Multilevel minimization in trust-region framework

Algorithmic and software developments

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

presented by
Alena Kopaničáková

under the supervision of
Rolf Krause

January 2021

Dissertation Committee

Michael Multerer	Università della Svizzera italiana, Switzerland
Olaf Schenk	Università della Svizzera italiana, Switzerland
Serge Gratton	Toulouse INP - IRIT
Rasmus Tamstorf	Walt Disney Animation Studios
Martin Weiser	Zuse Institute Berlin

Dissertation accepted on 28 January 2021

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

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

Alena Kopaničáková
Lugano, 28 January 2021

Abstract

The field of scientific computing is associated with the modeling of complex physical phenomena. The resulting numerical models are often described by differential equations, which, in many cases, can be related to non-convex minimization problems. Thus, after discretization, the solution of large-scale non-convex optimization problem is required. Various iterative solution strategies can be used to solve such optimization problems. However, the convergence speed of the majority of them deteriorates rapidly with increasing problem size. Multilevel methods are known to overcome this difficulty, and therefore we focus on a class of globally convergent multilevel solution strategies called the recursive multilevel trust-region (RMTR) method. The RMTR method combines globalization properties of trust-region method and the efficiency of multilevel methods. Despite its robustness and efficiency, the practical implementation of the RMTR method is a technically demanding task, which relies upon a suitable multilevel framework. This framework requires the careful design of two main components: i) multilevel hierarchy and transfer operators, ii) coarse-level models. To maximize the efficiency of the RMTR method, these components must be created with knowledge of the particular optimization problem in mind.

In this thesis, we propose three novel variants of the RMTR method. Our first variant of the RMTR method is tailored for solving phase-field fracture problems. It employs novel coarse-level models, that allow the representation of fine-level fractures on the coarser levels. Our second RMTR variant is developed for thin-shell cloth simulations. Here, we employ a subdivision-based multilevel hierarchy and transfer operators. Our third variant of the RMTR method is designed for the training of the deep residual networks (ResNets). We construct the multilevel hierarchy and transfer operators by leveraging a dynamical system's view-point, which casts ResNet as the discretization of an initial value problem. We analyze the convergence properties of all three novel variants of the RMTR method. To this aim, we consider numerical examples from respective scientific fields. A comparison with a single-level trust-region method is made and demonstrates the efficiency of the proposed RMTR variants. Furthermore, we introduce our open-source library UTOPIA, which incorporates the parallel implementation of the multilevel methods presented in this work. Weak and strong scaling properties of our implementation are investigated up to $12,000$ processors and a billion degrees of freedom.

Acknowledgements

Firstly, I would like to thank my advisor, Prof. Rolf Krause, for being brave and adventurous enough to give me the opportunity to pursue a PhD. I am grateful for his support, patient supervision, and many scientific discussions. I am also thankful to him for initiating many scientific collaborations that took place during my PhD. I greatly appreciate the opportunity to co-supervise several master students and present my work at various international conferences.

I also wish to express my gratitude to the other members of my dissertation committee, Prof. Serge Gratton, Prof. Michael Multerer, Prof. Olaf Schenk, Dr. Rasmus Tamstorf, and Dr. Martin Weiser, for their time and interest.

My sincere thanks also go to Carola Bilgen and Prof. Weinberg for long-lasting collaboration regarding phase-field fracture modelling. Our collaboration throughout the course of the SPP1748 project improved my understanding of phase-field fracture models. Moreover, I am incredibly grateful to Dr. Rasmus Tamstorf for his hospitality during my stay at Disney animation studios. Without his knowledge about cloth simulations, work presented in Chapter 5 would have not been possible. I would also like to thank Fatemeh Chegini and Dr. Martin Weiser for introducing me to the field of inverse problems and multi-fidelity multilevel methods. Attempting to solve their inverse problems (from the field of electro-cardiology) triggered many ideas as well as algorithmic and code developments contained in this work. I am also particularly grateful to late Prof. Peter Deuflhard for many useful discussions during our work on the affine similar trust-region method. The knowledge that I gained during this collaboration enhanced greatly my understanding of iterative solution strategies.

