol{\nabla}(u_\ell^k - u_{\langle\ell\rangle}^k)) \\
&= \|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}(u_\ell^k - u_{\langle\ell\rangle}^k)\|^2 \overset{(2.68)}{=} \frac{1}{d+1}\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\mathcal{E}_{\mathrm{L},\ell}^{\boldsymbol{a},k})^2,
\end{aligned} \tag{2.77a}$$

where, for all $\boldsymbol{a} \in \mathcal{V}_\ell$,

$$\mathcal{E}_{\mathrm{L},\ell}^{\boldsymbol{a},k} \coloneqq \|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}(u_\ell^k - u_{\langle\ell\rangle}^k)\|_{\omega_\ell^{\boldsymbol{a}}}. \tag{2.77b}$$

Similarly, using the definition (2.23) of $\mathcal{J}_\ell^{*,k-1}$, we can rewrite and upper-bound $\eta_{\mathrm{L},\ell}^k$ of (2.39b),

$$\begin{aligned}(\eta_{\mathrm{L},\ell}^k)^2 &\overset{(2.23)}{=} \left\|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla} u_\ell^k\right\|^2 + \left\|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\boldsymbol{\sigma}_\ell^k - \boldsymbol{b}_\ell^{k-1})\right\|^2 - 2(\boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla} u_\ell^k) - 2(f, u_\ell^k)\\ &= \|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1} + \boldsymbol{\sigma}_\ell^k)\|^2 - 2(\boldsymbol{\sigma}_\ell^k, \boldsymbol{\nabla} u_\ell^k) - 2(f, u_\ell^k)\\ &\overset{(2.28)}{=} \|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1} + \boldsymbol{\sigma}_\ell^k)\|^2 \overset{(2.69)}{\le} (d+1)\sum_{\boldsymbol{a}\in\mathcal{V}_\ell}(\eta_{\mathrm{L},\ell}^{\boldsymbol{a},k})^2,\end{aligned} \tag{2.78a}$$

where, for all $\boldsymbol{a}\in\mathcal{V}_\ell$,

$$\eta_{\mathrm{L},\ell}^{\boldsymbol{a},k} \coloneqq \|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\psi_\ell^{\boldsymbol{a}}(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1}) + \boldsymbol{\sigma}_\ell^{\boldsymbol{a},k})\|_{\omega_\ell^{\boldsymbol{a}}}. \tag{2.78b}$$

Proof of Theorem 2.5.5. Let $k\ge 1$. Since (2.38) is a linear problem, proceeding as in [52, 53], we obtain the a posteriori error estimate

$$\begin{aligned}(\mathcal{E}_{\mathrm{L},\ell}^k)^2 &\overset{(2.77a)}{=} (\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^k), \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^k))\\ &\overset{(2.38a)}{=} (f, u_{\langle\ell\rangle}^k - u_\ell^k) - (\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1}, \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^k))\\ &\overset{(2.27)}{=} ((I-\Pi_{\ell,p})f, u_{\langle\ell\rangle}^k - u_\ell^k) - (\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1} + \boldsymbol{\sigma}_\ell^k, \boldsymbol{\nabla}(u_{\langle\ell\rangle}^k - u_\ell^k))\\ &\overset{(2.46a)}{\le} (\widehat{\eta}_{\mathrm{osc},\ell}^k + \|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\boldsymbol{A}_\ell^{k-1}\boldsymbol{\nabla} u_\ell^k - \boldsymbol{b}_\ell^{k-1} + \boldsymbol{\sigma}_\ell^k)\|)\|(\boldsymbol{A}_\ell^{k-1})^{\frac12}\boldsymbol{\nabla}(u_\ell^k - u_{\langle\ell\rangle}^k)\|\\ &\overset{(2.77a),(2.78a)}{=} (\widehat{\eta}_{\mathrm{osc},\ell}^k + \eta_{\mathrm{L},\ell}^k)\mathcal{E}_{\mathrm{L},\ell}^k.\end{aligned} \tag{2.79}$$

Now, observing that (2.71) still holds with the version (2.26b) of the equilibrated flux, we obtain (2.47a) by writing

$$\begin{aligned}\mathcal{E}_\ell^k &\overset{(2.42)}{=} \frac12(\mathcal{E}_{\mathrm{N},\ell}^k + \lambda_\ell^k\mathcal{E}_{\mathrm{L},\ell}^k)\\ &\overset{(2.71),(2.79)}{\le} \frac12(\eta_{\mathrm{N},\ell}^k + 2\eta_{\mathrm{osc},\ell}^k + \lambda_\ell^k(\eta_{\mathrm{L},\ell}^k + \widehat{\eta}_{\mathrm{osc},\ell}^k))\\ &\overset{(2.42)}{=} \eta_\ell^k + \eta_{\mathrm{osc},\ell}^k + \frac{\lambda_\ell^k}{2}\widehat{\eta}_{\mathrm{osc},\ell}^k.\end{aligned} \tag{2.80}$$

It remains to prove (2.47b). For all vertices $\boldsymbol{a}\in\mathcal{V}_\ell$, denote by $\boldsymbol{\sigma}_{\mathrm{I},\ell}^{\boldsymbol{a},k}$ the minimizer of (2.25b) with the choice (2.26a). Recall also the notations (2.78b) and (2.25b) together with (2.46c) and (2.46d). Then, using the same reasoning as in (2.72), we infer

$$\begin{aligned}\eta_{\mathrm{L},\ell}^{\boldsymbol{a},k} &\overset{(2.46d)}{\le} \|(\boldsymbol{A}_\ell^{k-1})^{-\frac12}(\psi_\ell^{\boldsymbol{a}}\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}}\boldsymbol{\xi}_\ell^k + \boldsymbol{\sigma}_\ell^{\boldsymbol{a},k})\|_{\omega_\ell^{\boldsymbol{a}}} + \widehat{\eta}_{\mathrm{q},\ell}^{\boldsymbol{a},k}\\ &\overset{(2.11)}{\le} \frac{1}{(\inf_{\omega_\ell^{\boldsymbol{a}}} A_{\mathrm{m},\ell}^{k-1})^{\frac12}}\|\psi_\ell^{\boldsymbol{a}}\boldsymbol{\Pi}_{\ell,p-1}^{\boldsymbol{RTN}}\boldsymbol{\xi}_\ell^k + \boldsymbol{\sigma}_{\mathrm{I},\ell}^{\boldsymbol{a},k}\|_{\omega_\ell^{\boldsymbol{a}}} + \widehat{\eta}_{\mathrm{q},\ell}^{\boldsymbol{a},k}\\ &\overset{(2.72),(2.48)}{\lesssim} \frac{1}{(\inf_{\omega_\ell^{\boldsymbol{a}}} A_{\mathrm{m},\ell}^{k-1})^{\frac12}}\sup_{\substack{\varphi\in H_*^1(\omega_\ell^{\boldsymbol{a}})\\ \|\boldsymbol{\nabla}\varphi\|_{\omega_\ell^{\boldsymbol{a}}}=1}}\left[(f, \psi_\ell^{\boldsymbol{a}}\varphi)_{\omega_\ell^{\boldsymbol{a}}} - (\boldsymbol{\xi}_\ell^k, \boldsymbol{\nabla}(\psi_\ell^{\boldsymbol{a}}\varphi))_{\omega_\ell^{\boldsymbol{a}}}\right]\\ &\qquad + (1+\widehat{C}_\ell^k)\widehat{\eta}_{\mathrm{q},\ell}^{\boldsymbol{a},k} + \widehat{\eta}_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}\\ &\overset{(2.38a),(2.77b)}{\lesssim} \widehat{C}_\ell^k\mathcal{E}_{\mathrm{L},\ell}^{\boldsymbol{a},k} + \widehat{C}_\ell^k\widehat{\eta}_{\mathrm{q},\ell}^{\boldsymbol{a},k} + \widehat{\eta}_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k}.\end{aligned} \tag{2.81}$$

Therefore, we obtain (2.47b) by writing

$$\begin{aligned}
2\eta_\ell^k &\overset{(2.42)}{=} \eta_{\mathrm{N},\ell}^k + \lambda_\ell^k \eta_{\mathrm{L},\ell}^k \overset{(2.41)}{=} \lambda_\ell^k (2\eta_{\mathrm{L},\ell}^k + \widehat{\eta}_{\mathrm{iter},\ell}^k) \\
&\overset{(2.78)}{\lesssim} \lambda_\ell^k \left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell} (\eta_{\mathrm{L},\ell}^{\boldsymbol{a},k})^2 \right)^{\frac{1}{2}} + \lambda_\ell^k \widehat{\eta}_{\mathrm{iter},\ell}^k \\
&\overset{(2.81)}{\lesssim} \lambda_\ell^k \left[\widehat{C}_\ell^k \left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell} (\mathcal{E}_{\mathrm{L},\ell}^{\boldsymbol{a},k})^2 \right)^{\frac{1}{2}} + \widehat{C}_\ell^k \left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell} (\widehat{\eta}_{\mathrm{q},\ell}^{\boldsymbol{a},k})^2 \right)^{\frac{1}{2}} + \left(\sum_{\boldsymbol{a}\in\mathcal{V}_\ell} (\widehat{\eta}_{\mathrm{osc,q},\ell}^{\boldsymbol{a},k})^2 \right)^{\frac{1}{2}} \right] \\
&\qquad + \lambda_\ell^k \widehat{\eta}_{\mathrm{iter},\ell}^k \\
&\overset{(2.77),(2.46)}{\lesssim} \lambda_\ell^k \left[\frac{1}{2} \widehat{C}_\ell^k \mathcal{E}_{\mathrm{L},\ell}^k + \widehat{C}_\ell^k \widehat{\eta}_{\mathrm{q},\ell}^k + \widehat{\eta}_{\mathrm{osc,q},\ell}^k + \widehat{\eta}_{\mathrm{iter},\ell}^k \right] \\
&\overset{(2.42)}{\leq} \widehat{C}_\ell^k \mathcal{E}_\ell^k + \lambda_\ell^k (\widehat{C}_\ell^k \widehat{\eta}_{\mathrm{q},\ell}^k + \widehat{\eta}_{\mathrm{osc,q},\ell}^k + \widehat{\eta}_{\mathrm{iter},\ell}^k),
\end{aligned} \tag{2.82}$$

where we used the triangle inequality on $\ell_2(\mathbb{R}^{|\mathcal{V}_\ell|})$. Finally, we get $\widehat{C}_\ell^k = 1$ for the Zarantonello iteration since $A_{\mathrm{m},\ell}^{k-1} = A_{\mathrm{c},\ell}^{k-1} = \gamma$ in Ω in this case. □

2.A Equivalent assumptions on the nonlinear functions

In this section, we show some useful properties of the nonlinear functions a and ϕ. In particular, we show that inequalities (2.4) admit equivalent versions with the function ϕ defined in (2.7), preserving the same constants a_{c} and a_{m}.

Proposition 2.A.1 (Equivalent assumption on ϕ')**.** *Inequalities* (2.4) *are equivalent to the following ones: a.e. in* Ω*, for all* $r, s \in [0,\infty)$*,*

$$|\phi'(\cdot,r) - \phi'(\cdot,s)| \leq a_{\mathrm{c}}|r-s|, \tag{2.83a}$$

$$(\phi'(\cdot,r) - \phi'(\cdot,s))(r-s) \geq a_{\mathrm{m}}(r-s)^2. \tag{2.83b}$$

Proof. Differentiating (2.7) gives, a.e. in Ω and for all $r \in [0,\infty)$,

$$\phi'(\cdot,r) = a(\cdot,r)r. \tag{2.84}$$

Thus, assuming and using (2.4) (with $\boldsymbol{x} = (r,0,\dots,0)^{\mathrm{t}}$ and $\boldsymbol{y} = (s,0,\dots,0)^{\mathrm{t}}$) together with (2.84) yields (2.83).

Reciprocally, assuming (2.83), we have in particular (with $s=0$), using (2.84), a.e. in Ω and for all $r \in [0,\infty)$,

$$a_{\mathrm{m}} \leq a(\cdot,r) \leq a_{\mathrm{c}}. \tag{2.85}$$

We conclude by using the fact that for all $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d$, and all $\alpha,\beta \in [0,\infty)$, we have

$$|\boldsymbol{x}-\boldsymbol{y}|^2 = (|\boldsymbol{x}| - |\boldsymbol{y}|)^2 + 2(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y}), \tag{2.86a}$$

$$(\alpha\boldsymbol{x} - \beta\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y}) = (\alpha|\boldsymbol{x}| - \beta|\boldsymbol{y}|)(|\boldsymbol{x}| - |\boldsymbol{y}|) + (\alpha+\beta)(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y}), \tag{2.86b}$$

to obtain, a.e. in Ω and for all $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d$,

$$\begin{aligned}
&|a(\cdot,|\boldsymbol{x}|)\boldsymbol{x} - a(\cdot,|\boldsymbol{y}|)\boldsymbol{y}|^2 \\
&\quad \overset{(2.86a)}{=} (a(\cdot,|\boldsymbol{x}|)|\boldsymbol{x}| - a(\cdot,|\boldsymbol{y}|)|\boldsymbol{y}|)^2 + 2a(\cdot,|\boldsymbol{x}|)a(\cdot,|\boldsymbol{y}|)(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y}) \\
&\quad \overset{(2.84)}{=} (\phi'(\cdot,|\boldsymbol{x}|) - \phi'(\cdot,|\boldsymbol{y}|))^2 + 2a(\cdot,|\boldsymbol{x}|)a(\cdot,|\boldsymbol{y}|)(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y}) \\
&\quad \overset{(2.83),(2.85)}{\leq} a_{\mathrm{c}}^2[(|\boldsymbol{x}| - |\boldsymbol{y}|)^2 + 2(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y})] = a_{\mathrm{c}}^2|\boldsymbol{x}-\boldsymbol{y}|^2
\end{aligned}$$

and

$$\begin{aligned}
&(a(\cdot,|\boldsymbol{x}|)\boldsymbol{x} - a(\cdot,|\boldsymbol{y}|)\boldsymbol{y}) \cdot (\boldsymbol{x}-\boldsymbol{y}) \\
&\overset{(2.86b)}{=} (a(\cdot,|\boldsymbol{x}|)|\boldsymbol{x}| - a(\cdot,|\boldsymbol{y}|)|\boldsymbol{y}|)(|\boldsymbol{x}|-|\boldsymbol{y}|) + (a(\cdot,|\boldsymbol{x}|)+a(\cdot,|\boldsymbol{y}|))(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y}) \\
&\overset{(2.84)}{=} (\phi'(\cdot,|\boldsymbol{x}|) - \phi'(\cdot,|\boldsymbol{y}|))(|\boldsymbol{x}|-|\boldsymbol{y}|) + (a(\cdot,|\boldsymbol{x}|)+a(\cdot,|\boldsymbol{y}|))(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y}) \\
&\overset{(2.83),(2.85)}{\geq} a_{\mathrm{m}}[(|\boldsymbol{x}|-|\boldsymbol{y}|)^2 + 2(|\boldsymbol{x}||\boldsymbol{y}| - \boldsymbol{x}\cdot\boldsymbol{y})] = a_{\mathrm{m}}|\boldsymbol{x}-\boldsymbol{y}|^2,
\end{aligned}$$

hence (2.4). □

Proposition 2.A.2 (Equivalent assumption on ϕ'')**.** *Inequalities* (2.83) *are equivalent to the facts that* ϕ' *is weakly differentiable and, a.e. in* Ω *and for a.e.* $r \in [0,\infty)$*,*

$$a_{\mathrm{m}} \leq \phi''(\cdot,r) \leq a_{\mathrm{c}}. \tag{2.87}$$

Proof. Assuming (2.83), since ϕ' is Lipschitz continuous thanks to (2.83a), ϕ' is weakly differentiable. Furthermore, defining the difference quotient, a.e. in Ω and for all $r,s \in [0,\infty)$, $r \neq s$, by

$$\tau(\cdot,r,s) \coloneqq \frac{\phi'(\cdot,r) - \phi'(\cdot,s)}{r-s},$$

inequalities (2.83) are equivalent to the fact that, a.e. in Ω and for all $r,s \in [0,\infty)$, $r \neq s$,

$$a_{\mathrm{m}} \leq \tau(\cdot,r,s) \leq a_{\mathrm{c}}. \tag{2.88}$$

Hence, letting s tend to r in (2.88) gives (2.87).

On the other hand, assuming and integrating (2.87) gives, a.e. in Ω and for a.e. $r,s \in [0,\infty)$ with $r > s$,

$$a_{\mathrm{m}}(r-s) \leq \phi'(\cdot,r) - \phi'(\cdot,s) \leq a_{\mathrm{c}}(r-s), \tag{2.89}$$

and dividing (2.89) by $r-s$ gives (2.88), which becomes also true for $r < s$ by symmetry. □

Remark 2.A.3 (Convexity)**.** *Under Assumption 2.2.1, inequality* (2.83b) *implies that* ϕ' *is nondecreasing, i.e.,* ϕ *is convex. Moreover, from* (2.84)*, we have, a.e. in* Ω *and for a.e.* $r \in [0,\infty)$*,*

$$\phi''(\cdot,r) = a(\cdot,r) + a'(\cdot,r)r. \tag{2.90}$$

2.B Spectral properties of the tensor product

Lemma 2.B.1 (Spectral properties of the tensor product)**.** *The following holds:*

$$\mathrm{Spec}(\alpha \boldsymbol{I}_d + \beta \boldsymbol{A}) = \{\alpha\} + \beta\,\mathrm{Spec}(\boldsymbol{A}) \qquad \forall \alpha,\beta \in \mathbb{R},\ \forall \boldsymbol{A} \in \mathbb{R}^{d\times d}, \tag{2.91a}$$

$$\mathrm{Spec}(\boldsymbol{\xi}\otimes\boldsymbol{\xi}) = \{0, |\boldsymbol{\xi}|^2\} \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^d,\ d > 1. \tag{2.91b}$$

Proof. We refer to [37] for details about the tools used in the following. Denoting $P_{\boldsymbol{A}}$ the characteristic polynomial of $\boldsymbol{A}$, we obtain (2.91a) by writing, for all $\lambda \in \mathbb{R}$,

$$P_{\alpha\boldsymbol{I}_d+\beta\boldsymbol{A}}(\alpha+\beta\lambda) = \det((\alpha+\beta\lambda)\boldsymbol{I}_d - (\alpha\boldsymbol{I}_d+\beta\boldsymbol{A})) = \beta^d \det(\lambda\boldsymbol{I}_d - \boldsymbol{A}) = \beta^d P_{\boldsymbol{A}}(\lambda).$$

Moving to (2.91b), since $(\boldsymbol{\xi}\otimes\boldsymbol{\xi})\boldsymbol{\tau} = (\boldsymbol{\xi}\cdot\boldsymbol{\tau})\boldsymbol{\xi}$ for all $\boldsymbol{\tau} \in \mathbb{R}^d$, $\dim(\mathrm{Ker}(\boldsymbol{\xi}\otimes\boldsymbol{\xi})) = \dim(\mathrm{Ker}(\langle\boldsymbol{\xi},\cdot\rangle)) \geq d-1$, i.e. $0 \in \mathrm{Spec}(\boldsymbol{\xi}\otimes\boldsymbol{\xi})$ with a multiplicity of at least $d-1$. Thus, the sum of the eigenvalues of $\boldsymbol{\xi}\otimes\boldsymbol{\xi}$, being $\mathrm{tr}(\boldsymbol{\xi}\otimes\boldsymbol{\xi}) = |\boldsymbol{\xi}|^2$, is in $\mathrm{Spec}(\boldsymbol{\xi}\otimes\boldsymbol{\xi})$. Hence, 0 and $|\boldsymbol{\xi}|^2$ are the only elements of $\mathrm{Spec}(\boldsymbol{\xi}\otimes\boldsymbol{\xi})$, and we infer (2.91b). □

Chapter 3

Adaptive regularization for the Richards equation

This chapter corresponds to the paper https://hal.science/hal-04266827v1 currently submitted for publication.

3.1 Introduction

One of the most fundamental equations for modeling flows in porous media is the Richards equation. It can be viewed as a simplified two-phase model, e.g., for water and air, where one of the phases is assumed to be of constant pressure. For a detailed review of the role of the Richards equation in porous media modeling see, e.g., [34, 39]. The equation describes the evolution in space and in time of the pressure p and saturation s for a fluid in a porous medium. Given a domain $\Omega \subset \mathbb{R}^d$, for $d = 1, 2, 3$, and a final time $T > 0$, the Richards equation is given by

$$\partial_t s - \nabla \cdot [\bar{\mathbf{K}} \kappa(s)(\nabla p + \boldsymbol{g})] = f, \quad \text{in } \Omega \times [0, T], \tag{3.1}$$

where $-\boldsymbol{g}$ represents the constant force of gravity, $\bar{\mathbf{K}} : \Omega \to \mathbb{R}^{d \times d}$ is the absolute permeability tensor, $\kappa : [0, 1] \to [0, 1]$ is the relative permeability function, and $f : \Omega \times [0, T] \to \mathbb{R}$ a forcing term, all considered as data along with suitable initial and boundary conditions. The system is closed by an algebraic relationship expressing the saturation as a function of the pressure, i.e., there exists a function $S : \mathbb{R} \to [0, 1]$ such that

$$s = S(p). \tag{3.2}$$

For the well-posedness of this initial boundary value problem see, e.g., [2].

Realistic choices for the saturation function S of (3.2) and relative permeability κ are nonlinear cf. Figures 3.1 and 3.2. This means that once the equation (3.1) is discretized with an implicit time-stepping scheme, a linearization procedure must be applied at each timestep.

The design of robust and efficient linearization schemes for Richards equation is an active area of research. Newton's method [86, 87, 42] is an attractive choice due to its potentially quadratic convergence. A sufficient condition for the convergence of Newton's method in the context of the Richards equation was derived in [75]. In particular, the authors of [75]

considered the lowest order continuous Galerkin finite element method (FEM) as a spatial discretization and an implicit Euler time discretization and derived a condition of the form

$$\tau < C S_{\mathrm{m}}^{\frac{2+r}{r}} h^d, \tag{3.3}$$

where τ is the time step size, h is the mesh size, $S_{\mathrm{m}} := \inf S' \geq 0$, and $C, r > 0$ depends on the S and κ. In practice, satisfying this condition may render the timestep τ prohibitively small, or the condition may even be impossible to satisfy if the derivative $S' = 0$. Other linearization schemes include the modified Picard method [31], L-schemes [111, 87, 99, 92], and the Jäger–Kăcur method [77, 78]. These methods are generally more robust than Newton's method at the cost of slower convergence. In particular, the L-scheme was shown to be unconditionally convergent in [105], though it only converges linearly.

The tradeoff between the robustness of the linearly-converging methods and the speed of Newton's method have motivated hybrid methods such as those studied [87] where the authors apply several iterations of the slower scheme to provide an initial guess to Newton. More recently, this strategy was taken further in [114], where an a posteriori error estimator was designed to provide a criteria to switch between the L-scheme and Newton's method.