My sincere thanks go to all contributors of the UTOPIA library. Especially, I would like to thank Dr. Patrick Zulian for his time and effort spent developing the library and for many useful discussions regarding my code. I would like to thank Dr. Maria Nestola, for many scientific discussions and for the help with the MOOSE framework in the initial phase of my PhD. My sincere thanks also go to Andreas Fink and Dr. Nur Fadel for their technical support regarding the Piz Daint supercomputer. Developments made by Patrick, Maria, Andreas, and Nur contributed to obtaining large-scale scaling results presented in Section 4.5.3.

I am thankful to my colleagues at the ICS for a friendly working atmosphere and countless scientific discussions. Special thanks to Arne Hansen and Toby Simpson for

proof-reading parts of this thesis. I am also particularly grateful to master students, that I co-supervised: Lisa Gaedke-Merzhaeuser, Vanessa Braglia, and Samuel Cruz. Their work served as motivation to design a variant of the RMTR method for machine learning application, presented in Chapter 6.

Especially, I am thankful to Dr. Hardik Kothari for many scientific and non-scientific discussions, for his continuous help with the ICS cluster, for proof-reading of this thesis, and for his company during the late-night working sessions. I would also like to thank him for his love, support, encouragement, and attempts to calm me down whenever something would not work.

Last but not least, I am extremely grateful to my parents and my brothers for their love, encouragement, and never-ending support.

Contents

Figures

Tables

Introduction

The solution of non-convex optimization problems is of paramount interest for many scientific applications. In this work, we focus on solving optimization problems from three different research areas, namely fracture mechanics, cloth animation, and machine learning. Although the considered optimization problems model different physical phenomena, they can be represented in the following abstract form:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \mathbf{l} \leq \boldsymbol{x} \leq \mathbf{u}, \end{aligned} \tag{P}$$

where $n \in \mathbb{N}$ is typically very large. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and non-convex. Here, the lower bound $\mathbf{l} \in (\mathbb{R} \cup \{-\infty\})^n$, and upper bound $\mathbf{u} \in (\mathbb{R} \cup \{\infty\})^n$ provide constraints on the variable $\boldsymbol{x} \in \mathbb{R}^n$.

A common approach for solving the optimization problem (P) is to use variants of Newton's method. Although Newton's method converges quadratically in a neighbourhood of a solution, its convergence can not be guaranteed if a poor initial guess is supplied and no globalization strategy is employed. Furthermore, each Newton step requires the solution of a large-scale linear system. Solving such large-scale systems using direct methods is often infeasible due to time and memory constraints. To ease the computational burden, iterative methods are often employed in practice. Unfortunately, the convergence rate of iterative methods depends on the condition number of the underlying linear systems. As a consequence, their convergence rate deteriorates with increasing problem size, i.e., the number of iterations required to reach the desired tolerance grows rapidly with the number of unknowns. For example, if the optimization problem (P) arises from the discretization of an eliptic partial differential equations (PDEs) with the Lagrangian finite elements, the condition number of the underlying linear systems increases with decreasing discretization parameter. Thus, solving large-scale discretized PDEs, with direct solvers becomes prohibitive in practice.

Globalization strategies, such as line-search [DS96], trust-region [CGT00], or adaptive cubic regularization [CGT12], ensure convergence to first-order critical points regardless of the quality of the initial guess. However, the use of globalization strategies does not tackle difficulties related to the ill-conditioning of the resulting linear systems. Multilevel methods are known to be optimal solution strategies for many problems, as

their convergence rate is often independent of the problem size, and the number of required arithmetic operations grows proportionally with the number of unknowns. The main idea behind multilevel methods is to employ a multilevel hierarchy of auxiliary problems, which are then solved in order to speed up the solution process of the original problem. A multilevel hierarchy of problems can be obtained naturally for some practical applications. For example, when numerically solving discretized PDEs, the multilevel hierarchy can be generated using increasingly finer discretizations of the same PDE. However, for many practical applications, it is not trivial to create a suitable multilevel hierarchy and novel methods must be devised.

The recursive multilevel trust-region (RMTR) method represents a class of nonlinear multilevel solution strategies, which are suitable for solving large-scale non-convex constrained optimization problems [GST08, GMTW08, GK09b]. The RMTR method combines the global convergence properties of the trust-region method with the efficiency of multilevel methods. This gives rise to a globally convergent multilevel minimization method. Despite the robustness and efficiency of the RMTR method, its practical implementation is cumbersome for the following reasons:

- The RMTR method requires knowledge about a multilevel hierarchy of auxiliary problems. The choice of a suitable multilevel hierarchy often varies for different optimization problems. Indeed, in order to enhance the efficiency of the RMTR method, a multilevel hierarchy should be built by taking into account the underlying structure of the given optimization problem.