The degenerate nature of the Richards equation (see §3.3) partially explains the difficulty encountered by Newton's method. One way to address the degeneracy involves the choice of the unknown in (3.1). In particular, whenever the function S in (3.2) is invertible one has the choice of whether to solve for the pressure p or the saturation s in (3.1). This idea led to the so-called primary variable switching methods [56, 44]. Initially, these methods required a local choice (by looping over degrees of freedom in the context of a Galerkin method) for which variable to solve for. This idea was elegantly adapted in [22], where the authors achieved a continuous variable switch by introducing a global C^1 parameterization of the saturation curve (3.2). The parameter is chosen so that it is proportional to the saturation in the dry regions (where $s \ll 1$) and otherwise it is proportional to the pressure. The continuous variable switching was recently generalized in [16] as well as the PhD thesis [15] where the authors consider an additional switch that aids in the case of heterogeneous media.

On top of difficulties related to degeneracy, the specific forms of the nonlinearities for the most common models (Brooks–Corey and van Genuchten–Mualem defined in §3.2) also suffer from low regularity. Approaches to address this while keeping good convergence of Newton's method include the line search or trust region methods, where the step size of Newton is limited in certain critical zones of the nonlinearity, see, e.g., [79, 123, 14] and references therein. Another alternative to handle low regularity is the so-called semismooth Newton method [82, 104], where the main tenant is to work with elements of the subdifferential to a nonsmooth function.

A useful tool in the context of Richards equation, both theoretically and practically, is regularization, i.e, considering an auxiliary perturbed problem to obtain some desired properties. The authors of [100] rely on regularization to prove well-posedness of a certain case of the Richards equation. Regularization has already been explored for improving the performance of iterative schemes in, for example, [78, 14]. In [78], regularized versions of the nonlinear functions are introduced to control the degeneracy whereas in [14] a kind of slope limiting method was proposed to handle the case where the derivative of the relative permeability κ tends to infinity. However, in practice, a natural question to ask is how much regularization should be added to obtain a tradeoff between model error and performance.

In this work, we seek to provide a possible answer to the question by adaptively updating a regularization parameter. In particular, the adaptive choice is informed by a posteriori error

estimators. Our a posteriori estimators follow the spirit of those derived in [94] in the context of the fully degenerate Richards equation, where a rigorous a posteriori analysis leads to a reliable and efficient estimator.

We also take inspiration from our recent work [58], where we introduced an adaptive algorithm for regularizing a nonsmooth nonlinearity based on an additive decomposition of an estimator. The central observation is that *regularization (model) error is often dominated by the discretization error and hence does not impact the overall accuracy of the scheme.* In the setting of [58], we were able to recover the optimal rate of convergence with respect to total degrees of freedom by solving a sequence of regularized problems without ever sending the regularization parameter to 0. We seek to apply this same strategy to this setting, where component estimators will guide an adaptive algorithm and in particular ensure that the regularization component estimator remains sufficiently below that of discretization.

Our scheme has several advantages to those already mentioned. Firstly, it does not require the modification of the underlying Newton solver. Secondly, for the test cases we consider, the regularization allows Newton to converge where it takes hundreds of iterations without. Finally, by adaptively lowering the level of regularization, we are able to produce a solution that matches well visually with a solution obtained without regularization.

The rest of the paper is organized as follows. In §3.2, we introduce the necessary notation as well as the assumptions on the data. In §3.3, we discuss the various difficulties for a nonlinear solver and introduce our proposed regularization. In §3.4, we present the backward Euler–finite element discretization of the Richards equation with its corresponding regularized and linearized problems. In §3.5 we detail our adaptive algorithm. In §3.6, we present numerical experiments and we conclude in §3.7 where we also give an outlook on future research.

3.2 Setting and specification of the data

In this section we detail the necessary information to describe precisely the problem under consideration. We use the standard notation from functional analysis. Let $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$ be a domain with polygonal boundary. For $\omega \subseteq \Omega$, let $(\cdot,\cdot)_\omega$ and $\|\cdot\|_\omega$ correspond to the $L^2(\omega)$ inner product and norm respectively. We drop the subscripts when $\omega = \Omega$. Let $H^1(\Omega)$ be the Sobolev space of functions defined on Ω with first-order weak derivatives in $L^2(\Omega)$. We also introduce the space $\boldsymbol{H}(\mathrm{div}, \Omega) := \{\boldsymbol{v} \in [L^2(\Omega)]^d : \nabla\cdot\boldsymbol{v} \in L^2(\Omega)\}$.

We now specify the initial and boundary conditions for the problem (3.1). We consider a partition of the boundary $\partial\Omega = \Gamma_\mathrm{D} \cup \Gamma_\mathrm{N}$ into Dirichlet and Neumann boundaries, where Γ_D has strictly positive measure. The boundary conditions are specified as

$$p = p_\mathrm{D} \ \text{ on } \Gamma_\mathrm{D} \times (0,T], \quad \bar{\mathbf{K}}\kappa(s)(\nabla p + \boldsymbol{g})\cdot\boldsymbol{n} = 0 \ \text{ on } \Gamma_\mathrm{N} \times (0,T], \tag{3.4}$$

and the corresponding space incorporating the Dirichlet boundary condition is given by

$$H^1_\mathrm{D}(\Omega) := \{v \in H^1(\Omega) : v|_{\Gamma_\mathrm{D}} = p_\mathrm{D}\}. \tag{3.5}$$

The initial condition is imposed on the saturation,

$$s(\boldsymbol{x}, 0) = s_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{3.6}$$

We introduce a conforming triangulation $\mathcal{T}_h$ of Ω, i.e., $\mathcal{T}_h = \cup_K K$ where the intersection of (the closure of) two simplices $K, K' \in \mathcal{T}_h$ are either disjoint or an l-dimensional simplex for $0 \le l \le d-1$.

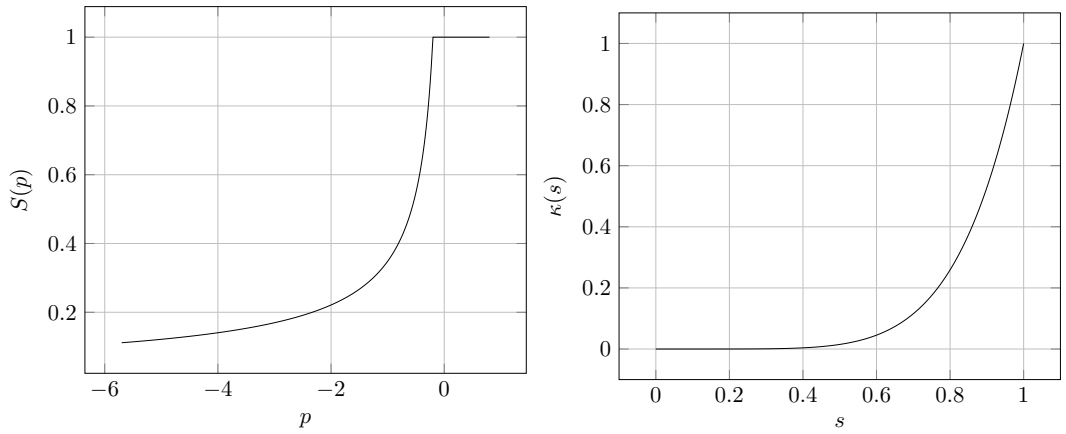


Figure 3.1: $[p_{\mathrm{M}} = -0.2, \lambda_1 = 0.66]$ Saturation and relative permeability functions for the Brooks–Corey equation of state (3.7).

We consider the following assumptions on the data of problem (3.1)-(3.2).

Assumption 3.2.1. *The absolute permeability tensor $\bar{\mathbf{K}} : \Omega \to \mathbb{R}^{d\times d}$ is piecewise constant with respect to $\mathcal{T}_h$ and satisfies the ellipticity and boundedness conditions, i.e., there exists constants $K_{\mathrm{m}}, K_{\mathrm{M}} > 0$ such that, for almost every $\boldsymbol{x} \in \Omega$, there holds*

$$K_{\mathrm{m}}|\boldsymbol{x}|^2 \leq \boldsymbol{x}^T \bar{\mathbf{K}}(\boldsymbol{x})\boldsymbol{x} \leq K_{\mathrm{M}}|\boldsymbol{x}|^2$$

The two most common equations of state for the nonlinear functions S and κ are the Brooks–Corey model [25]

$$\kappa(s) = s^{\frac{2+3\lambda_1}{\lambda_1}}, \quad S(p) = \begin{cases} (-p/p_{\mathrm{M}})^{-\lambda_1} & p \leq p_{\mathrm{M}}, \\ 1 & \text{otherwise}, \end{cases} \tag{3.7}$$

for parameters $p_{\mathrm{M}} < 0, \lambda_1 \in (0,1)$ and the van Genuchten–Mualem model [116], for $p_{\mathrm{M}} \in \mathbb{R}$,

$$\kappa(\mathcal{S}(s)) = \kappa_{\mathrm{c}}\sqrt{\mathcal{S}(s)}(1-(1-\mathcal{S}(s)^{1/\lambda_2})^{\lambda_2})^2, \tag{3.8a}$$

$$S(p) = \begin{cases} S_{\mathrm{R}} + (S_{\mathrm{V}} - S_{\mathrm{R}})\left[(1+(-\alpha p)^{\frac{1}{1-\lambda_2}}\right]^{-\lambda_2} & p \leq p_{\mathrm{M}}, \\ S_{\mathrm{V}} & \text{otherwise}, \end{cases} \tag{3.8b}$$

where

$$\mathcal{S}(s) = \frac{s - S_{\mathrm{R}}}{S_{\mathrm{V}} - S_{\mathrm{R}}}, \tag{3.9}$$

with S_{V} the water content, S_{R} the residual water content, and κ_{c} the hydraulic conductivity.

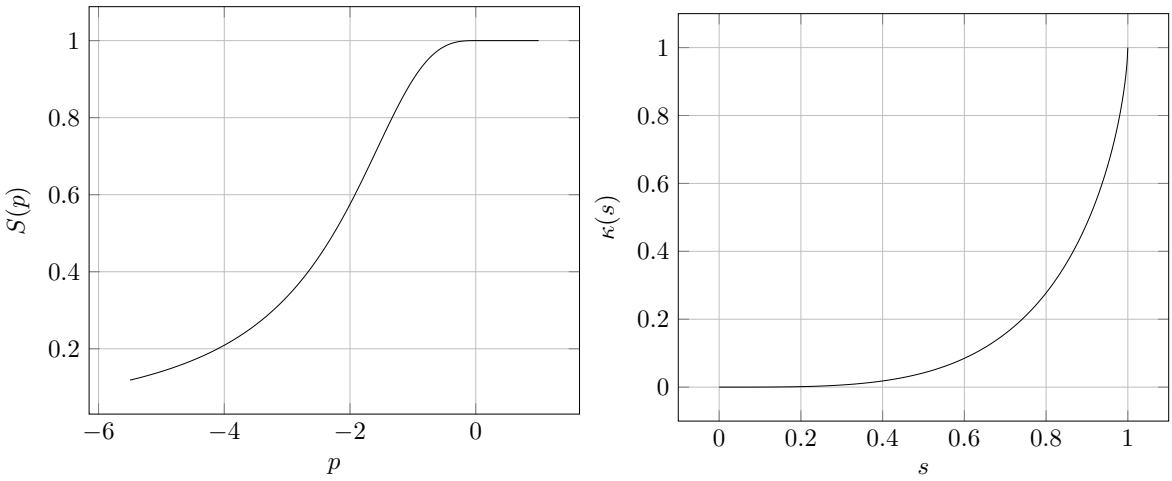


Figure 3.2: $[p_{\mathrm{M}} = 0, \lambda_2 = 0.66, S_{\mathrm{R}} = 0, S_{\mathrm{V}} = 1, \alpha = 0.551]$ Saturation and relative permeability functions for the van Genuchten–Mualem equation of state (3.8).

3.3 Difficulties related to the nonlinearities and proposed regularization

We first outline the possible difficulties a nonlinear solver can encounter in the context of the Richard's equation. We summarize them in the following list:

1. Hyperbolic degeneracy: if $\kappa = 0$ the terms containing spatial derivatives vanish and the PDE changes type from parabolic to an ODE.
2. Elliptic degeneracy: If $S' = 0$ the PDE changes type from parabolic to elliptic. This is typically not a serious problem for solvers in the pressure formulation that we have chosen.
3. For the van Genuchten–Mualem model (3.8), the derivative of the relative permeability function blows up, i.e., $\kappa'(\mathcal{S}(s)) \to \infty$ as $s \to S_{\mathrm{V}}$.
4. For the Brooks–Corey model (3.7), the saturation function $S(p)$ is non-differentiable at $p = p_{\mathrm{M}}$.

We thus replace the saturation function S and relative permeability function κ by regularized counterparts denoted by S_ϵ and κ_ϵ respectively. The parameter $\epsilon > 0$ determines the amount of added regularization, and is adaptively updated to minimize the incurred model error (see §3.5). The regularization is designed with the intention of alleviating the problems mentioned in the previous list. We now state the conditions on the regularization more precisely.

Assumption 3.3.1 (Assumptions on the regularization)**.** *The approximations S_ϵ to the saturation S and κ_ϵ to the relative permeability κ satisfy the following.*

(A1) the regularized relative permeability satisfies $\lim_{\epsilon\to 0} \kappa_\epsilon(s) = \kappa(s)$ *for all* $s \in [0, 1]$

(A2) The regularized saturation satisfies $\lim_{\epsilon\to 0} S_\epsilon(p) = S(p)$ *for all* $p \in \mathbb{R}$

(A3) There exists a constant $\underline{\kappa}_\epsilon > 0$ *such that* $\kappa_\epsilon(s) > \underline{\kappa}_\epsilon$ *for all* $s \in [0, 1]$.

(A4) The regularized saturation satisfies $S_\epsilon \in C^1(\mathbb{R})$

(A5) The regularized composite function satisfies $\kappa_\epsilon \circ S_\epsilon \in C^1(0,1)$

Assumptions (A1) and (A2) ensure that the regularized functions are good pointwise approximations of the true functions S and κ. The Assumption (A3), ensures that the regularized function κ_ϵ does not induce a hyperbolic degeneracy. The last two assumptions are smoothness requirements on the functions that are important, as derivatives of the functions appear in the bilinear form associated to the nonlinear solver as we will see below in §3.4.2.

We now introduce our chosen choices of regularization which satisfies Assumption 3.3.1. The choices depend on the chosen models, namely the Brooks–Corey model (3.7) and van Genuchten–Mualem (3.8). First, in the case of the Brooks–Corey model, the regularization of the relative permeability is simply

$$\kappa_\epsilon(s) = \kappa(s) + \epsilon. \tag{3.10}$$

This ensures that $\kappa_\epsilon > 0$ for all s and the smoothness requirements are already satisfied. The regularized saturation for the Brooks–Corey model is given by

$$S_\epsilon(p) = \begin{cases} S_\epsilon^k(p) & \text{if } |p - p_{\mathrm{M}}| < \epsilon, \\ S(p) & \text{otherwise.} \end{cases} \tag{3.11}$$

where $S_\epsilon^k(p)$ is determined by polynomial interpolation so that $S_\epsilon \in C^k(\mathbb{R})$ is k-times continuously differentiable. In particular we choose $k = 2$ so that Assumption (A4) is satisfied. A plot is given in Figure 3.3 for several values of ϵ. To satisfy Assumption (A5) we follow the approach in [14] where the relative permeability (3.8a) is replaced by a second degree polynomial near the critical point $\mathcal{S}(s) = 1$:

$$\kappa_\epsilon(\mathcal{S}) = \begin{cases} \kappa(s) + \epsilon, & \text{if } s \le 1 - \epsilon \\ \tilde{\kappa}(s) + \epsilon, & \text{otherwise} \end{cases} \tag{3.12}$$

where

$$\tilde{\kappa}(s) = \frac{\kappa''(1-\epsilon)}{2}(s - (1-\epsilon))^2 + \kappa'(1-\epsilon)(s - (1-\epsilon)) + \kappa(1-\epsilon) \tag{3.13}$$

see Figure 3.4 for a plot with a range of values of ϵ.

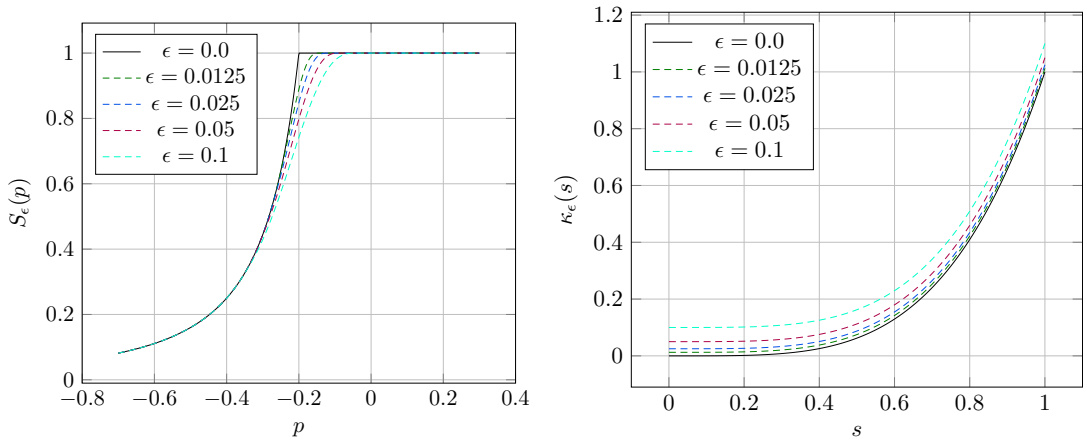


Figure 3.3: $[\lambda_1 = 2, p_M = -0.2, k = 2]$ Regularization of the relative permeability (left) (3.10) and of the saturation (3.11) (right) for the Brooks–Corey model (3.7).

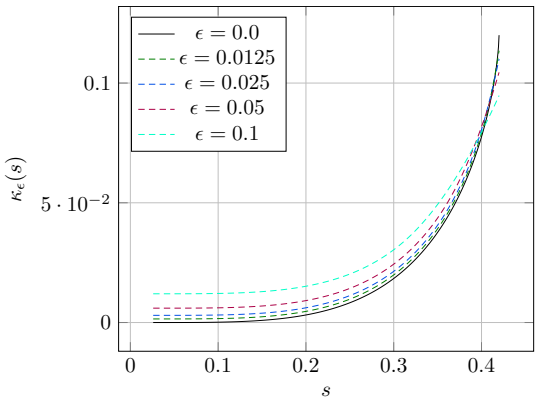


Figure 3.4: $[p_M = 0, \lambda_2 = 0.66]$ Regularization of the relative permeability (3.8) of the the van Genuchten–Mualem model (3.8). We do not regularize the saturation in this case, since it it already smooth.

3.4 Discrete problem and solution method

In this section we give details about the discretization strategy we employ to solve the Richards equation (3.1). We define the lowest order continuous finite element space for our trial space

$$V_h^{\mathrm{D}} = \left\{ u_h \in H_{\mathrm{D}}^1(\Omega),\ u_h|_K \in \mathcal{P}_1(K) \quad \forall\, K \in \mathcal{T}_h \right\}, \tag{3.14}$$

as well as the test space

$$V_h^0 = \left\{ u_h \in H_0^1(\Omega),\ u_h|_K \in \mathcal{P}_1(K) \quad \forall\, K \in \mathcal{T}_h \right\}. \tag{3.15}$$

For the time discretization we use the lowest order implicit method, i.e., the backward Euler method. We consider uniform time stepping with N time steps so that the interval $(0,T)$ is partitioned with fixed step size $\tau = T/N$ and the time points $\{t_n\}_{n=0}^N$ are given by $t_n = n\tau$ for all $n = 0,\dots,N$. The backward Euler problem for each $n \in \{1,\dots,N\}$ and a given $p_{n-1,h} \in V_h^{\mathrm{D}}$ is to find the approximate pressure $p_{n,h} \in V_h^{\mathrm{D}}$ satisfying

$$\tfrac{1}{\tau}(S(p_{n,h}) - S(p_{n-1,h}), \varphi_h) + (\boldsymbol{F}(p_{n,h}), \nabla\varphi_h) = (f(\cdot,t_n), \varphi_h) \quad \forall \varphi_h \in V_h^0, \tag{3.16}$$

where the flux function is defined as

$$\boldsymbol{F}(q) := \bar{\mathbf{K}}\kappa(S(q))[\nabla q + \boldsymbol{g}]. \tag{3.17}$$

3.4.1 Regularization

We also consider the regularized version of the problem. First, for a fixed timestep, we introduce a non-negative sequence $\{\epsilon^j\}_{j\geq 1}$ such that ϵ^1 is fixed at the beginning of the timestep (see §3.5). The regularized problem is then: given $p_{n-1,h}^{\bar{j}} \in V_h^{\mathrm{D}}$ find $p_{n,h}^j \in V_h^{\mathrm{D}}$ satisfying

$$\tfrac{1}{\tau}(S_{\epsilon^j}(p_{n,h}^j) - S_{\epsilon^j}(p_{n-1,h}^{\bar{j}}), \varphi_h) + (\boldsymbol{F}_{\epsilon^j}(p_{n,h}^j), \nabla\varphi_h) = (f(\cdot,t_n), \varphi_h) \quad \forall \varphi_h \in V_h^0, \tag{3.18}$$

where the corresponding regularized flux is given by

$$\boldsymbol{F}_{\epsilon^j}(q) := \bar{\mathbf{K}}\kappa_{\epsilon^j}(S_{\epsilon^j}(q))[\nabla q + \boldsymbol{g}], \tag{3.19}$$

and $\bar{j}$ is a stopping index that will be defined in §3.5.

3.4.2 Linearization

Due to the nonlinear nature of the problem (3.18), linear iterations are usually used to approximate $p_{n,h}^j$. To this end, we consider the following linearized problem: given an initial guess $p_{n,h}^{j,k-1}$, find $p_{n,h}^{j,k} \in V_h^{\mathrm{D}}$ such that

$$\tfrac{1}{\tau}(S_{\epsilon^j}(p_{n,h}^{j,k-1}) - S_{\epsilon^j}(p_{n-1,h}^{\bar{j},\bar{k}}), \varphi_h) + \tfrac{1}{\tau}(L(p_{n,h}^{j,k} - p_{n,h}^{j,k-1}), \varphi_h) + (\boldsymbol{F}_{\epsilon^j}^k, \nabla\varphi_h) = (f(\cdot,t_n), \varphi_h) \quad \forall \varphi_h \in V_h^0,$$

where $\bar{j}$ and $\bar{k}$ are stopping indices that will be defined in §3.5 and the linearized flux is given by

$$\boldsymbol{F}_{\epsilon^j}^k := \bar{\mathbf{K}}\kappa_{\epsilon^j}(S_{\epsilon^j}(p_{n,h}^{j,k-1}))[\nabla p_{n,h}^{j,k} + \boldsymbol{g}] + \boldsymbol{\xi}(p_{n,h}^{j,k} - p_{n,h}^{j,k-1}). \tag{3.20}$$

Here, $(L, \boldsymbol{\xi}) \in \mathbf{L}^\infty(\Omega; \mathbb{R}^{d+1})$ depends on the specific scheme used. For the case of the modified Picard iteration [31], we have that

$$L := S'_{\epsilon^j}(p_{n,h}^{j,k-1}), \quad \boldsymbol{\xi} := \mathbf{0}. \tag{3.21}$$

For Newton's method, we have that

$$L := S'_{\epsilon^j}(p_{n,h}^{j,k-1}), \quad \boldsymbol{\xi} := \bar{\mathbf{K}}(\kappa_{\epsilon^j} \circ S_{\epsilon^j})'(p_{n,h}^{j,k-1})[\nabla p_{n,h}^{j,k-1} + \boldsymbol{g}]. \tag{3.22}$$

Remark 3.4.1 (Appearance of derivatives in the linearization)**.** *We note that the for both the modified Picard scheme Newton's method, the derivative $S'(p_{n,h}^{j,k-1})$ appears. Additionally, in the case of Newton's method the derivative of the composite functions $(\kappa_{\epsilon^j} \circ S_{\epsilon^j})'(p_{n,h}^{j,k-1})$ appears. This is the motivation for the regularity requirements we impose on the regularization, i.e., Assumptions (A4) and (A5).*

3.4.3 A posteriori component error estimators by flux reconstruction

The key to our a posteriori error estimators will be a postprocessed approximation $\boldsymbol{\sigma}_{n,h}^{j,k}$ of the flux (3.20) that satisfies $\boldsymbol{\sigma}_{n,h}^{j,k} \subset \boldsymbol{H}(\mathrm{div}, \Omega)$. The main tool to achieve this is the Raviart–Thomas–Nédélec (RTN) finite element space [23]. We first introduce the lowest order broken RTN space,

$$\boldsymbol{RT}_0(\mathcal{T}_h) := \{\boldsymbol{v}_h \in [L^2(\omega)]^d : \boldsymbol{v}_h|_K \in [\mathcal{P}_0]^d + \boldsymbol{x}\mathcal{P}_0, \forall K \in \mathcal{T}_h\}, \tag{3.23}$$

and the $\boldsymbol{H}(\mathrm{div}, \Omega)$-conforming space

$$\boldsymbol{RT}_0(\Omega) := \boldsymbol{RT}_0(\mathcal{T}_h) \cap \boldsymbol{H}(\mathrm{div}, \Omega). \tag{3.24}$$

Our general approach is in the spirit of equilibrated flux reconstruction. The method of flux reconstruction in the context of a posteriori error analysis has origins in the works of Prager and Synge [101] as well as in Ladevèze and Leguillon [84], Destuynder and Métivet [41], Braess and Schöberl [21], and Ern and Vohralík [53]. However, in this work we do not consider a full equilibration by solving local minimization problems, but rather a flux based on averaging and prescription of the degrees of freedom in $\boldsymbol{H}(\mathrm{div}, \Omega)$ as in [121, 51]. In general this type of estimator satisfies the equilibration with the external load in a weak sense.

First, we introduce some additional notation for the mesh. Let $\mathcal{F}$ denote the set of faces in the mesh and for a face $F \in \mathcal{F}$ let $\mathcal{T}_F$ denote the edge patch of F, i.e.,

$$\mathcal{T}_F := \{K \in \mathcal{T}_h : F \subset \overline{K}\}. \tag{3.25}$$

Then we define the reconstructed flux $\boldsymbol{\sigma}_{n,h}^{j,k} \in \boldsymbol{RT}_0(\Omega)$ by

$$\frac{1}{|F|} \int_F \boldsymbol{\sigma}_{n,h}^{j,k} \cdot \boldsymbol{n}_F \, \mathrm{d}S = \frac{1}{|\mathcal{T}_F||F|} \sum_{K \in \mathcal{T}_F} \int_F -\boldsymbol{F}_{\epsilon^j}^k \cdot \boldsymbol{n}_F \, \mathrm{d}S \quad \forall F \in \mathcal{F}, \tag{3.26}$$

where $|\mathcal{T}_F|$ denotes the cardinality of $\mathcal{T}_F$.

The conditions in (3.26) totally determine the function $\boldsymbol{\sigma}_{n,h}^{j,k}$, see, e.g., [49]. We thus define the following estimators with the help of the reconstructed flux (3.26):

$$\eta_{\mathrm{dis}}^{\ell,j,k} := \|\boldsymbol{F}_{\epsilon^j}^k + \boldsymbol{\sigma}_{n,h}^{j,k}\| \qquad \text{(discretization)}, \tag{3.27a}$$

$$\eta_{\mathrm{lin}}^{\ell,j,k} := \|\boldsymbol{F}_{\epsilon^j}(p_{n,h}^{j,k}) - \boldsymbol{F}_{\epsilon^j}^k\| \qquad \text{(linearization)}, \tag{3.27b}$$

$$\eta_{\mathrm{reg}}^{\ell,j,k} := \|\boldsymbol{F}(p_{n,h}^{j,k}) - \boldsymbol{F}_{\epsilon^j}(p_{n,h}^{j,k})\| \qquad \text{(regularization)}. \tag{3.27c}$$

Remark 3.4.2 (Choice of estimators)**.** *We take inspiration for the discretization estimator from [94], where our estimator can be thought of a simplified version of* η^{F}*. We make this choice because the current definition is very cheap to compute as it does not require the solution of local problems. This decomposition into component estimators is very much inspired by those established in [53, 58]. In [53], a decomposition was established that identified errors associated with the discretization, linearization, algebra, and quadrature. These estimators were then used to define stopping criteria for the nested nonlinear and linear solvers. More recently in [58], we consider a regularized problem and introduce a corresponding regularization estimator, leading to the same (at least in spirit) choice of estimators as in the current work. In [58] we rigorously prove that the estimators tend to zero in their respective limits, i.e.,* $\eta_{\mathrm{reg}}^{\ell,j,k}$ *tends to zero as* $j \to \infty$ *and* $\eta_{\mathrm{lin}}^{\ell,j,k}$ *tends to zero as* $k \to \infty$*.*

3.5 Adaptive algorithm

In this section we present an adaptive algorithm for iteratively solving the system of nonlinear algebraic equations (3.16), Algorithm 4. For a given timestep t_n, the algorithm constructs a sequence of regularized problems, with regularization parameter ϵ^j, and linearization iterations indexed by k, producing intermediate solutions $p_{n,h}^{j,k}$ as per §3.4.2 and § 3.4.1. The algorithm takes some user-specified parameters, starting with an initial regularization parameter $\epsilon^1 > 0$ and an initial contraction factor $C_\epsilon^1 \in (0,1)$. We take inspiration from Algorithm 1 in [58] to define the following stopping criteria, where bars denote the stopping indices,

$$\eta_{\mathrm{lin}}^{n,j,\bar{k}} < \gamma_{\mathrm{lin}} \eta_{\mathrm{reg}}^{n,j,\bar{k}}, \tag{3.28a}$$

$$\eta_{\mathrm{reg}}^{n,\bar{j},\bar{k}} < \gamma_{\mathrm{reg}} \eta_{\mathrm{dis}}^{n,\bar{j},\bar{k}}, \tag{3.28b}$$

where $\gamma_{\mathrm{reg}}, \gamma_{\mathrm{lin}} > 0$ are further user-specified parameters. The first criterion (3.28a) states that the linearization procedure should not continue on a given regularized problem if it has sufficiently converged. The second criterion (3.28b) states that, on a given timestep t_n, the error introduced by regularization should only be γ_{reg} times smaller than the inherent error due to discretization.

In more details, the algorithm proceeds as follows: we start on a given timestep with the initial regularization parameter ϵ^j and contraction factor C_ϵ^j. We proceed to iterate in the linearization until the first stopping criterion (3.28b) is satisfied. However, we also have a safety measure (line 12 of the algorithm) to check whether the linearization error does not increase between the previous and current linearization iterates $k-1$ and k. If this is the case, we revert the regularization parameter and reset the approximate pressure. We also increase the current contraction factor C_ϵ^j which limits the amount we decrease the regularization parameter between the steps j and $j+1$. This strategy is comparable to common practice of cutting the timestep to provide a better initial guess, but the advantage here is that we only "go back" one value of the regularization parameter and not to the beginning of the timestep.

We also remark that the initial guess can be taken as $p_{0,h}^{0,0} := S^{-1}(s_0)$ in the regime where S is invertible, namely $s_0 < 1$. For the points $x \in \Omega$ where $s_0(x) = 1$, we simply take the initial guess $p_{0,h}^{0,0}(x) := p_{\mathrm{M}}(x)$.

Algorithm 4: Adaptive regularization for the Richard equation

Initialization : Choose an initial guess $p_{0,h}^{0,0} \in V_h^{\mathrm{D}}$ and initialize the time step counter $n := 0$

Parameters : $\gamma_{\mathrm{reg}}, \gamma_{\mathrm{rin}}, \epsilon^1, C_\epsilon^1 \in (0,1)$

1 **while** $t_n < T$ **do**
2 Update $n := n+1$
3 Initialize $j := 0, \bar{k} := 0$
4 **Loop** for regularization
5 Increment $j := j+1$
6 Initialize $k := 0$
7 $\eta_{\mathrm{lin}}^{n,j,k} := \infty$
8 **Loop** for linearization
9 Increment $k := k+1$
10 Solve for $p_{n,h}^{j,k}$ in (3.20)
11 Compute the estimators (3.27)
12 **if** $\eta_{\mathrm{lin}}^{n,j,k} > \eta_{\mathrm{lin}}^{n,j,k-1}$ **then**
13 Reset $p_{n,h}^{j,k} := p_{n,h}^{j-1,\bar{k}}$
14 Increase $C_\epsilon^{j+1} := \sqrt{C_\epsilon^j}$
15 Decrease $\epsilon^{j+1} := (C_\epsilon^j / C_\epsilon^{j+1})\epsilon^j$
16 **go to line** 4
17 **end**
18 **until** $\eta_{\mathrm{lin}}^{n,j,k} < \gamma_{\mathrm{lin}} \eta_{\mathrm{reg}}^{n,j,k}$
19 Update $\bar{k} := k$
20 Decrease $\epsilon^{j+1} := C_\epsilon^j \epsilon^j$
21 **until** $\eta_{\mathrm{reg}}^{n,j,\bar{k}} < \gamma_{\mathrm{reg}} \eta_{\mathrm{dis}}^{n,j,\bar{k}}$
22 Update $\bar{j} := j$
23 $p_{n,h}^{0,0} := p_{n,h}^{\bar{j},\bar{k}}$
24 **end**
25 **return** $\{p_{n,h}^{\bar{j},\bar{k}}\}_{n=1}^N$

3.6 Numerical experiments

In this section we detail numerical experiments using our adaptive regularization of Algorithm 4. In particular, we consider three rather involved examples where a plain Newton solver struggles to converge. In all cases our adaptive algorithm succeeds. All numerical experiments are conducted with the help of the `Gridap.jl` library [7, 118] in the Julia programming language. For all the experiments, we take the linearization parameters $\gamma_{\mathrm{reg}} = 0.2, \gamma_{\mathrm{lin}} = 0.3$, $C_\epsilon^1 = 0.1$, and $\epsilon^1 = 0.1$. For comparison, we also test the unregularized Newton's method (corresponding to taking $\epsilon^1 = 0$ in Algorithm 4), and the modified Picard scheme (3.21) In the unregularized case, instead of criterion (3.28a), we ensure that

$$\eta_{\mathrm{lin}}^{\ell,j,k} < 1\mathrm{e}{-6}. \tag{3.29}$$

Remark 3.6.1 (Choice of the stopping criterion in the unregularized case)**.** *We use a fixed*

stopping criterion for the linearization in (3.29) because we would like to compare our adaptive strategy with an unadaptive one, which is the much more common approach. For example, even in [114] where the solver is chosen adaptively, the authors use a fixed stopping criterion for terminating the linearization procedure. Namely, their criterion ensures that the difference of two consecutive iterates of the approximate measured in a iteration dependent norm is less than 1e-7. Moreover, our choice of the value 1e-6 is partially related to the observation that in Figure 3.6 (described in detail below in §3.6.1), the linearization estimator $\eta_{\text{lin}}^{\ell,j,k}$ *is on the order of 1e-5 at the end of the iterations. Thus, we choose a value that at least ensures this level of accuracy for all the numerical experiments.*

3.6.1 Strictly unsaturated medium

In this test case, we seek to reproduce the results obtained in [114, §4.1]. This means we have the following data:

- $\Omega = \Omega_1 \cup \Omega_2$, $\Omega_1 = (0,1) \times (0,1/4], \Omega_2 = (0,1) \times (1/4,1)$
- Uniform mesh with $40 \times 40 \times 2$ elements
- $T = 1$
- $\tau = 1$
- $\Gamma_{\text{D}} = \partial\Omega \cap \{y = 1\}$
- $\Gamma_{\text{N}} = \partial\Omega \setminus \Gamma_{\text{D}}$
- $\boldsymbol{g} = (0,1)^T$
- $f(x,y) = \begin{cases} 0 & (x,y) \in \Omega_1 \\ 0.06\cos(\frac{4}{3}\pi y)\sin(x) & (x,y) \in \Omega_2. \end{cases}$
- $p_0(x,y) = \begin{cases} -y - 1/4 & (x,y) \in \Omega_1 \\ -4 & (x,y) \in \Omega_2, \end{cases}, \quad s_0 = S(p_0)$
- $p_{\text{D}} = p_0|_{\Gamma_{\text{D}}}$

We use the van Genuchten-Mualem model (3.8) with the parameters specified in Figure 3.5. Please note that there is only 1 timestep. We first plot the approximate pressure at the final step of both Algorithm 4 as well as the modified Picard iteration with no regularization, Figure 3.5. We observe that the two not only match well but are also comparable with the results in [114, §4.1]. In this case, Newton's method without regularization diverged, which is consistent with what is reported in [114, §4.1].

We now look more carefully at the evolution of the estimators in the adaptive algorithm for this example. In Figure 3.6, we plot the component estimators as a function of cumulative linearization steps. The components all begin on the order of 0.1 on the first iteration. We see that the linearization estimator converges very rapidly for a given value of the regularization parameter $\epsilon^j := \epsilon^1 = 0.1$, then $\epsilon^j := 0.01, 0.001$ and 0.0001. Furthermore, once we lower the regularization parameter, the regularization component estimator clearly decreases. On the final iteration, we see the discretization and regularization estimators stabilize with a constant gap between the two.

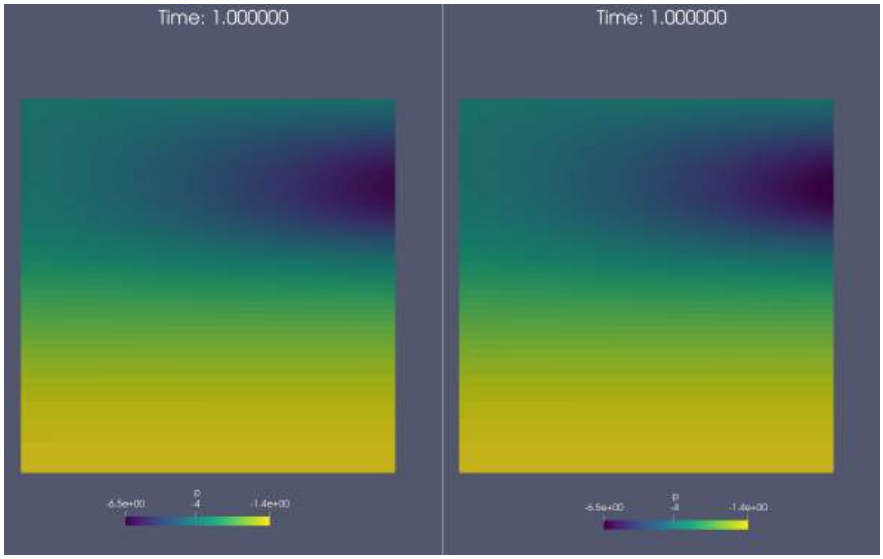


Figure 3.5: [§3.6.1, van Genuchten–Mualem model (3.8) with $p_{\mathrm{m}} = 0, S_{\mathrm{R}} = 0.026, S_{\mathrm{V}} = 0.42, \kappa_{\mathrm{c}} = 0.12, \alpha = 0.551, \lambda_2 = 0.655$, solver parameters: $\gamma_{\mathrm{reg}} = 0.2, \gamma_{\mathrm{lin}} = 0.3$, $C_\epsilon^1 = 0.1$] Approximate pressure $p_{n,h}^{\bar{j},\bar{k}}$ for the problem in §3.6.1 using Algorithm 4 with Newton's method and adaptive regularization $\epsilon^1 = 0.1$ (left) and modified Picard with no regularization $\epsilon^1 = 0$ (right).

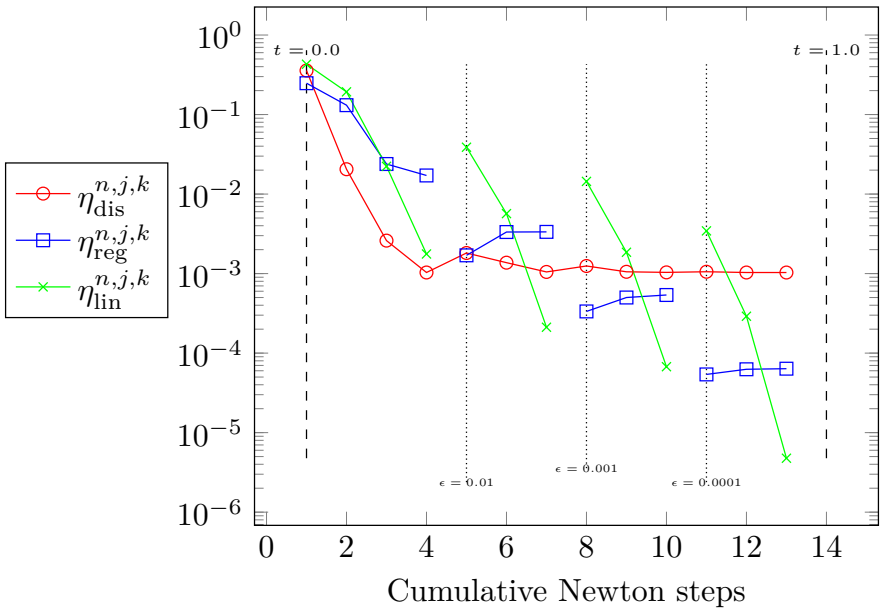


Figure 3.6: [§3.6.1, van Genuchten–Mualem model (3.8) with $p_{\mathrm{m}} = 0, S_{\mathrm{R}} = 0.026, S_{\mathrm{V}} = 0.42, \kappa_{\mathrm{c}} = 0.12, \alpha = 0.551, \lambda_2 = 0.655$, solver parameters: $\gamma_{\mathrm{reg}} = 0.2, \gamma_{\mathrm{lin}} = 0.3$, $C_\epsilon^1 = 0.1$, $\epsilon^1 = 0.1$] Evolution of the component estimators (3.27) for the Algorithm 4 with Newton's method and adaptive regularization with $\epsilon^1 = 0.1$ applied to the test problem in §3.6.1.

3.6.2 Injection test

This test is inspired by the one presented in [22, §4.1]. In particular, we use the following model parameters:

- $\Omega = (0,1)^2$
- $T = 1.0$
- $\tau = 2.82\text{e}{-2}$
- Quasi uniform mesh with $h = 2.82\text{e}{-2}$
- $\Gamma_{\text{D}} = \{(x_1, x_2) | x_1 \in (0, 0.3), x_2 = 1\}$
- $\Gamma_{\text{N}} = \partial\Omega \setminus \Gamma_{\text{D}}$
- $\boldsymbol{g} = (0, -1)^T$
- $f = 0$
- $p_0 = -1, \quad s_0 = S(p_0)$
- $p_{\text{D}} = 1$

We use the Brooks–Corey model (3.7) with parameters specified in Figure 3.8. We note firstly that there is a inconsistency between the trace of p_0 and the imposed boundary condition p_{D} at $t = 0$. This is mathematically valid, but can cause problems for the solver as we shall see shortly. The domain is initially "mildly dry" with $S(p_0) = s_0 = 0.027$. We remark that we were not able to consider a smaller value of s_0 as was done in, e.g., [22, 14]. It would likely be necessary to implement their variable switching strategy for this, which we discuss in §3.7. However, the current test still remains challenging for Newton's method.

In Figure 3.7, the total stepwise and cumulative iterations are plotted for Algorithm 4 using Newton's method with and without regularization as well as modified Picard without regularization. First of all, it is clear that Newton's method without regularization is not feasible and we cut the solver after 300 iterations on the first step. Next, we note that Modified Picard takes consistently more iterations than Newton solver with regularization, resulting in an approximate 3.3x speedup at the end of the simulation (1004 iterations for modfied Picard vs 297 for the regularized Newton solver). We also note that the number of iterations is somewhat less stable for modified Picard with peaks of 56 iterations at $t = 0.42$, and 48 iterations at $t = 0.22$. In contrast the regularized Newton solver takes no more than 13 iterations per timestep. In fact, this only occurs at the beginning of the simulation as we will not explore in more detail.

In Figure 3.8 we see that Newton's method has trouble converging for the reguarization parameter $\epsilon^j = 0.1$ and the linearization estimator increases during the second and third timesteps, thus triggering the if statement on line 12 of Algorithm 4. Indeed, we see consequently the algorithm recovers by simultaneously increasing the regularization parameter to $\epsilon^1 := \epsilon^1 / C_\epsilon^1$ and then increasing the contraction factor $C_\epsilon^1 = \sqrt{C_\epsilon^0}$. This combination allows the estimator to converge on the following series of regularized problem until the stopping criterion (3.28b) is achieved and the solver advances to the next timestep.

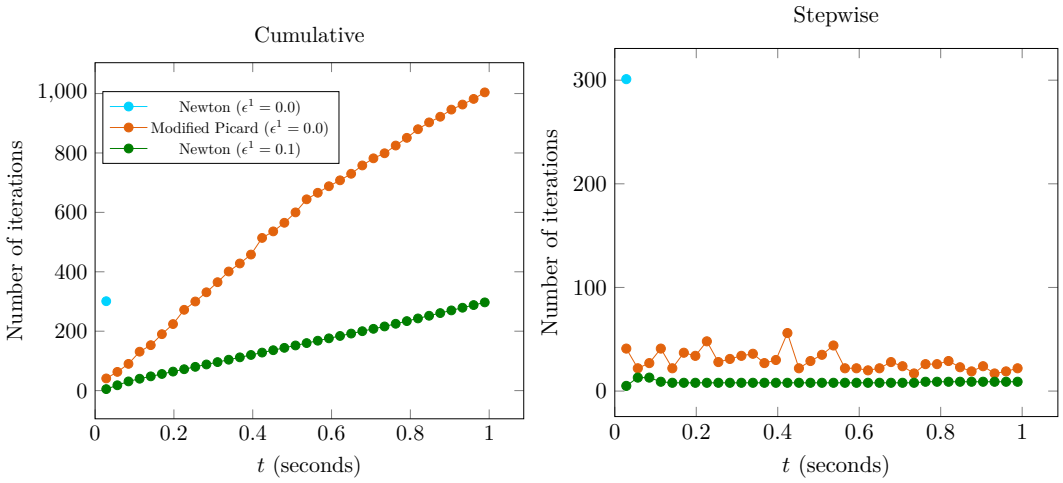


Figure 3.7: §3.6.2, Brooks–Corey model (3.7) with $p_{\mathrm{M}} = -0.2, \lambda_1 = 2$, solver parameters $\gamma_{\mathrm{reg}} = 0.2, \gamma_{\mathrm{lin}} = 0.3$] Comparison of the total cumulative and stepwise iterations for the three strategies.