- In order to solve large-scale problems, the devised RMTR algorithms should be be implemented such that they take advantage of modern computer architectures.

In this thesis, we focus on both algorithmic and software developments of the RMTR method. Part I is concerned with an abstract RMTR framework. We provide a detailed description of the method and discuss how to obtain an efficient algorithm in practice. Particular attention is given to the construction of multilevel hierarchy and transfer operators, and to the choice of coarse-level models. Furthermore, we discuss several algorithms, that can be used to handle point-wise constraints in multilevel settings. We also present an open-source library UTOPIA, which incorporates the parallel implementation of multiple variants of the RMTR method.

In Part II, we consider three non-convex optimization problems from the field of fracture mechanics, cloth animation, and machine learning, respectively. We solve these optimization problems using the RMTR method. Since all three optimization problems vary in several aspects, we design three novel variants of the RMTR method. The presented variants differ with respect to the techniques used to create the multilevel hierarchy and transfer operators as well as in the choice of coarse-level models. We analyze all variants by means of several numerical examples from their respective scientific fields.

Outline

This thesis is divided into two parts: Part I called Trust-region based optimization methods and Part II called Applications.

Part I

In Chapter 1, we formally introduce the optimization problem considered in this work. We also present the basic trust-region method and discuss techniques, which can be used to solve the resulting trust-region subproblems. Moreover, we provide a brief introduction to limited-memory secant methods, which can be used to approximate Hessian.

In Chapter 2, we introduce the recursive multilevel trust-region (RMTR) method. We describe the method and identify its three main building blocks: i) multilevel hierarchy and transfer operators, ii) level-dependent objective functions, and iii) multilevel treatment of trust-region and variable bounds. We discuss all three building blocks in detail and provide general guidance on their construction in practical application. Moreover, we present a novel variant of the RMTR method, which utilizes an active set strategy.

Chapter 3 provides a brief overview of the UTOPIA library [ZKN+16]. Our discussion focuses specifically on the software design of the trust-region module, which incorporates methods described in Chapters 1 and 2.

Part II

Part II consists of three chapters with the following structure: Firstly, we introduce a particular non-convex minimization problem. Secondly, we propose a variant of the RMTR method to solve the given minimization problem efficiently. Lastly, the proposed variant of the RMTR method is tested using several numerical examples.

In Chapter 4, we introduce optimization problems arising in phase-field fracture simulations and discuss main challenges associated with designing a multilevel method. We propose a variant of the RMTR method, which employs novel level-dependent objective functions. Convergence properties of the proposed RMTR method are demonstrated using several numerical examples. A comparison with a staggered solution scheme as well as with a single-level trust-region method illustrates the efficiency of our method. Finally, we present strong and weak scaling properties of our implementation based on the UTOPIA library.

In Chapter 5, we introduce optimization problems arising in thin-shell cloth simulations. Our multilevel framework for cloth simulations is based on Catmull–Clark subdivision surfaces, which facilitate the generation of the mesh hierarchy and also provides the basis for the finite element discretization. The prolongation and restriction operators are similarly constructed based on the subdivision rules. Finally, we leverage a reverse subdivision operator to transfer iterates from fine levels to coarser levels. In the end, we

present several numerical examples, which show a reduction in the number of iterations by several orders of magnitude when compared to a single-level trust-region method.

In Chapter 6, we introduce an optimization problem associated with the training process of Deep Residual Network (ResNet). We take advantage of the fact that the ResNet architecture can be formulated as a forward Euler discretization of a nonlinear initial value problem (IVP). The training of ResNet then consists of solving optimal control problems with a given dynamical system. Our multilevel framework takes advantage of the time-dependent nature of the IVP and creates a multilevel hierarchy by discretizing the original IVP with larger time-steps. The proposed variant of the RMTR method operates in a hybrid stochastic-deterministic regime and adaptively adjusts sample sizes during the training process. In contrast to the traditional stochastic optimization methods, our method also incorporates curvature information. We approximate the Hessian on each level of the multilevel hierarchy using a limited-memory secant method. Finally, we analyze the convergence behavior of our RMTR method and show its robustness using several numerical examples.