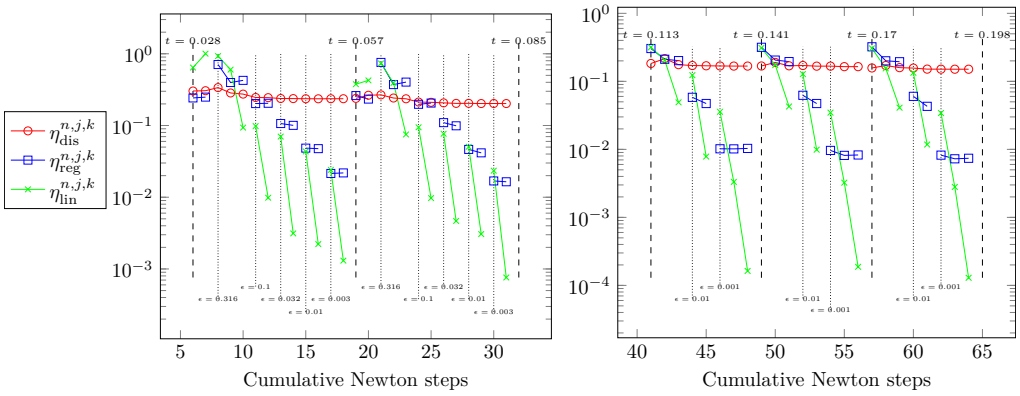


Figure 3.8: [§3.6.2, Brooks–Corey model (3.7) with $p_{\mathrm{M}} = -0.2, \lambda_1 = 2.239$, solver parameters $\gamma_{\mathrm{reg}} = 0.2, \gamma_{\mathrm{lin}} = 0.3, C^1_\epsilon = 0.1, \epsilon^1 = 0.1$] Plots of the evolution of the estimators on the second and third timesteps (left) and of the fifth, sixth and seventh timesteps (right).

We now consider the effect of the regularization on the solution. In particular, in Figure 3.9, we compare side by side plots of the saturation profile for the regularized and unregularized solutions. The profiles match well, and we also note that the regularized profile appears smoother at the interface.

3.6.3 Realistic test

In this test, we take inspiration from [94, §6.3] by using the following model parameters:

- $\Omega = (0, 1)^2$
- $T = 1$
- $\tau = 2.02\mathrm{e}{-2}$
- Quasi uniform mesh with $h = 2.02\mathrm{e}{-2}$

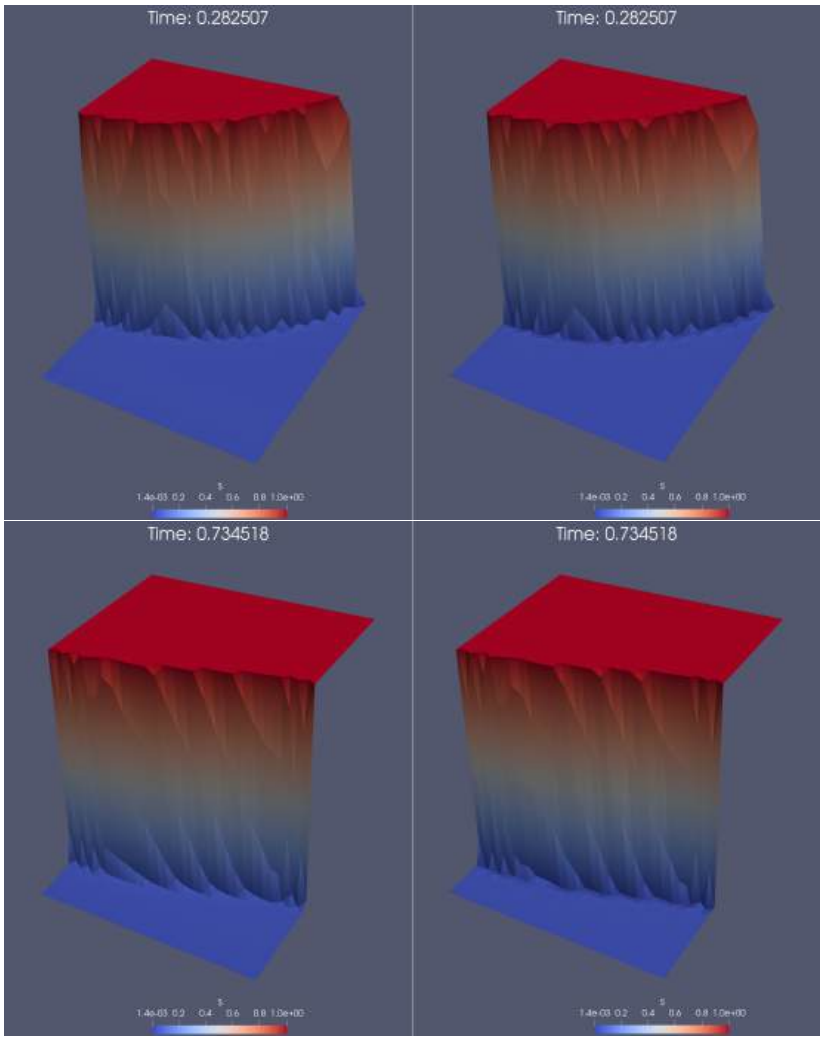


Figure 3.9: [§3.6.2, Brooks–Corey model (3.7) with $p_{\mathrm{M}} = -0.2, \lambda_1 = 2.239$, solver parameters $\gamma_{\mathrm{reg}} = 0.2, \gamma_{\mathrm{lin}} = 0.3, C_\epsilon^1 = 0.1$] Two snapshots comparing the evolution of the saturation field $s = S(p_{n,h}^{\bar{j},\bar{k}})$ using the Algorithm 4 with Newton's method and adaptive regularization $\epsilon^1 = 0.1$ (left) and modified Picard with no regularization $\epsilon^1 = 0$ (right).

- $\boldsymbol{g} = (-1, 0)^T$
- $p_{\mathrm{L}}(x) = \left(\frac{p_{\mathrm{out}} - p_{\mathrm{in}}}{0.5}\right) x$
- $\mathbf{Q} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$
- $K_\phi = 0.1$
- $p_{\mathrm{out}} = -2.0$
- $p_{\mathrm{in}} = -0.2$
- $p_{\mathrm{D}} = p_0|_{\Gamma_{\mathrm{D}}}$

where the initial condition and boundary conditions are fully specified with the help of the schema in Figure 3.10. We use the Brooks–Corey model (3.7) with parameters specified in Figure 3.12.

We begin by comparing the stepwise and cumulative number of iterations in Figure 3.11. We first remark that Newton's method without regularization takes an unreasonable number of iterations on the first step (we stop the solver at 300 iterations). Modified Picard is able to finish the simulation but has some big peaks namely 208 iterations for $t = 0.82$, 101 iterations at $t = 0.54$, and 77 iterations at $t = 0.78$. In contrast, the number of iterations per step for the regularized Newton solver does not exceed 20. This gain is reflected clearly when comparing the cumulative number of iterations where by the end, modified Picard has taken almost 5-times as many iterations as the regularized Newton solver (2359 vs. 576).

To better understand how the adaptive algorithm works, we refer to Figure 3.12. In the left figure we see that no problems are encountered at the timesteps $t = 0$ through $t = 0.061$. In the right figure we plot the estimators around the time of the contact with the interface at $t = 0.445$–0.486 and we see that the linearization estimator begin to increase on the first timestep for $\epsilon = 0.001$. The condition of the if statement on line 12 is then true, resetting to the result at the previous value of $\epsilon = 0.01$, and increasing the contraction factor C_ϵ^j thereby decreasing the "distance" between the two consecutive regularized problems. This allows the algorithm to proceed, albeit with more intermediate values of ϵ, to the end of the timestep.

Finally, we compare snapshots of the saturation for two timesteps $t = 0.40$ and $t = 0.95$ in Figure 3.13. As in the previous examples, the two profiles are comparable with the regularized version appearing smoother at the boundary of the evolving interface.

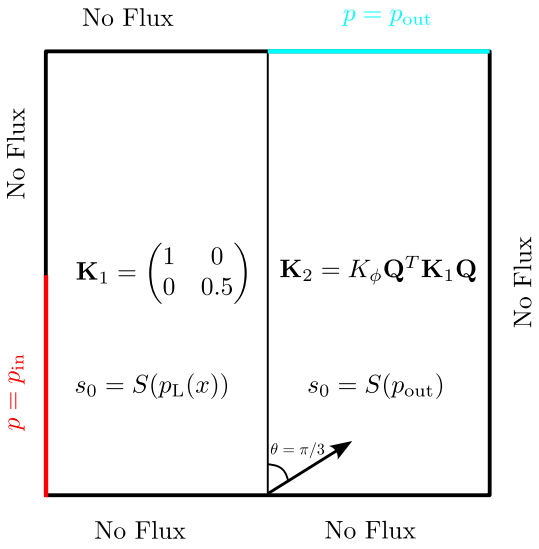


Figure 3.10: Schematic of the boundary and initial conditions for the test problem considered in §3.6.3.

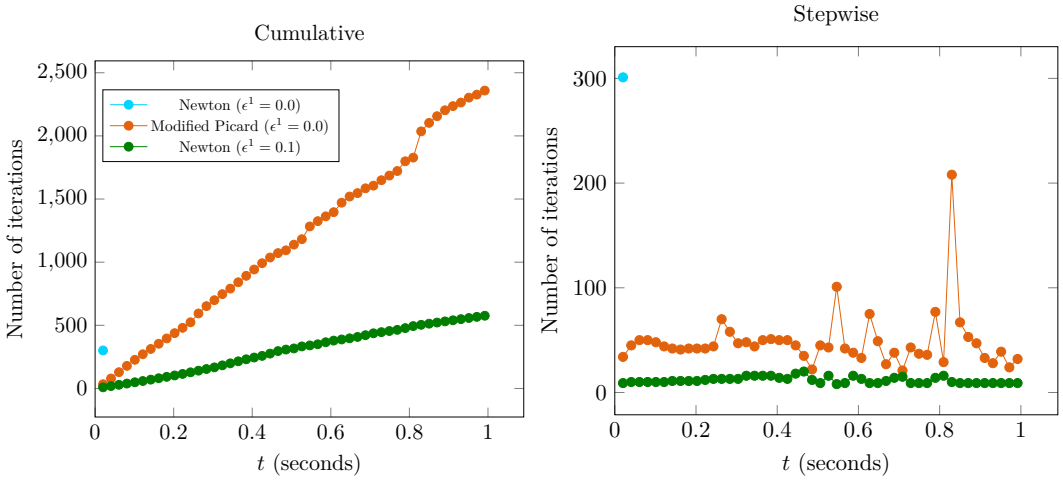


Figure 3.11: [§3.6.3, Brooks–Corey model (3.7) with $p_{\text{M}} = -0.2, \lambda_1 = 2$, solver parameters $\gamma_{\text{reg}} = 0.2, \gamma_{\text{lin}} = 0.3, C_\epsilon^1 = 0.1, \epsilon^1 = 0.1$] Comparison of the total cumulative and stepwise iterations for the three strategies.

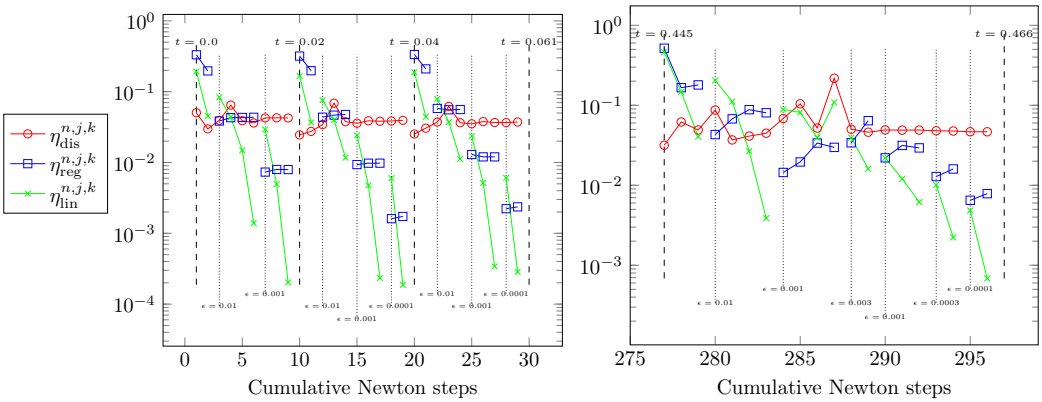


Figure 3.12: [§3.6.3, Brooks–Corey model (3.7) with $p_{\text{M}} = -0.2, \lambda_1 = 2$, solver parameters $\gamma_{\text{reg}} = 0.2, \gamma_{\text{lin}} = 0.3, C_\epsilon^1 = 0.1, \epsilon^1 = 0.1$] Evolution of the estimators on the first and second timesteps (upper) and of the 22nd, 23rd and 24th timesteps (lower).

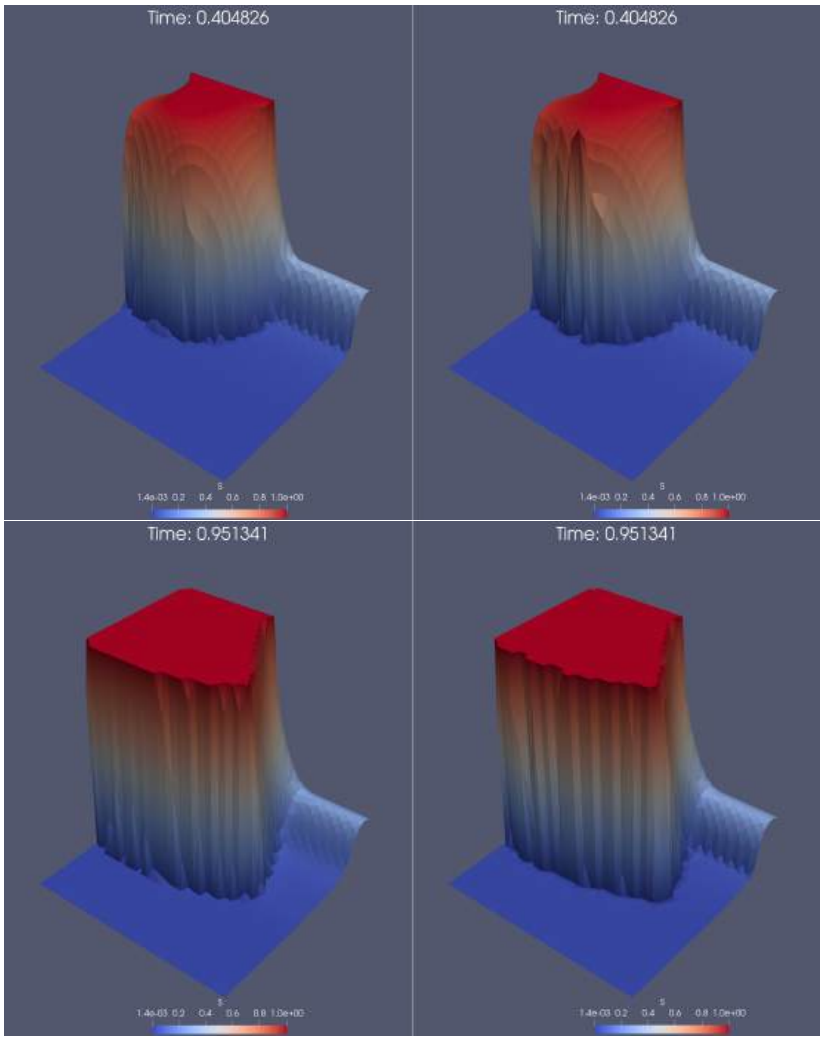


Figure 3.13: [§3.6.2, Brooks–Corey model (3.7) with $p_{\text{M}} = -0.2, \lambda_1 = 2$, solver parameters $\gamma_{\text{reg}} = 0.2, \gamma_{\text{lin}} = 0.3, C_\epsilon^1 = 0.1$] Two snapshots comparing the evolution of the saturation field $s = S(p_{n,h}^{\bar{j},\bar{k}})$ using the adaptive regularization Algorithm 4 with Newton's method and $\epsilon^1 = 0.1$ (left) and modified Picard with $\epsilon^1 = 0$ (right).

3.7 Conclusions and future work

In this work, we introduced an adaptive regularization algorithm to iteratively solve the Richards equation. The algorithm works with regularized versions of the nonlinearities present in Richards equation to improve the performance of Newton's method in solving the resulting nonlinear system. The proposed algorithm adaptively controls the level of regularization based on a posteriori error estimators. The proposed adaptive algorithm is able to converge where the unregularized version takes excessively many iterations. Furthermore, we compare the performance with the modified Picard scheme which is specific to the Richards equation. In all test cases the adaptive algorithm with regularization outperforms modified Picard and produces a perceptibly comparable solution.

In terms of future work, we note that our proposed algorithm is not able to converge in the dry regime $s \ll 1$. This is a well known difficulty and has been shown to be ameliorated by variable switching techniques, see [22] and references therein. We would like to emphasize that our strategy is not incompatible with these methods, and that we would like to test a combination of regularization and variable switching to tackle even more difficult benchmark problems. Another future direction would be to study two independent regularization parameters for the functions κ_ϵ and S_ϵ in the case of the Brooks–Corey model.

Chapter 4

Implementation of the equilibrated flux

In this section, we introduce a memory efficient, parallel implementation of the equilibrated flux, already used extensively throughout this thesis. To simplify notation, we consider construction the equilibrated flux for the Poisson equation with homogeneous boundary conditions, i.e., for a polygonal domain $\Omega \subset \mathbb{R}^2$,

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.1}$$

4.1 Notation and problem statement

In this context we introduce all the necessary notation to describe the equilibrated flux and is subsequent resolution. We consider a fixed triangular conforming mesh $\mathcal{T}_h = \cup_K \{K\}$ such that $\cup_{K\in\mathcal{T}_h} K = \overline{\Omega}$. Let $\mathcal{E}_h$ denote the set of edges in the mesh and $\mathcal{V}_h$ denote the set of vertices. We decompose these into the of interior edges $\mathcal{E}_h^{\text{int}}$ (respectively interior vertices $\mathcal{V}_h^{\text{int}}$) contained in the interior of Ω and boundary edges $\mathcal{E}_h^{\text{ext}}$ (respectively boundary vertices $\mathcal{V}_h^{\text{ext}}$) lying on $\partial\Omega$. We introduce the nodal patch $\mathcal{T}_{\mathbf{a}}$ of a node $\mathbf{a} \in \mathcal{V}_h$ by $\mathcal{T}_{\mathbf{a}} := \{K \in \mathcal{T}_h : \mathbf{a} \in K\}$. We denote the corresponding domain of $\mathcal{T}_{\mathbf{a}}$ by $\omega_{\mathbf{a}} := \{\boldsymbol{x} \in \operatorname{int} K : K \in \mathcal{T}_{\mathbf{a}}\} \subset \Omega$. See Figure 4.1 for an illustration. We assume that we work with a fixed polynomial degree $k \geq 0$. Next, for an element $K \in \mathcal{T}_h$, we introduce the Raviart–Thomas–Nédélec [23] mixed finite element space,

$$\boldsymbol{V}_h(K) := [\mathbb{P}_k(K)]^d + \boldsymbol{x}\mathbb{P}_k(K), \tag{4.2}$$

as well as the $\boldsymbol{H}(\operatorname{div}, \Omega)$-conforming counterpart for a collection of elements $\mathcal{T} \subseteq \mathcal{T}_h$, and its corresponding subdomain $\omega \subseteq \Omega$

$$\boldsymbol{V}_h(\omega) := \{\boldsymbol{v}_h \in \boldsymbol{H}(\operatorname{div}, \omega) : \boldsymbol{v}_h\big|_K \in \boldsymbol{V}_h(K) \quad \forall K \in \mathcal{T}\}, \tag{4.3}$$

as well as the broken space

$$Q_h(\omega) := \{v \in L^2(\omega) : v\big|_K \in \mathbb{P}_k(K) \quad \forall K \in \mathcal{T}\}. \tag{4.4}$$

Let $H^1(\mathcal{T}_h)$ be the broken (elementwise) H^1 space. The following definition will be the central object of study for this chapter.

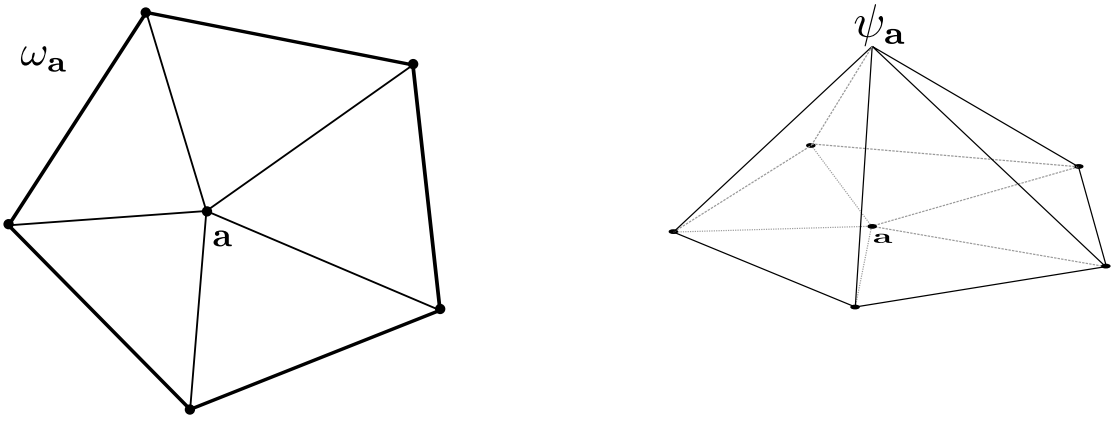


Figure 4.1: A nodal patch of elements and its associated hat function.

Definition 4.1.1 (Equilibrated flux)**.** *Let $u_h \in H^1(\mathcal{T}_h)$ satisfy the hat function orthogonality*

$$(\nabla u_h, \nabla \psi_{\mathbf{a}})_\omega = (f, \psi_{\mathbf{a}})_{\omega_{\mathbf{a}}} \quad \forall \mathbf{a} \in \mathcal{V}_h^{\text{int}}. \tag{4.5}$$

For each node $\mathbf{a} \in \mathcal{V}_h$, determine $\boldsymbol{\varsigma}_{\mathbf{a}} \in \boldsymbol{V}_h^{\mathbf{a}}$ and $r_{\mathbf{a}} \in Q_h^{\mathbf{a}}$ by solving

$$(\boldsymbol{\varsigma}_{\mathbf{a}}, \boldsymbol{v}_h)_{\omega_{\mathbf{a}}} - (r_h, \nabla \cdot \boldsymbol{v}_h)_{\omega_{\mathbf{a}}} = -(\psi_{\mathbf{a}} \nabla u_h, \boldsymbol{v}_h)_{\omega_{\mathbf{a}}}, \tag{4.6a}$$

$$(\nabla \cdot \boldsymbol{\varsigma}_{\mathbf{a}}, q_h)_{\omega_{\mathbf{a}}} = (f\psi_{\mathbf{a}} - \nabla u_h \cdot \nabla \psi_{\mathbf{a}}, q_h)_{\omega_{\mathbf{a}}}, \tag{4.6b}$$

for all $(\boldsymbol{v}_h, q_h) \in \boldsymbol{V}_h^{\mathbf{a}} \times Q_h^{\mathbf{a}}$. The spaces are defined as

$$\begin{aligned} \boldsymbol{V}_h^{\mathbf{a}} &:= \{\boldsymbol{v}_h \in \boldsymbol{V}_h(\omega_{\mathbf{a}}) : \boldsymbol{v}_h \cdot \boldsymbol{n}_{\omega_{\mathbf{a}}} = 0 \text{ on } \partial\omega_{\mathbf{a}}\}, \\ Q_h^{\mathbf{a}} &:= \{q_h \in Q_h(\omega_{\mathbf{a}}) : (q_h, 1)_{\omega_{\mathbf{a}}} = 0\}, \end{aligned} \qquad \mathbf{a} \in \mathcal{V}_h^{\text{int}}, \tag{4.7a}$$

$$\begin{aligned} \boldsymbol{V}_h^{\mathbf{a}} &:= \{\boldsymbol{v}_h \in \boldsymbol{V}_h : \boldsymbol{v}_h \cdot \boldsymbol{n}_{\omega_{\mathbf{a}}} = 0 \text{ on } \partial\omega_{\mathbf{a}} \setminus \partial\Omega\}, \\ Q_h^{\mathbf{a}} &:= Q_h(\omega_{\mathbf{a}}), \end{aligned} \qquad \mathbf{a} \in \mathcal{V}_h^{\text{ext}}. \tag{4.7b}$$

The global flux is then defined by

$$\boldsymbol{\varsigma}_h := \sum_{\mathbf{a} \in \mathcal{V}_h} \boldsymbol{\varsigma}_{\mathbf{a}}. \tag{4.8}$$

4.2 Linear algebra representation

Once a basis $\{\boldsymbol{v}_i^{\mathbf{a}}\}_{i=1}^{n_{\mathbf{a}}}$ of $\boldsymbol{V}_h^{\mathbf{a}}$ and $\{q_i^{\mathbf{a}}\}_{i=1}^{m_{\mathbf{a}}}$ of $Q_h^{\mathbf{a}}$ are chosen, We can also write the patch system (4.6) in matrix form,

$$A_{\mathbf{a}} \mathcal{X}_{\mathbf{a}} = \mathcal{F}_{\mathbf{a}} \tag{4.9}$$

with the block decomposition

$$A_{\mathbf{a}} = \begin{pmatrix} M_{\mathbf{a}} & -B_{\mathbf{a}}^T \\ B_{\mathbf{a}} & 0 \end{pmatrix}, \; \mathcal{X}_{\mathbf{a}} = \begin{pmatrix} \boldsymbol{x}_{\mathbf{a}} \\ x_{\mathbf{a}} \end{pmatrix}, \; \mathcal{F}_{\mathbf{a}} = \begin{pmatrix} \boldsymbol{F}_{\mathbf{a}} \\ F_{\mathbf{a}} \end{pmatrix}. \tag{4.10}$$

The block matrices $M_{\mathbf{a}} \in \mathbb{R}^{n_{\mathbf{a}} \times n_{\mathbf{a}}}$ and $B_{\mathbf{a}} \in \mathbb{R}^{m_{\mathbf{a}} \times n_{\mathbf{a}}}$ are given by

$$[M_{\mathbf{a}}]_{ij} := (\boldsymbol{v}_j^{\mathbf{a}}, \boldsymbol{v}_i^{\mathbf{a}})_{\omega_{\mathbf{a}}} \tag{4.11a}$$

$$[B_{\mathbf{a}}]_{ij} := (\nabla \cdot \boldsymbol{v}_j^{\mathbf{a}}, q_i^{\mathbf{a}})_{\omega_{\mathbf{a}}}, \tag{4.11b}$$

while the right hand side vectors $\boldsymbol{F}_{\mathbf{a}} \in \mathbb{R}^{n_{\mathbf{a}}}$ and $F_{\mathbf{a}} \in \mathbb{R}^{m_{\mathbf{a}}}$ are given by

$$[\boldsymbol{F}_{\mathbf{a}}]_j := -(\psi_{\mathbf{a}} \nabla u_h, \boldsymbol{v}_j)_{\omega_{\mathbf{a}}}, \tag{4.12a}$$

$$[F_{\mathbf{a}}]_j := (f\psi_{\mathbf{a}} - \nabla u_h \cdot \nabla \psi_{\mathbf{a}}, q_j)_{\omega_{\mathbf{a}}}. \tag{4.12b}$$

The degrees of freedom for the local equilibrated flux $\boldsymbol{\varsigma}_{\mathbf{a}}$ are contained in the vector $\boldsymbol{x}_{\mathbf{a}}$.

4.3 Naive implementation with dynamic allocations

We first consider in Algorithm 5 a possible implementation of the equilibrated flux assembly Definition 4.1.1 at a very high level. In this implementation, we assume that we have access to a finite element library that can create the required spaces $\boldsymbol{V}_h^{\mathbf{a}}$ and $Q_h^{\mathbf{a}}$ as well as evaluate the bilinear forms in (4.11) and (4.12). While this algorithm is very simple to write down, there are some subtleties to consider. Firstly, the geometrical and topological patch information needs to be extracted from the mesh. Many libraries do not support this, as it is not part of the standard assembly procedure. Next, on Line 7, the local to global degree of freedom (DOF) mapping between the patch and the global DOF index space needs to be accessible. Finally, the question of boundary conditions and imposing the mean free condition are quite technical, and we postpone them for the moment.

Even if all the aforementioned subtleties can be addressed, there is still a major drawback of the implementation in Algorithm 5. Namely, this algorithm inherently requires many dynamic memory allocations inside the loop on patches. Firstly, at each iteration of the loop on line 3, the finite element spaces are rebuilt completely from scratch, including in particular all the information pertaining to the reference element (which is often the most heavyweight in terms of allocation). In addition, on line 4 the matrix and vectors associated to the patch system are not reused between iterations, invoking more dynamic allocations.

Dynamic memory allocation can be a significant performance bottleneck due to overhead and data locality. Firstly, the process of allocating and deallocating memory dynamically involves system calls, which are substantially slower compared to accessing stack or pre-allocated heap memory. This added computational overhead is particularly problematic in scenarios involving frequent allocations/deallocations, such as in tight loops or high-frequency function calls. Secondly, modern processors use a hierarchy of caches to speed up access to frequently used data. Good data locality means that the data a program needs is either already in the cache or close together in memory, allowing for efficient cache usage. Dynamic memory allocation can lead to poor data locality because it often allocates memory in a non-contiguous manner. Objects that are logically related in a program might end up being physically scattered in memory. This scattering can result in more cache misses (where the required data is not in the cache), leading to slower performance as the processor has to fetch data from the slower main memory. These considerations are all the more critical in a shared memory parallelism paradigm (which is the one we consider here).

In shared memory parallelism, the extensive use of dynamic memory allocation can lead to bus saturation, a critical performance bottleneck. This issue arises from the heightened memory traffic caused by multiple cores simultaneously accessing and managing memory, on top of the overhead of allocation and deallocation operations. As each core competes for memory resources, the shared bus, which facilitates communication between the CPU cores and the memory subsystem, becomes increasingly congested. This congestion not only

slows down memory operations due to exceeded bandwidth limits but also strains the cache coherency mechanisms. Maintaining cache coherency, essential in multicore systems for consistent memory views across cores, demands additional memory access and bandwidth, further contributing to the saturation of the buses and impacting overall system performance.

These considerations are independant of the programming language, and apply just as much to a statically compiled language with manual memory management like C as to Julia which uses just-in-time compilation and is garbage collected. Thus, our approach to reduce dynamic memory allocations are applicable and relevant to any choice of language for someone interested in serial performance and even moreso for parallel performance.

We therefore reduce dynamic memory allocations in the following ways

1. Assemble cellwise versions of (4.11) and (4.12)
2. Introduce lightweight mappings from the cell DOF index space to the patch DOF index space
3. Reuse patch-local vectors and matrices

In the following section we show how these three steps lead to a much more efficient algorithm, with very few, or even zero memory allocations inside the main loop on patches.

Algorithm 5: Naive equilibrated flux assembly

Input: Mesh $\mathcal{T}_h$, approximate flux $-\nabla u_h$, source term f
Output: Global equilibrated flux ς_h

Initialize $\varsigma_h \in \boldsymbol{V}_h(\Omega)$
for *each node* $\mathbf{a}$ *in* $\mathcal{V}_h$ **do**
  Instantiate the finite element spaces $\boldsymbol{V}_h^{\mathbf{a}}$ and $Q_h^{\mathbf{a}}$ of (4.7)
  Allocate patch matrix and vector $A_{\mathbf{a}}$ and $\mathcal{F}_{\mathbf{a}}$
  Populate $A_{\mathbf{a}}$ and $\mathcal{F}_{\mathbf{a}}$ using $\boldsymbol{V}_h^{\mathbf{a}}, Q_h^{\mathbf{a}}$
  Solve $\mathcal{X}_{\mathbf{a}} = A_{\mathbf{a}}^{-1}\mathcal{F}_{\mathbf{a}}$
  Scatter $\boldsymbol{x}_{\mathbf{a}} = \texttt{DOFs}(\varsigma_{\mathbf{a}})$ to $\texttt{DOFs}(\varsigma_h)$
return ς_h

4.4 High level view of the efficient algorithm

In this section we consider a more involved algorithm that relies less on the builtin functionality provided by a standard finite element library. However, the result is a more efficient algorithm that is also more amenable to shared memory paralellization. As explained in the previous section, one key to the efficient implementation will be to perform an initial cellwise assembly. For an element K and bases $\{\boldsymbol{v}_i^K\}_{i=1}^{n_K}$ of $\boldsymbol{V}_h(K)$ as well as $\{q_i^K\}_{i=1}^{m_K}$ of $Q_h(K)$, we define the cell block matrices $M_K \in \mathbb{R}^{n_K \times n_K}$ and $B_K \in \mathbb{R}^{m_K \times n_K}$ by

$$[M_K]_{ij} := (\boldsymbol{v}_j^K, \boldsymbol{v}_i^K)_K \tag{4.13a}$$
$$[B_K]_{ij} := (\nabla\cdot\boldsymbol{v}_j^K, q_i^K)_K. \tag{4.13b}$$

The right-hand sides of (4.6) are slightly more involved because they depend on the hat functions $\psi_{\mathbf{a}}$, of which there are $d+1$ for a d-dimensional simplex K. Thus, the right-hand

sides are also indexed by the node **a**,

$$[\boldsymbol{F}_K^{\mathbf{a}}]_j := -(\psi_{\mathbf{a}}\big|_K \nabla u_h, \boldsymbol{v}_j^K)_K, \tag{4.14a}$$
$$[F_K^{\mathbf{a}}]_j := (f\psi_{\mathbf{a}}\big|_K - \nabla u_h \cdot \nabla\left(\psi_{\mathbf{a}}\big|_K\right), q_j^K)_K. \tag{4.14b}$$

We now present Algorithm 6, which starts with a loop on cells creating the objects in (4.13) and (4.14). This loop is a common paradigm in finite element codes, and in certain cases this functionality is exposed. This is indeed the case in `Gridap.jl`, and in our implementation this part of the algorithm relies mostly on existing technology in the library. We discuss this in more detail in §4.6.

The next step of the algorithm is the pre-allocation of $A_{\mathbf{a}}$ and $\mathcal{F}_{\mathbf{a}}$, on line 5. This is nontrivial since the dimensions of the patch problems are variable, and we will address this in §4.7. Next, we see that we only perform read operations on the matrices M_K, B_K inside the loop on patches. This effectively reduces the number of allocations inside the loop on patches to zero. We also remark that many details are missing, and in particular we have not discussed the scatters on lines 10, 11, and 14. This requires careful handling of indices. There is also some index juggling required in the imposition of the boundary conditions on line 12. We will give more details about managing indices in §4.8. We also need to remove allocations due to the resolution of the linear system on line 13, which we discuss in §4.7. A prerequisite for all the aforementioned steps is some topological information about the patches, which we discuss in §4.5.

Algorithm 6: Efficient equilibrated flux assembly

Input: Mesh $\mathcal{T}_h$, approximate flux $-\nabla u_h$, source term f
Output: Global equilibrated flux $\boldsymbol{\varsigma}_h$

`// Loop on cells`
1 **for** *each cell* K *in* $\mathcal{T}_h$ **do**
2 Build the cellwise matrices M_K, B_K of (4.13)
3 **for** *each node* **a** *in the cell* K **do**
4 Build the cellwise vectors $\boldsymbol{F}_K^{\mathbf{a}}$, $F_K^{\mathbf{a}}$ of (4.14)
5 Allocate $A_{\mathbf{a}}, \mathcal{F}_{\mathbf{a}}$
6 Initialize $\boldsymbol{\varsigma}_h \in \boldsymbol{V}_h(\Omega)$
`// Loop on patches`
7 **for** *each node* **a** *in the mesh* **do**
8 Zero out $A_{\mathbf{a}}$ and $\mathcal{F}_{\mathbf{a}}$
9 **for** *each cell* $K \in \mathcal{T}_{\mathbf{a}}$ **do**
10 Scatter M_K, B_K to $A_{\mathbf{a}}$
11 Scatter $F_K^{\mathbf{a}}, \boldsymbol{F}_K^{\mathbf{a}}$ to $\mathcal{F}_{\mathbf{a}}$
12 Impose boundary conditions on $A_{\mathbf{a}}, \mathcal{F}_{\mathbf{a}}$
13 Solve $\mathcal{X}_{\mathbf{a}} = A_{\mathbf{a}}^{-1}\mathcal{F}_{\mathbf{a}}$
14 Scatter $\boldsymbol{x}_{\mathbf{a}} = \texttt{DOFs}(\boldsymbol{\varsigma}_{\mathbf{a}})$ to $\texttt{DOFs}(\boldsymbol{\varsigma}_h)$
15 **return** $\boldsymbol{\varsigma}_h$

4.5 Topological patch information

We construct a small type hierarchy that contains the necessary topological information pertaining to patches.

```
abstract type Patch{T} end

struct PatchData{T}
  node_to_offsets::Vector{T}
  patch_cell_ids::Vector{T}
  bdry_edge_ids::Vector{T}
  all_edge_ids::Vector{T}
end

struct DirichletPatch{T} <: Patch{T}
  data::PatchData{T}
end

struct InteriorPatch{T} <: Patch{T}
  data::PatchData{T}
end
```

The two types of patches `DirichletPatch` and `InteriorPatch` corresond to the differences in the definitions of the spaces (4.7). The `PatchData` struct contains topological information related to the patch which is extracted through the `Gridap.Geometry.get_faces` function. For example, `node_to_cell = Geometry.get_faces(topo, 0, 2)` gives a vector of vectors for each node in the mesh to the ids of the cells it belongs to.

4.6 Cellwise assembly

As established in §4.4, we want to compute cellwise matrices and vectors of (4.13) and (4.14). To achieve this, we first build the spaces $\boldsymbol{V}_h(K)$ and $Q_h(K)$ on the reference element and then (lazily) extend it to the whole mesh $\mathcal{T}_h$ via the following code

```
# Assume we already have a desired polynomial order k
# and Triangulation 𝒯ₕ
reffeRT = ReferenceFE(raviart_thomas, Float64, k)
reffeP = ReferenceFE(lagrangian, Float64, k)
# Raviart-Thomas space for the flux
RT_space = FESpace(𝒯ₕ, reffeRT, conformity = :HDiv)
# Broken L^2 space for the Lagrange multiplier
L2_space = FESpace(𝒯ₕ, reffeP, conformity = :L2)
```

With these spaces in hand, we can evaluate bilinear forms (lazily) cell-wise on the mesh. The ingredients for this will be the basis of these respective spaces, as well as a `CellQuadrature` to perform cellwise integration at quadrature points. We create these objects in the next snippet.

```
Qₕ = CellQuadrature(𝒯ₕ, quad_order)
dvp = get_trial_fe_basis(L2_space)
duRT = get_fe_basis(RT_space)
dvRT = get_trial_fe_basis(RT_space)
```

The interface for evaluating the bilinear forms cellwise is then very straightforward,

```
cell_mass_mats = ∫(duRT ⋅ dvRT) * Qₕ
cell_mixed_mats = ∫((∇ ⋅ duRT) * dvp) * Qₕ
```

These two objects are then the mass matrices $\{M_K\}_{K\in\mathcal{T}_h}$, and the second contains the mixed form matrices $\{B_K\}_{K\in\mathcal{T}_h}$ as in of (4.13).

The right-hand sides are more involved due to the fact that in two dimensions the hat functions are supported on three triangles. First, we consider the two following helper functions to get the hat basis.

```
function _get_hat_function_cellfield(i, basis_data, 𝒯ₕ)
  cell_to_ith_node(c) = c[i]
  A = lazy_map(cell_to_ith_node, basis_data)
  return GenericCellField(A, 𝒯ₕ, ReferenceDomain())
end

function _get_hat_functions_on_cells(𝒯ₕ)
  # Always order 1
  reffe = ReferenceFE(lagrangian, Float64, 1)
  V0 = TestFESpace(𝒯ₕ, reffe; conformity = :H1, dirichlet_tags = "boundary")
  fe_basis = get_fe_basis(V0)
  bd = Gridap.CellData.get_data(fe_basis)
  return bd
end
```

Once these two functions are in place to extract the hat functions, we can assemble the right-hand sides: one for each hat function.

```
 hat_fns_on_cells = _get_hat_functions_on_cells(𝒯ₕ)
  RHS_RT_form(ψ) = ∫(-ψ * (∇uₕ ⋅ dvRT))*Qₕ
  RHS_L²_form(ψ) = ∫((f * ψ - ∇uₕ ⋅ ∇(ψ))*dvp)*Qₕ
  cur_num_cells = num_cells(𝒯ₕ)
  # Hardcoded for 2D
  nodes_per_cell = 3
  for i = 1:nodes_per_cell
    ψᵢ = _get_hat_function_cellfield(i, hat_fns_on_cells, 𝒯ₕ)
    cell_RHS_RTs[i] = RHS_RT_form(ψᵢ)
    cell_RHS_L²s[i] = RHS_L²_form(ψᵢ)
  end
```

These arrays correspond to $\{\boldsymbol{F}_K^{\mathbf{a}}\}_{K\in\mathcal{T}_h}$ and $\{F_K^{\mathbf{a}}\}_{K\in\mathcal{T}_h}$ of (4.14). We add a Lagrange multiplier row to handle the mean free condition in the case of interior patches and collect all these objects in a `NamedTuple`

```
cell_Λ_vecs = ∫(1 * dvp) * Qₕ
co = (; cell_mass_mats, cell_Λ_vecs, cell_mixed_mats, cell_RHS_RTs, cell_RHS_L²s)
```

4.7 Patch-level linear algebra

In this section, we detail our strategy for reusing the matrices and vectors of the patch system between iterations. The dimensions of the spaces on the reference element are known and calculated via the following:

```
function get_dofs_per_cell(k, d)
  RT_dofs_per_cell = (k + d + 1) * binomial(k + d - 1, k)
  L²_dofs_per_cell = binomial(k + d, k)
  (RT_dofs_per_cell, L²_dofs_per_cell)
end
```

Next, we need to know the maximum number of cells in a patch. We obtain this information during the initial creation of the `Patch` objects of §4.5, which we call `max_patch_cells`.

```
    (RT_dofs_per_cell, L²_dofs_per_cell) = get_dofs_per_cell(k, d)
    M = zeros(RT_dofs_per_cell * max_patch_cells, RT_dofs_per_cell * max_patch_cells)
    B = zeros(RT_dofs_per_cell * max_patch_cells, L²_dofs_per_cell * max_patch_cells)
    A = [M B; transpose(B) zeros(size(B)[2], size(B)[2])]
    # Pre-allocate the pivot vector for the matrix A
    ws = LUWs(A)
    RHS_RT = zeros(max_patch_cells * RT_dofs_per_cell)
    RHS_L² = zeros(max_patch_cells * L²_dofs_per_cell)
    RHS = [RHS_RT; RHS_L²]
    Λ = similar(RHS_L²)
    σ_loc = similar(RHS)
```

As for the cellwise objects, we collect everything into

```
    linalg = (; M, B, A, ws, Λ, RHS_RT, RHS_L², RHS, σ_loc)
```

Finally, we note that we also instantiate a pivot vector `ws`. This is required for the library `FastLapackInterface.jl` which allows efficient (in terms of allocation) calls to the underlying `LAPACK` solver. This is ultimately used in the following (during the loop on patches once these objects are correctly populated)

```
    A_free_dofs = @view linalg.A[1:n_free_dofs, 1:n_free_dofs]
    RHS_free_dofs = @view linalg.RHS[1:n_free_dofs]
    σ_free_dofs = @view linalg.σ_loc[1:n_free_dofs]
    ldiv!(σ_free_dofs, LU(LAPACK.getrf!(linalg.ws, A_free_dofs)...), RHS_free_dofs)
```

where `n_free_dofs` depends on the patch.