Contributions

Here, we provide a brief list of the contributions made by this thesis:

- Parallel open-source implementation of single-level and multilevel trust-region methods; including various constrained quadratic programming solvers and limited-memory secant methods.
- An active set variant of the recursive multilevel trust-region (RMTR) method. The method is designed to solve minimization problems with pointwise constraints.
- A variant of the RMTR method designed for solving phase-field fracture simulations. The proposed method utilizes novel level-dependent objective functions in order to obtain suitable coarse-grid corrections.
- A variant of the RMTR method for solving thin-shell cloth simulations. This variant of the RMTR method constructs multilevel hierarchy and transfer operators using subdivision surfaces.
- A variant of the RMTR method designed for training deep residual networks (ResNets). The developed method builds multilevel hierarchy and transfer operators by leveraging dynamical systems's view-point, which casts ResNet as the discretization of an initial value problem. Moreover, the proposed RMTR method operates in hybrid stochastic-deterministic regime, and incorporates curvature using limited-memory secant methods.

Part I

Trust-region based optimization methods

Chapter 1

Trust-region method

1.1 Definition of minimization problem

In this work, we focus on solving optimization problems of the following type:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\boldsymbol{x}), \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{F}, \end{aligned} \tag{1.1}$$

$\boldsymbol{x} \in \mathbb{R}^n$ is real vector with $n \geq 1$ components and $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable objective function, which is bounded from below. The minimization problem (1.1) is subjected to the bound constraints defined in terms of a feasible set $\mathcal{F} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{l} \leq \boldsymbol{x} \leq \mathbf{u}\}$, where $\mathbf{l}, \mathbf{u} \in (\mathbb{R} \cup \{-\infty\} \cup \{\infty\})^n$ denote the lower bound and upper bound, respectively. These constraints are defined component-wise. Thus, each component $(\boldsymbol{x})_k$ of $\boldsymbol{x}$ has to satisfy $(\mathbf{l})_k \leq (\boldsymbol{x})_k \leq (\mathbf{u})_k$, where $k \in \{1, \ldots, n\}$.

The formulation (1.1) is generic and comprises minimization problems arising in various scientific fields, for instance, solid mechanics, cloth animation, machine learning, etc. Indeed, the problem definition (1.1) includes also unconstrained problems by setting components of vector $\mathbf{l}$ and $\mathbf{u}$ to $-\infty$ and ∞, respectively. In this work, we focus on developing solution strategies for finding a minimizer of problem (1.1) in an efficient way.

Solutions

The function f is neither assumed to be quadratic nor convex. Hence, it might be challenging to find a global minimizer of (1.1). In this work, we are concerned to find (weak) local minimizers of problem (1.1). The point $\boldsymbol{x}_* \in \mathbb{R}^n$ is a weak local minimizer if there exists an open neighborhood $\mathcal{N}$ of $\boldsymbol{x}_*$ such that

$$f(\boldsymbol{x}_*) \leq f(\boldsymbol{x}), \quad \forall\, \boldsymbol{x} \in \mathcal{N} \cap \mathcal{F}.$$

The point $\boldsymbol{x}_* \in \mathbb{R}^n$ is a local minimizer if there is an open neighborhood $\mathcal{N}$ of $\boldsymbol{x}_*$ such that

$$f(\boldsymbol{x}_*) < f(\boldsymbol{x}), \quad \forall\, \boldsymbol{x} \in \mathcal{N} \cap \mathcal{F} \quad \text{and } \boldsymbol{x} \neq \boldsymbol{x}_*.$$

Necessary conditions

Suppose that $\boldsymbol{x}_* \in \mathcal{F}$ is a local minimizer of problem (1.1), then the following first-order necessary conditions are satisfied:

$$\begin{aligned} \nabla f(\boldsymbol{x}_*)_k &= 0, \quad \text{if} \quad (\mathbf{l})_k \leq (\boldsymbol{x}_*)_k \leq (\mathbf{u})_k, \\ \nabla f(\boldsymbol{x}_*)_k &\leq 0, \quad \text{if} \quad (\boldsymbol{x}_*)_k = (\mathbf{u})_k, \\ \nabla f(\boldsymbol{x}_*)_k &\geq 0, \quad \text{if} \quad (\mathbf{l})_k = (\boldsymbol{x}_*)_k. \end{aligned} \tag{1.2}$$

Here, we use the symbol $(\cdot)_k$ to denote the k-th component of a vector. Any point $\boldsymbol{x}_*$, which satisfies the conditions (1.2) is called first-order critical point or first-order stationary point. The conditions (1.2) are known as Karush-Kuhn-Tucker (KKT) conditions [NW06]. To guarantee that the point $\boldsymbol{x}_*$ is a (local) minimizer, the knowledge of higher-order derivatives of the function f is required. For thorough information, we refer the reader to References [NW06, CGT00].