4.8 The DOF manager

We now arrive at the most challenging part of the assembly: the treatment of the DOFs. In particular, the cell objects of §4.6 are all cell-locally indexed. The `Gridap.jl` library exposes the so-called local-to-global map that is standard in finite element codes. This map allows the scatter procedure from the cell-locally indexed objects to objects with mesh global indexing. We will be able to reuse this information to create a local-to-global map where the global index space is a patch $\omega_{\mathbf{a}}$ and not the full mesh $\mathcal{T}_h$. To this end, we introduce the following `struct`

```
struct DOFManager{T, M <: Matrix{T}, V <: Vector{T}}
  # All the cell dofs stored as a matrix
  all_cell_dofs_gl::M
  # The current patch dofs in the global ennumeration
  patch_dofs_gl::V
  # The current free dofs in patch local ennumeration for slicing
  # into the patch local objects
  free_patch_dofs_loc::V
  # The current cell's dofs in the patch local ennumeration
  cell_dofs_loc::V
end
```

This struct is built on the cell-local to mesh-global map to construct cell-local to patch-global maps for each patch. The field `all_cell_dofs_gl` is created via the function `Gridap.FESpaces.get_cell_dof_ids` which takes a `FESpace`, e.g., `RT_space` or `L2_space`, and returns an `ncells` × `ndofs` lazy vector of vector representing the cell to mesh-global map. Here, `ndofs` represents the number of dofs on a cell, e.g., $n_K = \dim(\boldsymbol{V}_h(K))$ or $m_K = \dim(Q_h(K))$. The other fields of the `DOFManager` are updated dynamically, but with

no memory allocations using the `empty!` and `push!` methods (the convention in Julia is to append non-allocating function names with an exclamation point and modify the first argument). For example, the following function updates the `cell_dofs_loc` in place:

```
function update_cell_local_dofs!(dm::DOFManager, cellid)
  empty!(dm.cell_dofs_loc)
  for id in cur_cell_dofs_gl
    new_id = findfirst(n -> n == id, dm.patch_dofs_gl)
    new_id isa Nothing && error("Cannot update cell local dofs!")
    push!(dm.cell_dofs_loc, new_id)
  end
end
```

4.9 The loop on patches

We now consider the heart of the code, i.e., the loop on patches. The efforts up to this point have been so that this part of the algorithm can be completely free of memory allocations so that the shared memory parallelism is efficient. We only show a serial version of this loop for simplicity.

```
    for patch in patches
      # Change the local numbering for the current patch
      update_patch_dofs!(dm_RT, patch.data)
      update_patch_dofs!(dm_L², patch.data)
      # Scatter the cell based matrices in cell_objects to linalg
      matrix_scatter!(linalg.M, co.cell_mass_mats, dm_RT, dm_RT, patch.data)
      matrix_scatter!(linalg.B, co.cell_mixed_mats, dm_L², dm_RT, patch.data)
      # Idem for vectors
      vector_scatter!(linalg.RHS_RT, co.cell_RHS_RTs, dm_RT, patch.data)
      vector_scatter!(linalg.RHS_L², co.cell_RHS_L²s, dm_L², patch.data)
      single_vector_scatter!(linalg.Λ, co.cell_Λ_vecs, dm_L², patch.data)
      # Now that scatter to local system is complete, remove fixed dofs
      remove_homogeneous_neumann_dofs!(dm_RT, patch.data, RT_order)
      # Count the free dofs for this patch once the BCs are imposed
      n_free_dofs_RT = count_free_dofs(dm_RT)
      n_free_dofs_L² = count_free_dofs(dm_L²)
      n_free_dofs = n_free_dofs_RT + n_free_dofs_L²
      # Use the sub-matrices and vectors generated from the scatters to build
      # the monolithic objects
      setup_patch_system!(linalg.A, linalg.RHS, linalg, dm_RT, dm_L²)
      # Handle the pure Neumann case
      if patch isa InteriorPatch
        add_lagrange!(linalg.A, dm_RT, dm_L², linalg.Λ)
        n_free_dofs += 1
      end
      solve_patch!(linalg, n_free_dofs)
      # Scatter to the global FE object's free_values
      scatter_to_global_σ!(σ_gl_chunk, dm_RT, linalg.σ_loc, n_free_dofs_RT)
    end
  end
```

First, we assume we have created `DOFManagers` `dm_RT` and `dm_L2` out of their respective spaces. The various scatter functions are non-allocating. For example, `vector_scatter` is given by

```
function vector_scatter!(patch_vec, cell_vecs, dm, patch_data)
  fill!(patch_vec, 0)
  node_to_offsets = patch_data.node_to_offsets
  for (i, cellid) in enumerate(patch_data.patch_cell_ids)
    cell_vec_all_nodes = @view cell_vecs[cellid, :]
    offset = node_to_offsets[i]
    update_cell_local_dofs!(dm, cellid)
```

```
      cell_vec = cell_vec_all_nodes[offset]
      for i in axes(cell_vec, 1)
        patch_vec[dm.cell_dofs_loc[i]] += cell_vec[i]
      end
    end
  end
```

4.10 Performance study for the estimator

In this section we discuss the cost associated to the evaluation of the estimators in (1.91). We constrast this with the cost of solving the linearization step (1.30). In particular, we consider the cost of assembling the flux reconstruction given in (1.71) since this is by far the most expensive part of calculating the estimators. The patchwise problems are mutually independent and therefore can be solved in parallel. We use the threading model in Julia to possibly take advantage of this parallelism in the results presented below. We also use routines from the `Gridap.jl` library [7, 118] in the assembly with all the details described in the previous sections. The tests in this section were carried out on a cluster with 20 dual socket Cascade Lake Intel Xeon 5218 nodes and 192GB of 2667 MHz RAM. We do not discuss any gains due to significantly reducing the number of linearization iterations due to our adaptive stopping criteria.

For the setting of §1.10.1, we plot the time to assemble the flux via (1.71) versus the time required to solve the linearization step (1.30) by a direct LU solver in Figure 4.2 for polynomial degrees $p = 1, 2, 3, 4$. We use differently-sized meshes for the various polynomial degrees to have a roughly constant number of DOFs. We first remark that the assembly time for (1.71) is linear in the number of DOFs for sufficiently many DOFs. This is in contrast to the time to solve the linearization step (1.30), which is superlinear as we use a direct LU solver. Next, we observe that the total time to assemble the flux decreases with more processors for sufficiently many DOFs, and that this is more pronounced for $p \geq 2$.

We now consider Table 4.1, where we tabulate the percentage of the total runtime, i.e.,

$$P_{\mathrm{f}} := \frac{T_{\mathrm{f}}}{T_{\mathrm{f}} + T_{\mathrm{N}}}, \tag{4.15}$$

where T_{f} is the time for assembling the flux and T_{N} is the time for solving the linearization step. We calculate this quantity for the largest number of DOFs that we consider, approximately 1.25e6 DOFs for the various polynomial degrees. We see that this value is less than 0.5 (meaning the flux assembly is cheaper than solving the linearization step) for $p \geq 2$ and 8 or more processors. The table also reflects a monotone decrease of P_{f} as we increase the number of processors, which is more pronounced for $p \geq 2$. We conclude that the cost of the estimator can be effectively reduced by parallelization (adding computational resources), at least for problems of a sufficient size.

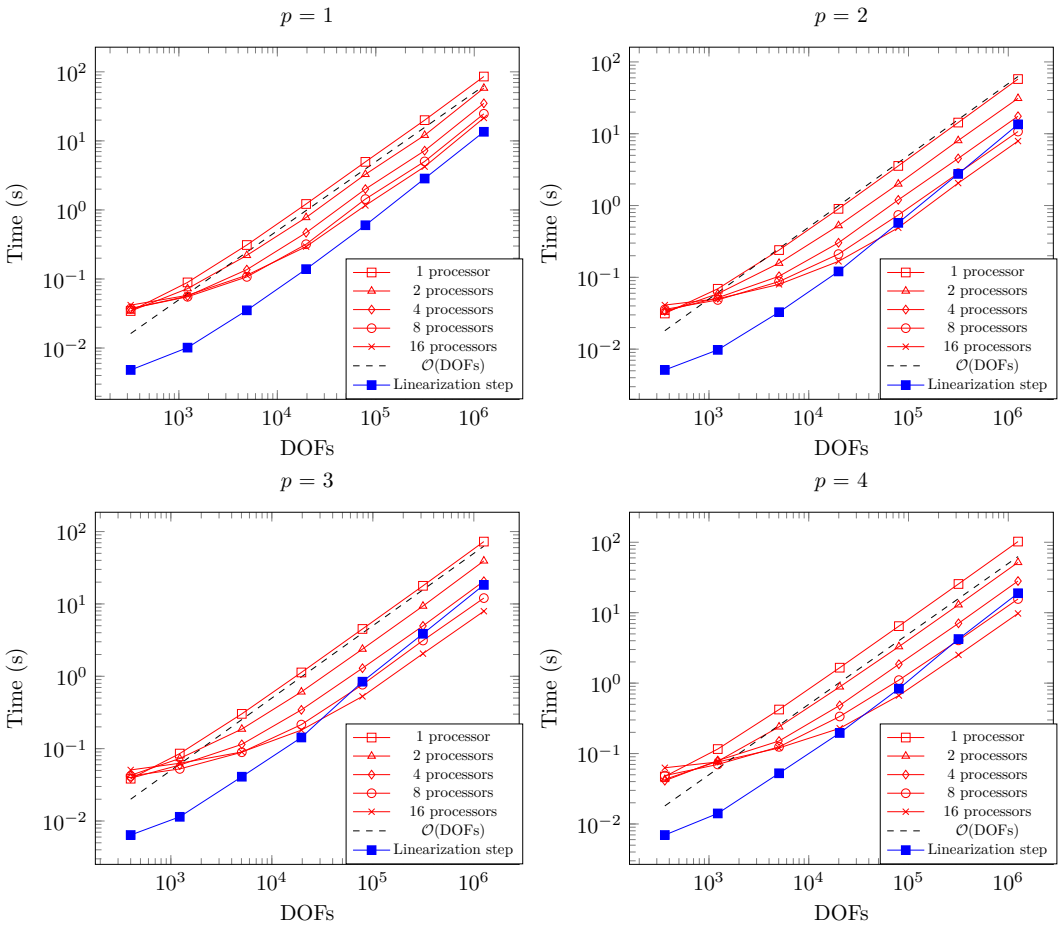


Figure 4.2: Comparison between solving linearization step (1.30) and assembling the equilibrated flux via (1.71) for different polynomial degrees p and different numbers of processors.

Table 4.1: Percentage of total runtime (4.15) for 1.25e6 DOFs.

p	Processors				
	1	2	4	8	16
1	87%	81%	74%	66%	62%
2	81%	70%	58%	45%	39%
3	81%	68%	54%	41%	28%
4	85%	74%	61%	45%	33%

Chapter 5

An adaptive iterative refinement multigrid method

5.1 Introduction

Double precision (FP64) floating point numbers have long been the gold standard for scientific computations. However, recently there has been a great deal of interest in mixed precision algorithms, i.e. incorporating single precision (FP32) and lower (FP16 and even FP8). Mixed precision algorithms judiciously blend different numerical precision levels within a computation, allowing to retain the original accuracy while still performing many operations in the lower precision. By using lower precision, the memory footprint of the algorithm can be reduced substantially.

A big driver for the interest in mixed precision algorithms is the reduced memory footprint (half the bytes in the case of single vs. double precision for example) that comes with lower precision formats. This is especially important for *memory-bound* kernels for which the bottleneck is the time required to move data through the memory hierarchy rather to perform actual floating point operations (FLOPs) on the data. Many kernels in scientific computing including sparse matrix-vector multiplication are memory-bound [128, 125]. This is not suprising considering on modern computer architectures, the disparity between the fast processing speeds of modern CPUs and the slower data retrieval times from memory. CPUs, often operating at speeds measured in GHz, can perform billions of cycles per second, with each cycle taking just a fraction of a nanosecond. In contrast, accessing data from main memory (DRAM) typically takes tens to hundreds of nanoseconds.

While mixed precision can have great benefits in terms of savings for memory bound kernels, the inherent challenge of using lower precision is its reduced robustness with respect to ill-conditioning. More precisely, lower precision formats, with fewer bits for number representation, result in greater rounding errors (especially in operations like subtraction), and a heightened sensitivity to perturbations, see e.g. [71] and references therein. These limitations make lower precision unsuitable for accurately resolving ill-conditioned problems, as the accumulated errors from reduced precision can lead to significant deviations from correct results, undermining the reliability of the computations.

Specifically in the context of solving linear systems, a mixed precision algorithm that seeks to retain robustness is iterative refinement, first introduced in [124]. Iterative refinement is an iterative method consisting of an inner solver and a residual correction. In particular, an equation for the error is solved in lower precision and the residual calculation is computed

in higher precision, see Algorithm 7. The basic tenant of iterative refinement is that solving for the correction generally requires less precision since the entries of the residual are on a smaller order compared to the right hand side b. Since the residual is small, and if A is not too ill-conditioned, the correction y should also be small. Furthermore, if y is small, then the rounding errors in computing y should be negligible when compared to the rounding errors in x. Iterative refinement has been studied in the context of direct solvers [124, 38], Krylov subspace methods e.g. in [27, 26], and more recently in [61, 88, 115] for multigrid methods.

Algorithm 7: Iterative refinement

Input: initial guess x^0, A, b, `high_precision`, `low_precision`, `tol`
$i := 0$, $r^0 := b - Ax^0$
while $\|r^i\| >$ tol **do**
  $r^i := b - Ax^i$ `* Compute residual in high precision`
  $y^i := \text{InnerSolve}(A, r^i)$ `* Compute correction in low precision`
  $x^{i+1} := x^i + y^i$ `* Update in high precision`
  $i := i + 1$

In the context of iterative refinement algorithms, the issue of stopping criterion is central [5]. However, most standard stopping criterion are based on the algebraic residual vector $b - Ax^i$ and therefore scale with the true error as some function of the matrix condition number. In this work we will use a stopping criteria based on a computable a posteriori estimator of the error evaluated directly in the energy (A) norm.

To compute our error estimator, we choose a particular inner solve as a special case of the geometric multigrid method propsed in [91, 89] and in the PhD thesis [90]. This version of geometric multigrid has several advantages: 1) there are no pre-smoothing steps and only one post-smoothing step; 2) the smoother is a simple Jacobi smoother in the lowest order case and block-Jacobi in general; 3) the solver is p-robust, that is, the contraction property of the solver is independent of the polynomial order p; 4) the solver has a built-in a posteriori estimator. This last property will be crucial for us, as we will design an adaptive algorithm based on this estimator. an a-posteriori estimator that is equivalent to the true error (efficient and reliable) and in particular is constant free in the lower bound [91]. Furthermore, the contraction factor for this method satisfies explicit bounds that make it attractive from the point of view of analysis.

The main results of this chapter are 1) the introduction of an iterative refinement variant where the stopping criterion is based on a rigorous estimator of the algebraic energy error; 2) the validation of a performance model of a matrix-vector product based inner solver; and 3) the design of an adaptive algorithm that is more robust for highly ill-conditioned problems while automatically switching the precision if needed and safe guarding the overall procedure. The rest of the chapter is organized as follows. In §5.2 we present our continous model problem and its discretization as well as recalling some basic elements from geometric multigrid. In §5.3 we develop estimates for the compression of low precision for sparse CSC matrices and present numerical speedup results to estimate best possible gains. In §5.4 we show an example of divergence of iterative refinement and propose an adaptive safeguarded version backed with numerical evidence. Finally, we present our conclusions in §5.5.

5.2 Model problem and discretization

The linear systems we consider arise in the discretization of second-order elliptic diffusion problems posed on an open bounded polytope with Lipshitz-continuous boundary, $\Omega \subset \mathbb{R}^d, d \in \{1,2,3\}$. For a given source function $f \in L^2(\Omega)$ and a (possibly) tensor-valued diffusion coefficient $\mathcal{K} \in [L^\infty(\Omega)]^{d\times d}$, the problem reads as: find $u : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla\cdot(\mathcal{K}\nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{5.1}$$

The corresponding weak form of this problem is to find $u \in H_0^1(\Omega)$ such that

$$(\mathcal{K}\nabla u, \nabla v) = (f, v) \quad \forall v \in H_0^1(\Omega), \tag{5.2}$$

where $(\cdot,\cdot)$ denotes the $L^2(\Omega)$ or $[L^2(\Omega)]^d$ inner product. We then consider a matching simplicial mesh $\mathcal{T}_J$ and consider the finite-dimensional space

$$V_J := \mathcal{P}_1(\mathcal{T}_j) \cap H_0^1(\Omega), \tag{5.3}$$

and in turn $\mathcal{P}_1(\mathcal{T}_J) := \{v_J \in L^2(\Omega) : v_J|_K \in \mathcal{P}_1(K) \quad \forall K \in \mathcal{T}_J\}$. The finite-dimensional problem is then stated as: find $u_J \in V_J$ such that

$$(\mathcal{K}\nabla u_J, \nabla v_J) = (f, v_J) \quad \forall v_J \in V_J. \tag{5.4}$$

Once we have chosen a basis $\{\phi_i^J\}_{1\le i\le \dim(V_J)}$ of V_J, this is equivalent to the following matrix equation:

$$A_J x_J = b_J, \tag{5.5}$$

where $[A_J]_{ij} = (\mathcal{K}\nabla\phi_j^J, \nabla\phi_i^J)$ and $[b_J]_i = (f, \phi_i^J)$. We further assume that there exists a hierarchy of nested simplicial meshes $\{\mathcal{T}_j\}_{1\le j\le J-1}$ such that $\mathcal{T}_j$ is obtained by bisecting the longest edges of elements of $\mathcal{T}_{j-1}$ for all $1 \le j \le J-1$ and such that previously introduced fine mesh $\mathcal{T}_J$ is obtained by bisecting $\mathcal{T}_{J-1}$. This mesh hierarchy allows us to introduce a sequence of prolongation matrices P_j for $2 \le j \le J$ representing the usual piecewise linear interpolation operators standard in geometric multigrid, see e.g., [65, 24]. We also introduce `num_coarse_dofs` as the number of dofs on the coarsest level.

We define the algebraic energy error by

$$e_J^i = \|\mathcal{K}^{1/2}\nabla(u_J - u_J^i)\|, \tag{5.6}$$

where u_J^i is the approximate solution on step i of some iterative linear solver to be defined below. We also introduce the following data that we will use later in numerical experiments.

Example 5.2.1 (Skyscraper)**.** *This example is inspired by [91, Section 9] and [4, Section 5.3]. We take $\Omega := (0,1)^2$, $f = 1$ and $u|_{\partial\Omega} := \sqrt{x}$, and the diffusion coefficient $\mathcal{K} := c(x,y)I$ where*

$$c(x,y) := \begin{cases} 10^7\lfloor 9x+1\rfloor, & \text{if } \operatorname{mod}(\lfloor 9x\rfloor, 2) = 0 \text{ and } \operatorname{mod}(\lfloor 9y\rfloor, 2) = 0; \\ 1 & \text{otherwise,} \end{cases} \tag{5.7}$$

the function c is plotted in Figure 5.1.

We also consider two more examples

Example 5.2.2 (Sine)**.** *This is a smooth example with a known solution. We take $\Omega := (-1,1)^2$, $u(x) := \sin(2\pi x)\sin(2\pi y)$, and the diffusion coefficient $\mathcal{K} := I$.*

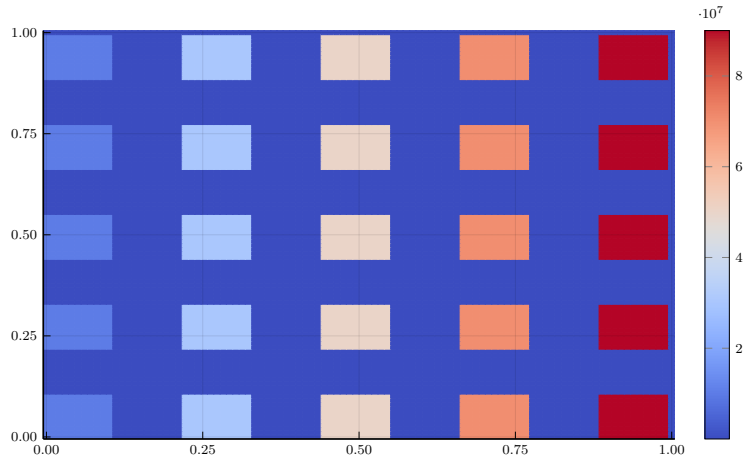


Figure 5.1: The scalar coefficient function $c(x, y)$ defined in (5.7) for the skyscraper example in the unit square.

5.3 Performance and mixed precision

5.3.1 Storage of sparse matrices

In this section we will develop a performance model that will reveal best-possible savings in the context of reduced precision for sparse matrices. In particular, we will concentrate on the storage of compressed column storage (CSC) matrices. For a general sparse matrix $A \in \mathbb{R}^{n\times n}$, the CSC storage format uses three vectors, which we call $\texttt{colptr}(A)$, $\texttt{rowval}(A)$, and $\texttt{nzval}(A)$. The vector $\texttt{colptr}(A)$ is of length $n+1$ and gives information about the distribution of the nonzero elements across the columns of A. More precisely, `colptr` is defined such that the absolute indices (counting all the nonzeros) in column j of A are between $\texttt{colptr}[j]$ and $\texttt{colptr}[j+1]$. The array $\texttt{rowval}(A)$ is defined such that $i = \texttt{rowval}[j^*]$ if and only if $[A]_{ij} = \texttt{nzvals}[j^*]$ for $j^* \in \{\texttt{colptr}[j], \cdots, \texttt{colptr}[j+1]\}$. Therefore, $|\texttt{rowval}(A)| = |\texttt{nzrow}(A)| = \texttt{nnz}(A)$.

To illustrate the storage pattern, we consider an example of the well-known tridiagonal matrix,

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{5\times 5}. \tag{5.8}$$

The CSC representation of this matrix is given in Figure 5.2.

We will assume the index arrays `colptr` and `rowval` are stored using `Int32` (4 bytes). We use the notation A_p to mean that the `nzval` array is stored using either double precision $p = 64$ (8 bytes) or single precision $p = 32$ (4 bytes). We now calculate the compression ratio of storing the matrix in single versus double precision,

$$\begin{aligned} C_{64\to 32} &:= \frac{\texttt{bytes}(A_{64})}{\texttt{btyes}(A_{32})} \\ &= \frac{\texttt{bytes}(\texttt{colptr}(A_{64}) + \texttt{rowval}(A_{64}) + \texttt{nzval}(A_{64}))}{\texttt{bytes}(\texttt{colptr}(A_{32}) + \texttt{rowval}(A_{32}) + \texttt{nzval}(A_{32}))} \end{aligned}$$

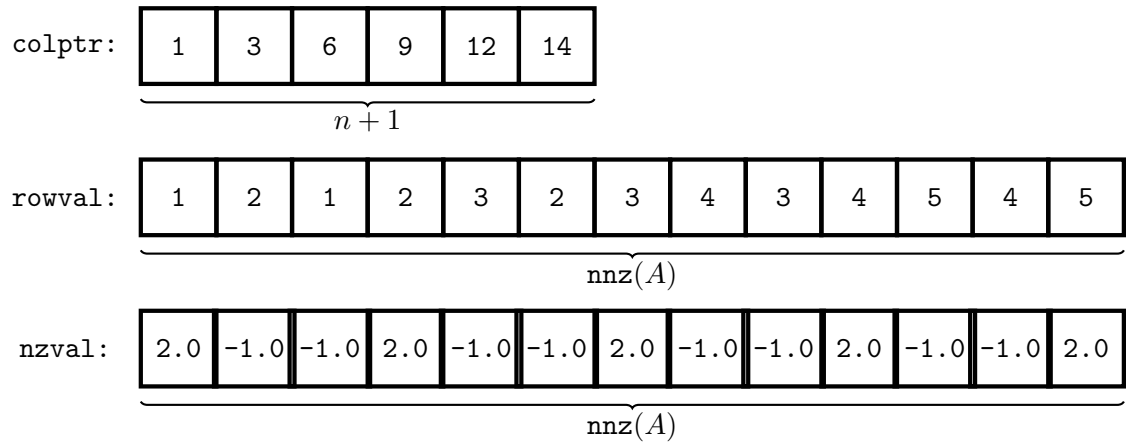


Figure 5.2: Sparse CSC storage of the 5×5 tridiagonal matrix (5.8).

$$= \frac{12\,\mathtt{nnz}(A) + 4(n+1)}{8\,\mathtt{nnz}(A) + 4(n+1)}.$$

We assume that $\mathtt{nnz}(A) \approx 3^d n$, due to the stencil structure of the lowest order continuous Galerkin FEM described in §5.2, and hence

$$C_{64\to32} = \frac{12\,\mathtt{nnz}(A) + 4(n+1)}{8\,\mathtt{nnz}(A) + 4(n+1)} \approx \frac{12(3^d n) + (4n+4)}{8(3^d n) + (4n+4)} \overset{n\to\infty}{\longrightarrow} \frac{12}{8} = 1.5. \tag{5.9}$$

For $d = 2, n = 10000$ this gives $C_{64\to32} = 1.47$ which is close to the 1.5 bound. We remark that this is much lower than the factor of 2 if one considers only the vector `nzval`.

The bulk of our mixed precision algorithm below will be sparse matrix-vector products (spMV). Since we know that matrix vector multiplication is a memory-bound operation [128, 125], we can only hope for better performance in by reducing the overall memory throughput. In the best case then, we could hope for a speedup of $C_{64\to32}$.

Remark 5.3.1 (Choice of `Int32`)**.** *We would like to point that the preceding analysis would result in an even worse compression ratio using* `Int64`*. Since none of our example require indexing exceeding the maximum value of* $2^{31} - 1$ *(*2.14×10^9*), we can use* `Int32` *indices. Another optimization would be to use unsigned integers since we are dealing with indices.*

5.3.2 Iterative refinement and optimal multigrid V-cycle

We now give a more detailed description of our proposed algorithm combining standard iterative refinement with a the lowest order case of the optimal multigrid solver in [91, 89]. First we introduce a modified version of the iterative refinement algorithm 7, given in Algorithm 8. In particular, the main difference here is that the inner solve is now specified as the optimal V-cycle in Algorithm 9 described shortly. This choice of inner solver has the added advantage that it also produces an estimator on each iteration i: namely η^i, which we then use instead of checking the norm of the residual as was the case in Algorithm 7. As proven in [91], the estimator η^i provides a constant-free lower bound to the algebraic energy error (5.6), as well as an upper bound where the constant in the upper bound depends on the contraction factor of the solver. This is advantageous when compared to the standard

residual based estimator based on the algebraic residual vector r^i used in 7, which depends inherently on the condition number of the matrix A.

We now explain in detail the optimal V-cycle of Algorithm 9. In particular, the optimal V-cycle corresponds to a standard geometric multigrid V-cycle [24, 65] with zero pre-smoothing steps and a single Jacobi post-smoothing step on each level (merely one step is sufficient for convergence; in practice we ν number of post-smoothing steps, typically $\nu = 3$). The main difference with respect to a usual V-cycle is that the update of the error correction on line 11 is weighted by the parameter λ_j, which is the optimal step size, ensures an algebraic energy error Pythagorean decreases formula. In particular, this choice of step size also permits the calculation of an error estimator η, initialized on line 3 and updated on line 10. We use this estimator in the stopping criterion of the iterative refinement algorithm in place of the standard condition on the residual.

5.3.3 Performance test

We first note that the main operations in Algorithm 9 are sparse matrix vector multiplications. Thus, we can assume based on the discussion in §5.3.1 that the speedup when using single precision storage of the sparse matrices will not exceed $C_{64\to32}$, again due to the memory-bound nature of sparse matrix vector multiplication.

For our performance test, we use Example 5.2.2. We use a uniform coarse mesh, with varying `num_coarse_dofs` as well as varying the number of levels J to give different amounts of work for the overall V-cycle. We then present the time to complete one iteration of the while loop starting on line 4 of Algorithm 8. First, in Figure 5.3, we plot the time taken if we perform all operations of Algorithm 8 in double precision including the Optimal V-Cycle on the x-axis, and the time for iterative refinement (the algorithm as is it written) and the time for computing all operations in the loop in single precision for the same problem size on the y axis. We see that as the problem gets bigger and hence the time gets longer, the advantage in terms of timing for both single precision and double precision (in blue in Algorithm 8) is clear, and there is actually not too much added overhead for iterative refinement with double precision. Next, we plot the speedup, i.e. the time using all double precision divided by the time using either `high_precision` = single or `high_precision` = double in Figure 5.4. We also plot our theoretical maximum speedup coming from the compression factor discussed in §5.3.1. We see that the results are quite reasonable and in certain cases we obtain a speedup close to theoretical best speedup (5.9) for pure sparse matrix vector multiplication.

Algorithm 8: Estimator based iterative refinement

1 **Input:** initial guess x^0, A, b, `high_precision`, `low_precision`
2 Initialize: $i := 0, \eta^0 := \infty$
3 Carry out all the following steps in specified `precision`
4 **while** $\eta^i >$ `tol` **do**
5 $\quad r_J^i := b_J - A_J x^i$ `* Compute the fine level residual`
6 $\quad (y_{\text{sum}}^{i+1}, \eta^i) :=$ Optimal V-Cycle(r_J^i) `* Compute correction and estimator`
7 $\quad x^{i+1} := x^i + y_{\text{sum}}^{i+1}$ `* Update the current solution`
8 $\quad i := i + 1$

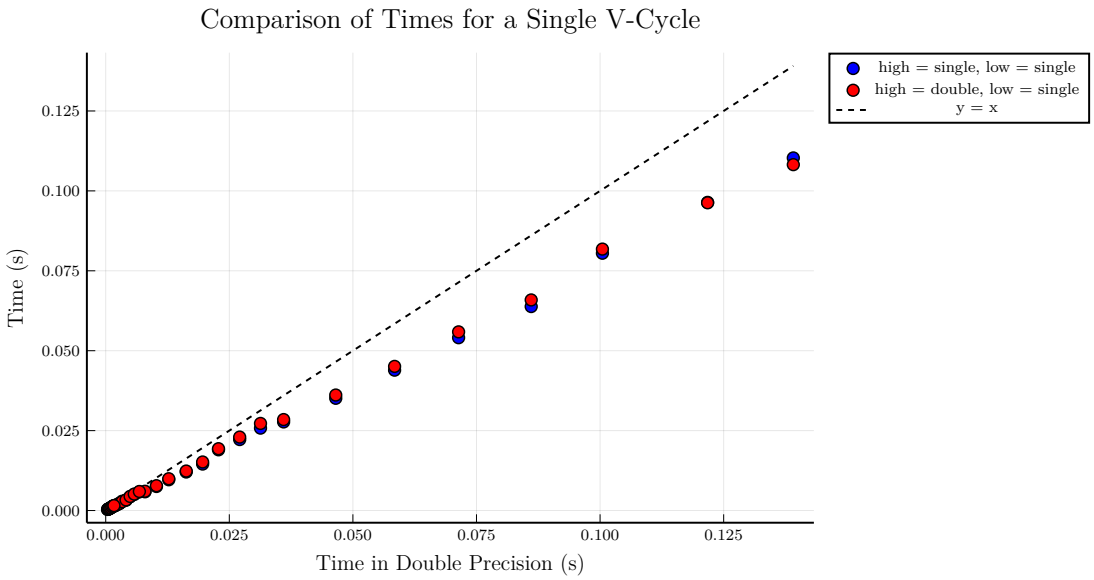


Figure 5.3: [J, `num_coarse_dofs` varying, $\nu = 3$] Comparison wall times of performing one iteration of the iterative refinement loop in double precision on the x-axis with the time using only single precision operations and storage, or using the combination of single and double precison indicated in the Algorithm 8.

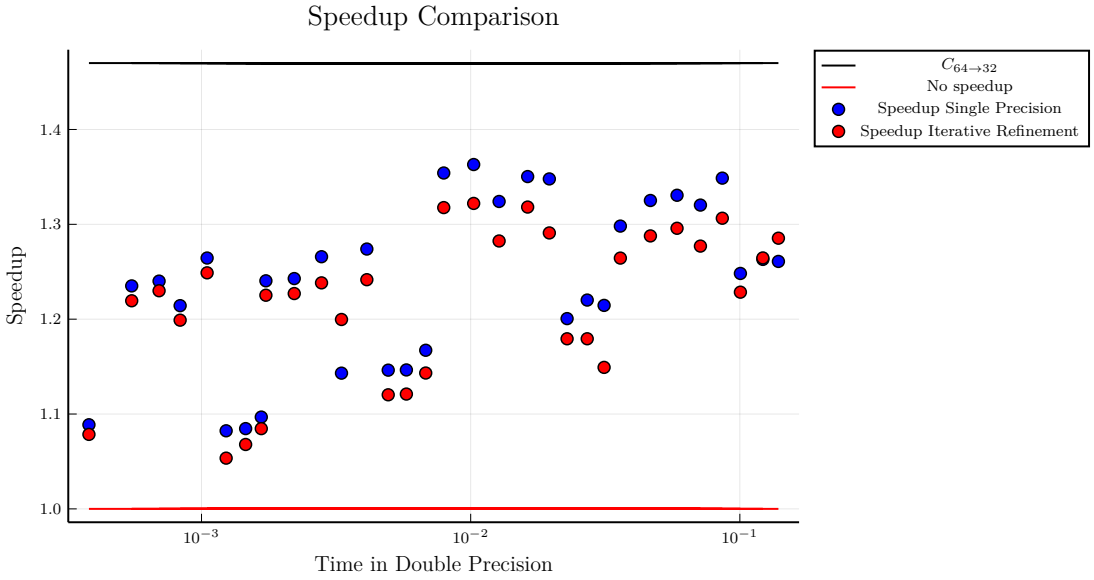


Figure 5.4: [J, `num_coarse_dofs` varying, $\nu = 3$] Showing the speedup corresponding to one iteration i of Algorithm 8 where all the operations are performed in double, single, by the color code in Algorith 8 with `high_precision` = double and `low_precision` = single (standard iterative refinement).

Algorithm 9: Optimal V-Cycle

Input : $\{A_j\}_{j=1}^J$, $\{P_j\}_{j=2}^J$, r_J, tol, precision, ν

$r_k := \prod_{j=J}^k P_j^T r_J$, $1 \le k \le J-1$ ∗ Restrict the fine mesh residual

$y_1 := A_1^{-1} r_1$ ∗ Solve exactly on coarsest mesh

$\eta^2 := (y_1)^T A_1 y_1$ ∗ Initialize estimator

for $j = 2, \dots, J$ **do**

  $y_{\text{sum}} := P_j y_{j-1}$ ∗ Interpolate the correction

  for $k = 1, \dots, \nu$ **do**

    $r_j := r_j - A_j y_{\text{sum}}$ ∗ Update the residual

    $y_j := \text{diag}(A_j)^{-1} r_j$ ∗ $p = 1$ patches equivalent to Jacobi

    $\lambda_j := \frac{y_j^T r_j}{(y_j)^T A_j y_j}$ ∗ Calculate optimal step size

    $\eta^2 := \eta^2 + (\lambda_j)^2 (y_j)^T A_j y_j$ ∗ Update square of estimator

    $y_{\text{sum}} := y_{\text{sum}} + \lambda_j y_j$ ∗ Update the correction

return $(y_{\text{sum}}, \sqrt{\eta^2})$

Algorithm 11: Adaptive safeguarded iterative refinement

Input : $\{A_j\}_{j=1}^J$, $\{P_j\}_{j=2}^J$, x^0 initial guess, b, tol, N_{b}, γ

$r^0 := b - A_J x^0$ ∗ Compute the initial residual in double

$(y_{\text{sum}}^1, \eta^1) :=$ V-Cycle$(r_J^0,$ double$)$ ∗ Solve initial step in double

$x^1 := x^0 + y_{\text{sum}}^1$ ∗ Update the solution in double

Initialize: $i := 2$

while $\eta^i >$ tol **do**

  for $n = 1, \dots, N_{\text{b}}$ **do**

    $r_J^{i+n} := b - A_J x^{i+n}$ ∗ Compute the residual in double

    $(y_{\text{sum}}^{i+n+1}, \eta^{i+n}) :=$ V-Cycle$(r_J^{i+n},$ single$)$ ∗ Solve step in single

    $x^{i+n+1} := x^{i+n} + y_{\text{sum}}^{i+n+1}$ ∗ Update the solution in double

  $r^{i+N_{\text{b}}} := b - A_J x^{i+N_{\text{b}}}$ ∗ Compute the residual in double

  $(y_{\text{sum}}^{i+N_{\text{b}}+2}, \eta^{i+N_{\text{b}}+1}) :=$ V-Cycle$(r_J^i,$ double$)$ ∗ Solve step in double

  $x^{i+N_{\text{b}}+2} := x^{i+N_{\text{b}}+1} + y_{\text{sum}}^{i+N_{\text{b}}+2}$ ∗ Update the solution in double

  $i := i + N_{\text{b}} + 1$ ∗ Update the index by total steps taken

  Update ρ_η^i as in (5.11) ∗ Geometric mean of double estimators

  if $\rho_\eta^i > \gamma$ **then**

    Set $N_{\text{b}} := 0$ ∗ All subsequent iterations in fully double

5.4 Stability considerations and adaptive precision

In this section, we first consider a motivating example where standard iterative refinement fails and we propose an adaptive algorithm that is able to overcome this difficulty. We now define the effectivity index for $i \ge 1$ as

$$I_{\text{eff}}^i = \frac{e_J^i}{\eta^i}. \tag{5.10}$$

Note however that η^i is actually only computed on step $i+1$ so that we do not have I^i_{eff} of the last iteration. To calculate the exact finite dimensional solution u_J, we solve the linear system with a direct solve, which is feasible in the test cases we consider.

5.4.1 Motivation for adaptivity

We first consider applying our new iterative refinement algorithm 8 with the optimal V-Cycle inner solver to Example 5.2.1. Note that Algorithm 8 is not adaptive and the choice of high precision and low precision is made once and for all. The result is given in Figure 5.5, where we see that Algorithm 8 diverges after several iterations, but it able to converge with the choice of `low_precision` = double. We are thus motivated to design an adaptive algorithm using the estimator to determine if iterative refinement is failing and if we need to switch to double precision.

5.4.2 Adaptive safeguarded iterative refinement

Our adaptive algorithm will be based on the two following simple facts:

1. We can trust the estimator when all operations are performed in double precision.
2. Barring precision issues, the optimal V-cycle of Algorithm 9 has been proved to be contractive [91], and thus should be decreasing with a constant contraction factor.

To address the first point, we introduce a parameter, N_{b}. This will correspond to the number of consecutive iterations we will perform using iterative refinement before performing an iteration with only double precision. This allows us to trust the estimator calculated every $N_{\text{b}}+1$-th step.

To address the second point, we will then look at the ratio between the estimator at step i and step $i-(N_{\text{b}}+1)$ to determine whether the solver is still contracting normally. To measure the contraction, we define the following quantity

$$\rho^i_\eta := \left(\frac{\eta^i}{\eta^{i-(N_{\text{b}}+1)}}\right)^{\frac{1}{N_{\text{b}}+1}}. \tag{5.11}$$

The idea of this definition is that if all is going well, between the time we last computed the estimator in double precision ($N_{\text{b}}+1$ iterations ago) we should have contracted by some constant factor, i.e., we should have that

$$\frac{\eta^i}{\eta^{i-(N_{\text{b}}+1)}} \approx C^{N_{\text{b}}+1}$$

for some constant $C<1$. Thus, we compare ρ^i_η with a user specified parameter $\gamma<1$ to check if we are indeed converging at a constant contraction factor. If this is not the case, we break out of the iterative refinement algorithm and perform all calculations with double precision floating point numbers. The adaptive algorithm putting these elements together is given in Algorithm 11. The main elements are the initial step entirely in double precision to compute an initial reliable estimator (lines 1–3), then the loop on $N_{\text{b}}+1$ of standard iterative refinement (lines 6–9), followed by another fully double precision estimator calculation (lines 10–14). Finally, on line 15, we compare ρ^i_η against the user specified γ to see if the solver is still contracting. If it is not, we set N_{b} to 0, since we then bypass the loop starting on on line 6 and continue hereafter entirely in double precision.

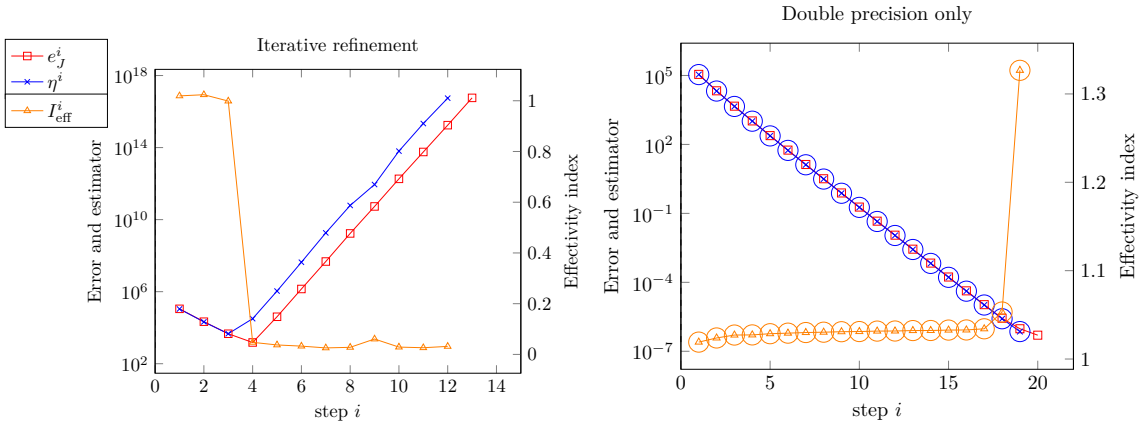


Figure 5.5: $[J = 4, \nu = 3$, `num_coarse_dofs` $= 208$, `tol`$= 10^{-6}]$ The estimator and error of Algorithm 8 applied to Example 5.2.1 with `high_precision`$=$ double and `low_precision`$=$ single, i.e. standard iterative refinement (left) and `high_precision`$=$ double and `low_precision`$=$ double (right), i.e. all operations in double precision. The circled iterations are performed entirely in double precision.

5.4.3 Test of the adaptive safeguarded algorithm

We first consider applying our adaptive safeguarded Algorithm 11 to the troublesome example 5.2.1. The result is given in Figure 5.6, where we consider $N_{\text{b}} := 4$ and $\gamma := 0.6$. The algorithm catches the divergence at the first comparision occuring on the 6th iteration. Furthermore, the progress of the iterative refinement (before starting to diverge) is not entirely lost. Since we catch the divergence early, we can use the current iterate as an initial guess for the double precision only iterative refinement, i.e., the optimal V-cycle multigrid of Algorithm 9. Overall, we end up using 4 extra iterations than if we performed all iterations in double from the beginning.

Next, we consider the result of applying this algorithm to a well-conditioned problem, namely Example 5.2.2. The result is given in Figure 5.7. We see that in this case, since there is no divergence, the adaptive algorithm does not make the switch to double precision, and the majority of the iterations are still computed using iterative refinement, more precisely $\frac{N_{\text{b}}}{N_{\text{b}}+1} = 0.8$, discounting the first iteration all in double. Furthermore, the same exact number of iterations are required to achieve the user-specified tolerance as for Algorithm 8 without adaptivity/safeguarding.

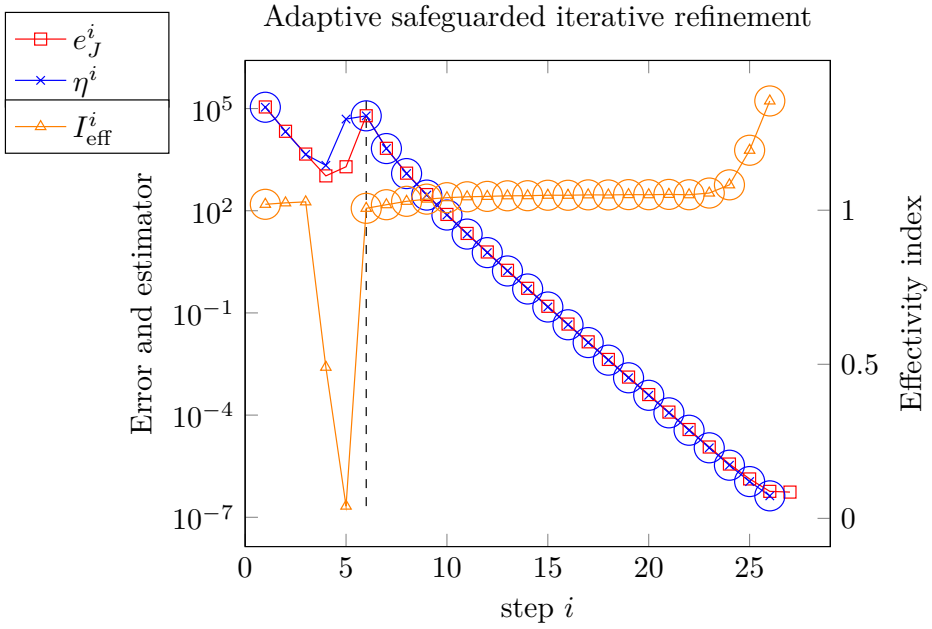


Figure 5.6: [$J = 4$, $\nu = 3$, `num_coarse_dofs` $= 208$, $N_b = 4, \gamma = 0.6$, `tol` $= 10^{-6}$] Applying the adaptive safeguarded iterative refinement of Algorithm 11 to Example 5.2.1. The circled iterations are performed entirely in double precision.

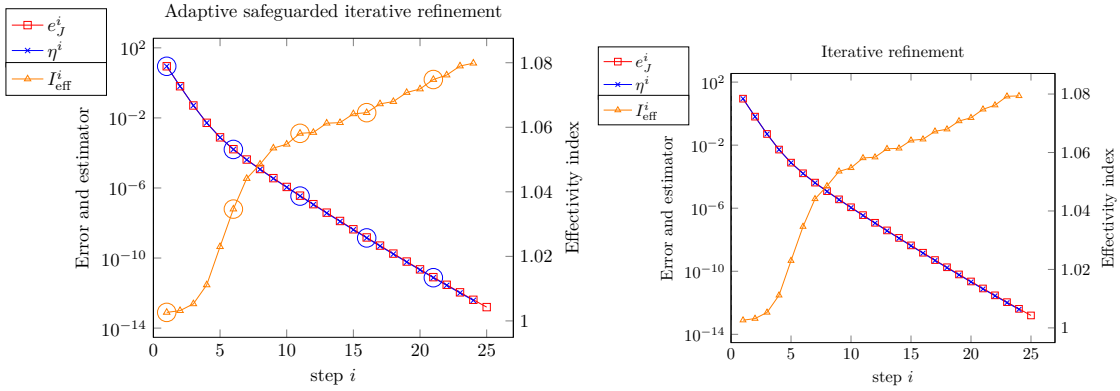


Figure 5.7: [$J = 4, \nu = 3$, `num_coarse_dofs` $= 226$, $N_b = 4$, $\gamma = 0.6$, `tol` $= 10^{-12}$] Comparision of the estimator-based iterative refinement 8 with the adaptive safegaurd variant of Algorithm 11 applied to the well-conditioned example 5.2.2. The circled iterations are performed entirely in double precision.