Criticality measure

Iterative solution strategies search for the first-order critical points of problem (1.1) by producing a sequence of iterates $\{\boldsymbol{x}_i\}$, which converges to the first-order critical point $\boldsymbol{x}_*$. The practical implementation of these solution strategies requires a suitable criticality measure in order to verify, whether the first-order necessary conditions (1.2) are satisfied, or not. To this end, we employ the following criticality measure:

$$\mathcal{E}(\boldsymbol{x}) := \|\mathcal{P}(\boldsymbol{x} - \nabla f(\boldsymbol{x})) - \boldsymbol{x}\|, \tag{1.3}$$

where $\mathcal{P}$ is an orthogonal projection onto the feasible set, see Section A.1 for details. Alternative types of critical measures can be found, for example in References [CGT00, CGST93, GMT11].

Remark 1. *In the unconstrained case, the criticality measure* (1.3) *reduces to* $\mathcal{E}(\boldsymbol{x}) := \|\nabla f(\boldsymbol{x})\|$.

1.2 The basic trust-region method

A trust-region (TR) method is a convergence control strategy for minimizing nonlinear, convex/non-convex objective functions [CGT00]. At each iteration i, the TR method approximates the objective function f by means of a model $m_i : \mathbb{R}^n \to \mathbb{R}$. The model m_i

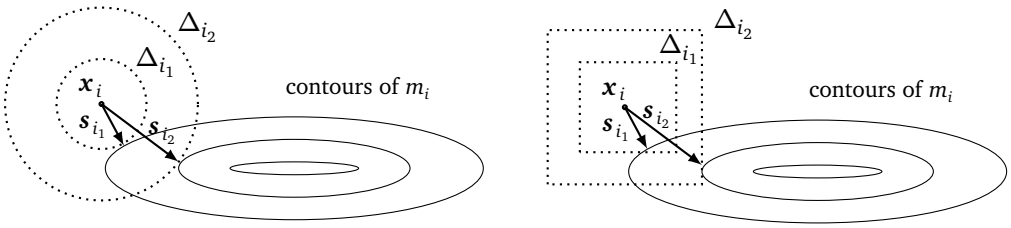

Figure 1.1: Step direction $\boldsymbol{s}_i$ is found as a minimizer of a model m_i within the trust-region radius Δ_i. *Left:* The shape of the trust-region is defined by the ℓ_2-norm. *Right:* The shape of the trust-region is defined by the ℓ_∞-norm.

is often constructed as a second-order Taylor approximation of the function f around the current iterate $\boldsymbol{x}_i \in \mathbb{R}^n$, given as

$$m_i(\boldsymbol{s}_i) = f_i + \langle \boldsymbol{g}_i, \boldsymbol{s}_i \rangle + \frac{1}{2}\langle \boldsymbol{s}_i, \mathbf{B}_i \boldsymbol{s}_i \rangle,$$

where we use the shortened notation $f_i := f(\boldsymbol{x}_i)$, $\boldsymbol{g}_i := \nabla f(\boldsymbol{x}_i)$. The symbol $\mathbf{B}_i$ denotes either the exact Hessian $\nabla^2 f(\boldsymbol{x}_i)$, or its numerical approximation. By $\langle \cdot, \cdot \rangle$, we denote the Euclidean inner product defined as $\langle \boldsymbol{x}, \mathbf{y} \rangle = \sum_{z=1}^{n} (\boldsymbol{x})_z (\mathbf{y})_z$ for any $\boldsymbol{x}, \mathbf{y} \in \mathbb{R}^n$.