5.5 Conclusion

In this chapter, we have presented an adapted version of the classical iterative refinement algorithm where the stopping criteria comes from information in the inner solve and are based on an estimator of the algebraic error in the energy norm in place of the L^2 norm of the algebraic vector. We then study the compression effect of storing sparse matrices in single precision and estimate the maximal possible speedup for matrix vector multiplicaiton. Finally, we present an example of an ill-conditioned problem where classical iterative refine-

ment diverges and propose an adaptive algorithm that provides a safeguard switch to double precision in this case. The adaptive algorithm is also shown to improve the performance in the case of a well-conditioned problem for which the switch is not achieved.

Bibliography

[1] M. Ainsworth and J. T. Oden. *A posteriori error estimation in finite element analysis*. Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], New York, 2000.

[2] H. W. Alt and S. Luckhaus. Quasilinear elliptic-parabolic differential equations. *Math. Z.*, 183(3):311–341, 1983.

[3] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41, 2001.

[4] A. Anciaux-Sedrakian, L. Grigori, Z. Jorti, J. Papež, and S. Yousef. Adaptive solution of linear systems of equations based on a posteriori error estimators. *Numer. Algorithms*, 84(1):331–364, 2020.

[5] M. Arioli, J. W. Demmel, and I. S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.*, 10(2):165–190, 1989.

[6] I. Babuška and T. Strouboulis. *The finite element method and its reliability*. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York, 2001.

[7] S. Badia and F. Verdugo. Gridap: an extensible finite element toolbox in Julia. *Journal of Open Source Software*, 5(52):2520, 2020.

[8] V. Barbu and T. Precupanu. *Convexity and Optimization in Banach Spaces*. Springer Netherlands, 2014.

[9] S. Bartels. Error control and adaptivity for a variational model problem defined on functions of bounded variation. *Math. Comp.*, 84(293):1217–1240, 2015.

[10] S. Bartels. *Numerical methods for nonlinear partial differential equations*, volume 47 of *Springer Series in Computational Mathematics*. Springer, Cham, 2015.

[11] S. Bartels and A. Kaltenbach. Explicit and efficient error estimation for convex minimization problems, 2022. ArXiv preprint 2204.10745.

[12] S. Bartels and M. Milicevic. Efficient iterative solution of finite element discretized nonsmooth minimization problems. *Comput. Math. Appl.*, 80(5):588–603, 2020.

[13] S. Bartels and M. Milicevic. Primal-dual gap estimators for *a posteriori* error analysis of nonsmooth minimization problems. *ESAIM Math. Model. Numer. Anal.*, 54(5):1635–1660, 2020.

[14] S. Bassetto, C. Cancès, G. Enchéry, and Q. H. Tran. Robust Newton solver based on variable switch for a finite volume discretization of Richards equation. In *Finite Volumes for Complex Applications IX—Methods, Theoretical Aspects, Examples—FVCA 9, Bergen, Norway, June 2020*, volume 323 of *Springer Proc. Math. Stat.*, pages 385–393. Springer, Cham, 2020.

[15] S. Bassetto, C. Cancès, G. Enchéry, and Q.-H. Tran. Upstream mobility finite volumes for the Richards equation in heterogenous domains. *ESAIM: M2AN*, 55(5):2101–2139, 2021.

[16] S. Bassetto, C. Cancès, G. Enchéry, and Q.-H. Tran. On several numerical strategies to solve Richards' equation in heterogeneous media with finite volumes. *Comput. Geosci.*, 26(5):1297–1322, 2022.

[17] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC. Springer, Cham, second edition, 2017.

[18] L. Belenki, L. Diening, and C. Kreuzer. Optimality of an adaptive finite element method for the p-Laplacian equation. *IMA J. Numer. Anal.*, 32(2):484–510, 2012.

[19] J. Blechta, J. Málek, and M. Vohralík. Localization of the $W^{-1,q}$ norm for local a posteriori efficiency. *IMA J. Numer. Anal.*, 40(2):914–950, 2020.

[20] D. Boffi, F. Brezzi, and M. Fortin. *Mixed finite element methods and applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2013.

[21] D. Braess and J. Schöberl. Equilibrated residual error estimator for edge elements. *Math. Comp.*, 77(262):651–672, 2008.

[22] K. Brenner and C. Cancès. Improving Newton's method performance by parametrization: the case of the Richards equation. *SIAM J. Numer. Anal.*, 55(4):1760–1785, 2017.

[23] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991.

[24] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[25] R. H. Brooks and A. T. Corey. Properties of Porous Media Affecting Fluid Flow. *J. Irrig. Drain. Div.*, 92(2):61–88, 1966.

[26] E. Carson and N. J. Higham. A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. *SIAM J. Sci. Comput.*, 39(6):A2834–A2856, 2017.

[27] E. Carson and N. J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM J. Sci. Comput.*, 40(2):A817–A847, 2018.

[28] C. Carstensen, M. Feischl, M. Page, and D. Praetorius. Axioms of adaptivity. *Comput. Math. Appl.*, 67(6):1195–1253, 2014.

[29] C. Carstensen, M. Maischak, D. Praetorius, and E. P. Stephan. Residual-based a posteriori error estimate for hypersingular equation on surfaces. *Numer. Math.*, 97(3):397–425, 2004.

[30] J. M. Cascón, C. Kreuzer, R. H. Nochetto, and K. G. Siebert. Quasi-optimal convergence rate for an adaptive finite element method. *SIAM J. Numer. Anal.*, 46(5):2524–2550, 2008.

[31] M. A. Celia, E. T. Bouloutas, and R. L. Zarba. A general mass-conservative numerical solution for the unsaturated flow equation. *Water Resour. Res.*, 26(7):1483–1496, 1990.

[32] A. L. Chaillou and M. Suri. Computable error estimators for the approximation of nonlinear problems by linearized models. *Comput. Methods Appl. Mech. Eng.*, 196(1-3):210–224, 2006.

[33] B. Chen, X. Chen, and C. Kanzow. A penalized Fischer-Burmeister NCP-function. *Mathematical Programming. A Publication of the Mathematical Programming Society*, 88(1, Ser. A):211–216, 2000.

[34] Z. Chen, G. Huan, and Y. Ma. *Computational Methods for Multiphase Flows in Porous Media*. Society for Industrial and Applied Mathematics, 2006.

[35] R. Cottereau, L. Chamoin, and P. Díez. Strict error bounds for linear and nonlinear solid mechanics problems using a patch-based flux-free method. *Mechanics & Industry*, 11(3-4):249–254, 2010.

[36] R. Cottereau, P. Díez, and A. Huerta. Strict error bounds for linear solid mechanics problems using a subdomain-based flux-free method. *Comput. Mech.*, 44(4):533–547, 2009.

[37] C. W. Curtis. *Linear algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, fourth edition, 1993. An introductory approach.

[38] P. I. Davies, N. J. Higham, and F. Tisseur. Analysis of the Cholesky method with iterative refinement for solving the symmetric definite generalized eigenproblem. *SIAM J. Matrix Anal. Appl.*, 23(2):472–493, 2001.

[39] R. De Boer. *Theory of Porous Media*. Springer, Berlin, Heidelberg, 2000.

[40] K. Deckelnick, G. Dziuk, and C. M. Elliott. Computation of geometric partial differential equations and mean curvature flow. *Acta Numer.*, 14:139–232, 2005.

[41] P. Destuynder and B. Métivet. Explicit error bounds in a conforming finite element method. *Math. Comp.*, 68(228):1379–1396, 1999.

[42] P. Deuflhard. *Newton methods for nonlinear problems*, volume 35 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2004.

[43] D. Di Pietro, M. Vohralík, and S. Yousef. Adaptive regularization, linearization, and discretization and a posteriori error control for the two-phase Stefan problem. *Math. Comp.*, 84(291):153–186, 2015.

[44] H. J. G. Diersch and P. Perrochet. On the primary variable switching technique for simulating unsaturated–saturated flows. *Adv. Water Res.*, 23(3):271–301, 1999.

[45] V. Dolejší, A. Ern, and M. Vohralík. *hp*-adaptation driven by polynomial-degree-robust a posteriori error estimates for elliptic problems. *SIAM J. Sci. Comput.*, 38(5):A3220–A3246, 2016.

[46] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM J. Numer. Anal.*, 33(3):1106–1124, 1996.

[47] L. El Alaoui, A. Ern, and M. Vohralík. Guaranteed and robust a posteriori error estimates and balancing discretization and linearization errors for monotone nonlinear problems. *Comput. Methods Appl. Mech. Eng.*, 200(37-40):2782–2795, 2011.

[48] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1996.

[49] A. Ern and J.-L. Guermond. *Theory and practice of finite elements*, volume 159 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2004.

[50] A. Ern and J.-L. Guermond. *Finite elements I—Approximation and interpolation*, volume 72 of *Texts in Applied Mathematics*. Springer, Cham, 2021.

[51] A. Ern, S. Nicaise, and M. Vohralík. An accurate $\mathbf{H}(\mathrm{div})$ flux reconstruction for discontinuous Galerkin approximations of elliptic problems. *C. R. Math. Acad. Sci. Paris*, 345(12):709–712, 2007.

[52] A. Ern, I. Smears, and M. Vohralík. Discrete p-robust $\boldsymbol{H}(\mathrm{div})$-liftings and a posteriori estimates for elliptic problems with H^{-1} source terms. *Calcolo*, 54(3):1009–1025, 2017.

[53] A. Ern and M. Vohralík. Adaptive inexact Newton methods with a posteriori stopping criteria for nonlinear diffusion PDEs. *SIAM J. Sci. Comput.*, 35(4):A1761–A1791, 2013.

[54] A. Ern and M. Vohralík. Stable broken H^1 and $H(\mathrm{div})$ polynomial extensions for polynomial-degree-robust potential and flux reconstruction in three space dimensions. *Math. Comp.*, 89(322):551–594, 2020.

[55] R. E. Ewing. Finite element methods for nonlinear flows in porous media. volume 51, pages 421–439. 1985. FENOMECH '84, Part I, II (Stuttgart, 1984).

[56] P. A. Forsyth, Y. S. Wu, and K. Pruess. Robust numerical methods for saturated-unsaturated flow with dry initial conditions in heterogeneous media. *Adv. Water Resour.*, 18(1):25–38, 1995.

[57] A. Friedman. The Stefan problem in several space variables. *Trans. Amer. Math. Soc.*, 133(1):51–87, 1968.

[58] F. Févotte, A. Rappaport, and M. Vohralík. Adaptive regularization, discretization, and linearization for nonsmooth problems based on primal–dual gap estimators. *Comput. Methods Appl. Mech. Eng.*, 418:116558, 2024.

[59] G. Gantner, A. Haberl, D. Praetorius, and B. Stiftner. Rate optimal adaptive FEM with inexact solver for nonlinear operators. *IMA J. Numer. Anal.*, 38(4):1797–1831, 2018.

[60] E. M. Garau, P. Morin, and C. Zuppa. Convergence of an adaptive Kačanov FEM for quasi-linear problems. *Appl. Numer. Math.*, 61(4):512–529, 2011.

[61] D. Göddeke. Fast and accurate finite-element multigrid solvers for PDE simulations on GPU clusters. 2011.

[62] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[63] M. Guo, W. Han, and H. Zhong. Legendre-Fenchel duality and a generalized constitutive relation error. arXiv preprint 1611.05589, 2016.

[64] A. Haberl, D. Praetorius, S. Schimanko, and M. Vohralík. Convergence and quasi-optimal cost of adaptive algorithms for nonlinear operators including iterative linearization and algebraic solver. *Numer. Math.*, 147(3):679–725, 2021.

[65] W. Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1985.

[66] W. Han. A posteriori error analysis for linearization of nonlinear elliptic problems and their discretizations. *Math. Methods Appl. Sci.*, 17(7):487–508, 1994.

[67] W. Han. *A posteriori error analysis via duality theory*, volume 8 of *Advances in Mechanics and Mathematics*. Springer-Verlag, New York, 2005. With applications in modeling and numerical approximations.

[68] A. Harnist, K. Mitra, A. Rappaport, and M. Vohralík. Robust energy a posteriori estimates for nonlinear elliptic problems. HAL preprint: hal-04033438, 2023.

[69] P. Heid, D. Praetorius, and T. P. Wihler. Energy contraction and optimal convergence of adaptive iterative linearized finite element methods. *Comput. Methods Appl. Math.*, 21(2):407–422, 2021.

[70] P. Heid and T. P. Wihler. On the convergence of adaptive iterative linearized Galerkin methods. *Calcolo*, 57(3), 2020.

[71] N. J. Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

[72] J. Hiriart-Urruty and C. Lemarechal. *Convex Analysis and Minimization Algorithms I: Fundamentals*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1996.

[73] B. Hofmann, B. Kaltenbacher, C. Pöschl, and O. Scherzer. A convergence rates result for Tikhonov regularization in Banach spaces with non-smooth operators. *Inverse Problems*, 23(3):987–1010, 2007.

[74] P. Houston, E. Süli, and T. P. Wihler. A posteriori error analysis of *hp*-version discontinuous Galerkin finite-element methods for second-order quasi-linear elliptic PDEs. *IMA J. Numer. Anal.*, 28(2):245–273, 2008.

[75] D. Illiano, I. S. Pop, and F. A. Radu. Iterative schemes for surfactant transport in porous media. *Comput. Geosci.*, 25(2):805–822, 2021.

[76] F. Irgens. *Rheology and Non-Newtonian Fluids.* Springer International Publishing, Cham, 2014.

[77] W. Jäger and J. Kačur. Solution of porous medium type systems by linear approximation schemes. *Numer. Math.*, 60(1):407–427, 1991.

[78] W. Jäger and J. Kačur. Solution of doubly nonlinear and degenerate parabolic problems by relaxation schemes. *ESAIM: M2AN*, 29(5):605–627, 1995.

[79] P. Jenny, H. A. Tchelepi, and S. H. Lee. Unconditionally convergent nonlinear solver for hyperbolic conservation laws with S-shaped flux functions. *J. Comput. Phys.*, 228(20):7497–7512, 2009.

[80] C. T. Kelley. *Iterative methods for linear and nonlinear equations*, volume 16 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. With separately available software.

[81] K.-Y. Kim. A posteriori error estimators for locally conservative methods of nonlinear elliptic problems. *Appl. Numer. Math.*, 57(9):1065–1080, 2007.

[82] S. Kräutle. The semismooth Newton method for multicomponent reactive transport with minerals. *Water Res.*, 34(1):137–151, 2011.

[83] P. Ladevèze and N. Moës. A posteriori constitutive relation error estimators for nonlinear finite element analysis and adaptive control. In *Advances in adaptive computational methods in mechanics (Cachan, 1997)*, volume 47 of *Stud. Appl. Mech.*, pages 231–256. Elsevier Sci. B. V., Amsterdam, 1998.

[84] P. Ladevèze and D. Leguillon. Error estimate procedure in the finite element method and applications. *SIAM J. Numer. Anal.*, 20(3):485–509, 1983.

[85] M. G. Larson and A. J. Niklasson. A conservative flux for the continuous Galerkin method based on discontinuous enrichment. *Calcolo*, 41(2):65–76, 2004.

[86] F. Lehmann and Ph. Ackerer. Comparison of iterative methods for improved solutions of the fluid flow equation in partially saturated porous media. *Transport Porous Med.*, 31(3):275–292, 1998.

[87] F. List and F. A. Radu. A study on iterative methods for solving Richards' equation. *Comput. Geosci.*, 20(2):341–353, 2016.

[88] S. F. McCormick, J. Benzaken, and R. Tamstorf. Algebraic error analysis for mixed-precision multigrid solvers. *SIAM J. Sci. Comput.*, 0(0):S392–S419, 2021.

[89] A. Miraçi, J. Papež, and M. Vohralík. A multilevel algebraic error estimator and the corresponding iterative solver with p-robust behavior. *SIAM J. Numer. Anal.*, 58(5):2856–2884, 2020.

[90] A. Miraçi. *A-posteriori-steered and adaptive p-robust multigrid solvers.* Theses, Sorbonne Université, Dec. 2020.

[91] A. Miraci, J. Papež, and M. Vohralík. A-posteriori-steered p-robust multigrid with optimal step-sizes and adaptive number of smoothing steps. *SIAM J. Sci. Comput.*, 0(0):S117–S145, 0.

[92] K. Mitra and I. S. Pop. A modified L-scheme to solve nonlinear diffusion problems. *Comp. & Math. Appl.*, 77(6):1722–1738, 2019.

[93] K. Mitra and M. Vohralík. Guaranteed, locally efficient, and robust a posteriori estimates for nonlinear elliptic problems in iteration-dependent norms. An orthogonal decomposition result based on iterative linearization. HAL preprint hal-04156711, 2022.

[94] K. Mitra and M. Vohralík. A posteriori error estimates for the Richards equation. *Math. Comp.* (2024), accepted for publication, 2022.

[95] P. Neittaanmäki and S. Repin. A posteriori error identities for nonlinear variational problems. *Ann. Acad. Rom. Sci. Ser. Math. Appl.*, 7(1):157–172, 2015.

[96] R. H. Nochetto. Error estimates for multidimensional singular parabolic problems. *Japan Journal of Applied Mathematics*, 4(1):111–138, 1987.

[97] R. H. Nochetto and A. Veeser. Primer of adaptive finite element methods. In *Multiscale and adaptivity: modeling, numerics and applications*, volume 2040 of *Lecture Notes in Math.*, pages 125–225. Springer, Heidelberg, 2012.

[98] S. Pereverzev and E. Schock. On the adaptive selection of the parameter in regularization of ill-posed problems. *SIAM J. Numer. Anal.*, 43(5):2060–2076, 2005.

[99] I. S. Pop, F. Radu, and P. Knabner. Mixed finite elements for the Richards' equation: Linearization procedure. *J. Comput. Appl. Math.*, 168(1-2):365–373, 2004.

[100] I. S. Pop and B. Schweizer. Regularization schemes for degenerate Richards equations and outflow conditions. *Math. Models Method Appl. Sci.*, 21(8):1685–1712, 2011.

[101] W. Prager and J. L. Synge. Approximations in elasticity based on the concept of function space. *Q. Appl. Math.*, 5:241–269, 1947.

[102] H.-D. Qi and L.-Z. Liao. A smoothing Newton method for general nonlinear complementarity problems. *Comput. Optim. Appl.*, 17(2):231–253, Dec. 2000.

[103] L. Qi and D. Sun. Smoothing functions and smoothing Newton method for complementarity and variational inequality problems. *J. Optim. Theory Appl.*, 113(1):121–147, 2002.

[104] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Math. Prog.*, 58(1):353–367, 1993.

[105] F. A. Radu, I. S. Pop, and P. Knabner. Newton-type methods for the mixed finite element discretization of some degenerate parabolic equations. In A. B. de Castro, D. Gómez, P. Quintela, and P. Salgado, editors, *Numer. Math. Adv. Appl.*, pages 1192–1200, Berlin, Heidelberg, 2006. Springer.

[106] M. M. Rao and Z. D. Ren. *Theory of Orlicz spaces*, volume 146 of *Monographs and Textbooks in Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, 1991.

[107] S. Repin. *A posteriori estimates for partial differential equations*, volume 4 of *Radon Series on Computational and Applied Mathematics*. Walter de Gruyter GmbH & Co. KG, Berlin, 2008.

[108] S. I. Repin. A posteriori error estimates for approximate solutions of variational problems with functionals of power growth. *Journal of Mathematical Sciences*, 101(5):3531–3538, 2000.

[109] S. I. Repin. A posteriori error estimation for variational problems with uniformly convex functionals. *Math. Comp.*, 69(230):481–500, 2000.

[110] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[111] M. Slodička. A robust and efficient linearization scheme for doubly nonlinear and degenerate parabolic problems arising in flow in porous media. *SIAM J. Sci. Comput.*, 23(5):1593–1614, 2002.

[112] I. Smears and M. Vohralík. Simple and robust equilibrated flux a posteriori estimates for singularly perturbed reaction-diffusion problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 54(6):1951–1973, 2020.

[113] E. M. Stein and R. Shakarchi. *Real analysis*, volume 3 of *Princeton Lectures in Analysis*. Princeton University Press, Princeton, NJ, 2005.

[114] J. S. Stokke, K. Mitra, E. Storvik, J. W. Both, and F. A. Radu. An adaptive solution strategy for richards' equation. *Computers & Mathematics with Applications*, 152:155–167, 2023.

[115] R. Tamstorf, J. Benzaken, and S. F. McCormick. Discretization-error-accurate mixed-precision multigrid solvers. *SIAM J. Sci. Comput.*, 0(0):S420–S447, 2021.

[116] M. T. van Genuchten. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am. J.*, 44(5):892–898, 1980.

[117] J. L. Vázquez. An introduction to the mathematical theory of the porous medium equation. In *Shape optimization and free boundaries (Montreal, PQ, 1990)*, volume 380 of *NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci.*, pages 347–389. Kluwer Acad. Publ., Dordrecht, 1992.

[118] F. Verdugo and S. Badia. The software design of Gridap: a finite element package based on the Julia JIT compiler. *Comput. Phys. Commun.*, 276:108341, 2022.

[119] R. Verfürth. A posteriori error estimation and adaptive mesh-refinement techniques. In *Proceedings of the Fifth International Congress on Computational and Applied Mathematics (Leuven, 1992)*, volume 50, pages 67–83, 1994.

[120] R. Verfürth. *A posteriori error estimation techniques for finite element methods*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2013.

[121] M. Vlasák. On polynomial robustness of flux reconstructions. *Appl. Math.*, 65(2):153–172, 2020.

[122] Z. Wan, H. Li, and S. Huang. A smoothing inexact Newton method for nonlinear complementarity problems. *Abstr. Appl. Anal.*, 2015:e731026, 2015.

[123] X. Wang and H. A. Tchelepi. Trust-region based solver for nonlinear transport in heterogeneous porous media. *J. Comput. Phys.*, 253:114–137, 2013.

[124] J. H. Wilkinson. *Rounding errors in algebraic processes*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1963.

[125] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel. Optimization of sparse matrixvector multiplication on emerging multicore platforms. *Parallel Computing*, 35(3):178–194, 2009. Revolutionary Technologies for Acceleration of Emerging Petascale Applications.

[126] J. Xia. Efficient structured multifrontal factorization for general large sparse matrices. *SIAM J. Sci. Comput.*, 35(2):A832–A860, 2013.

[127] E. H. Zarantonello. *Solving Functional Equations by Contractive Averaging*. Tech. Report 160, Mathematics Research Center. United States Army, University of Wisconsin, 1960.

[128] P. Zardoshti, F. Khunjush, and H. Sarbazi-Azad. Adaptive sparse matrix representation for efficient matrixvector multiplication. *The Journal of Supercomputing*, 72(9):33663386, 2015.

[129] E. Zeidler. *Nonlinear functional analysis and its applications. II/B*. Springer-Verlag, New York, 1990.