The model m_i is only assumed to be an adequate representation of f in a certain region, called the trust-region, defined as the closed ball around current iterate $\boldsymbol{x}_i$, thus as

$$\mathcal{B}_i := \{\boldsymbol{x}_i + \boldsymbol{s}_i \in \mathbb{R}^n \mid \|\boldsymbol{s}_i\|_p \leq \Delta_i\}, \tag{1.4}$$

where $\Delta_i > 0$ stands for a trust-region radius. The choice of the norm $\|\cdot\|_p$ in (1.4) defines the shape of the trust-region. We consider only two types of trust-region shapes, defined by ℓ_2- and ℓ_∞-norm, denoted by $\|\cdot\|$ and $\|\cdot\|_\infty$, respectively. Figure 1.1 illustrates the trust-regions defined by both, ℓ_2-norm and ℓ_∞-norm.

At each iteration i, the trust-region method obtains the search direction $\boldsymbol{s}_i$ by approximately solving the following constrained quadratic minimization problem:

$$\begin{aligned} \underset{\boldsymbol{s}_i \in \mathbb{R}^n}{\text{minimize}} \quad & m_i(\boldsymbol{s}_i), \\ \text{subject to} \quad & \boldsymbol{x}_i + \boldsymbol{s}_i \in \mathcal{W}_i, \end{aligned} \tag{1.5}$$

where the working set $\mathcal{W}_i$ is defined as an intersection of the feasible set $\mathcal{F}$ with the trust-region $\mathcal{B}_i$, thus

$$\mathcal{W}_i = \mathcal{B}_i \cap \mathcal{F}.$$

Please note that the working set $\mathcal{W}_i$ has to be recomputed for each iteration, as the trust-region $\mathcal{B}_i$ is iteration dependent. The numerical evaluation of $\mathcal{W}_i$ can be performed in a component-wise manner, if the trust-region $\mathcal{B}_i$ is defined by the ℓ_∞-norm. In contrast,

Algorithm 1 Basic_TR(f, $\boldsymbol{x}_0$, $\mathcal{F}$, Δ_0, $i_{\max}$)

Require: $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\mathcal{F}$, $\Delta_0 \in \mathbb{R}$, $i_{\max} \in \mathbb{N}$
Constants: $\epsilon_g \in \mathbb{R}$

```
for i = 0, ..., i_max do
    W_i := B_i ∩ F                                          ▷ Generate working set

    minimize_{s_i ∈ R^n} m_i(s_i)                           ▷ Solve constrained QP problem
    subject to x_i + s_i ∈ W_i

    ρ_i = (f(x_i) − f(x_i + s_i)) / (m_i(x_i) − m_i(x_i + s_i))   ▷ Evaluate TR ratio

    [x_{i+1}, Δ_{i+1}] = Convergence_control(ρ_i, x_i, s_i, Δ_i)   ▷ Call Algorithm 2

    if E(x_{i+1}) ≤ ε_g then                                ▷ Check convergence
        return x_{i+1}, Δ_{i+1}                             ▷ Terminate
    end if
end for
return x_{i+1}, Δ_{i+1}
```

if the trust-region $\mathcal{B}_i$ is defined by the ℓ_2-norm, the evaluation of $\mathcal{W}_i$ is not trivial. For this reason, we restrict ourselves to use the ℓ_2-norm only if the problem at the hand is unconstrained.

The new iterate, also called the trial point, $\boldsymbol{x}_i + \boldsymbol{s}_i$, is not taken immediately. Instead, the trust-region algorithm decides whether the trial point is accepted and how to adjust the trust-region radius Δ_i. To this end, a trust-region ratio ρ_i is defined as the quotient of the actual reduction in f and the reduction predicted by the model m_i. Given a step $\boldsymbol{s}_i$, the trust-region ratio ρ_i is defined as

$$\rho_i = \frac{f(\boldsymbol{x}_i) - f(\boldsymbol{x}_i + \boldsymbol{s}_i)}{m_i(\boldsymbol{x}_i) - m_i(\boldsymbol{x}_i + \boldsymbol{s}_i)} = \frac{\text{actual reduction}}{\text{predicted reduction}}. \tag{1.6}$$

A value of ρ_i close to 1 indicates a good agreement between the model m_i and the function f. In this case, it is safe to accept the trial step and expand the trust-region radius. On the other hand, if the value of ρ_i is negative or close to 0, the algorithm must reject the trial step and shrink the trust-region radius. Algorithm 1 summarizes the described process.

Algorithm 2 Convergence_control(ρ_i, $\boldsymbol{x}_i$, $\boldsymbol{s}_i$, Δ_i)

Require: $\rho_i \in \mathbb{R}$, $\boldsymbol{x}_i \in \mathbb{R}^n$, $\boldsymbol{s}_i \in \mathbb{R}^n, \Delta_i \in \mathbb{R}$
Constants: $\eta_1 \in \mathbb{R}$, where $0 < \eta_1$

```
if ρ_i > η_1 then
    x_{i+1} := x_i + s_i                    ▷ Accept trial point
else
    x_{i+1} := x_i                          ▷ Reject trial point
end if
[Δ_{i+1}] = Radius_update(ρ_i, Δ_i)         ▷ Call Algorithm 3
return x_{i+1}, Δ_{i+1}
```

Algorithm 3 Radius_update

Require: ρ_i, $\Delta_i^l \in \mathbb{R}$
Constants: $\eta_1, \eta_2, \gamma_1, \gamma_2 \in \mathbb{R}$, where $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$

$$\Delta_{i+1} = \begin{cases} \gamma_1 \Delta_i, & \rho_i < \eta_1, \\ \Delta_i, & \rho_i \in [\eta_1, \eta_2], \\ \gamma_2 \Delta_i, & \rho_i > \eta_2 \end{cases}$$

return Δ_{i+1}

1.2.1 The solution of trust-region subproblem

The most challenging part of the trust-region method is solving the trust-region subproblem (1.5). Although, the convergence theory of the trust-region method does not require exact solution of Subproblem (1.5), the approximate solution $\boldsymbol{s}_i$ has to fulfill the following sufficient decrease condition:

$$m_i(\boldsymbol{x}_i) - m_i(\boldsymbol{x}_i + \boldsymbol{s}_i) \geq \kappa_{red}\, \mathcal{E}(\boldsymbol{x}_i) \min\left[1, \frac{\mathcal{E}(\boldsymbol{x}_i)}{1 + \|\mathbf{B}_i\|}, \Delta_i\right], \tag{SDC}$$

where $\kappa_{red} \in (0, 0.5)$. This condition is also known as the Cauchy condition and provides a lower bound on the model decrease (predicted reduction) obtained by following the search direction $\boldsymbol{s}_i$.

A search direction $\boldsymbol{s}_i$, that satisfies the (SDC) condition, can be obtained by several practical algorithms. In this section, we provide a brief overview of the algorithms that can be used to find a solution of arising trust-region subproblems. Our discussion has two parts, related to the trust-region defined by ℓ_2- and ℓ_∞-norms.

Finding ℓ_2-norm model minimizer

Let us assume that the minimization problem (1.1) is unconstrained and that the trust-region is defined by the Euclidean norm. Then, the search direction $\boldsymbol{s}$ is obtained by

solving the following quadratic programming (QP) minimization problem:

$$\begin{aligned} \underset{\boldsymbol{s}\in\mathbb{R}^n}{\text{minimize}} \quad & q(\boldsymbol{s}) := \langle \boldsymbol{g}, \boldsymbol{s}\rangle + \frac{1}{2}\langle \boldsymbol{s}, \mathbf{B}\boldsymbol{s}\rangle, \\ \text{subject to} \quad & \|\boldsymbol{s}\| \leq \Delta. \end{aligned} \tag{1.7}$$

Please note that we have dropped iteration subscripts and discarded the $f(\boldsymbol{x})$ term from the quadratic model, since it has no effect on the value of the minimizer $\boldsymbol{s}$.

A popular algorithm for solving problem (1.7) exactly is the Moré and Sorensen method (MSM) [Gay83, MS83, Heb73] and its variants, described in References [RSS01, Roj07, Hag01, TTWM09]. An accurate solution of (1.7) can be also obtained using subspace projection methods [EGG09, GRT10] or methods based on solving the generalized eigenvalue problem [GGM88, AINT17]. Although these methods compute the accurate solution of (1.7), their practical use is often limited to small scale problems and sequential programming environments. Alternatively one can solve trust-region subproblem (1.7) approximately, such that the (SDC) condition is fulfilled. For example, a search direction $\boldsymbol{s}$ can be obtained using Cauchy point-based algorithms [NW06]. Dogleg and double-Dogleg methods obtain the search direction by combining the steepest descent direction with an approximate Newton's step (convex case), or some other direction of sufficient negative curvature (non-convex case) [Pow70a, Pow70b, DM79]. The methods based on (preconditioned) Conjugate Gradient method, such as the Steihaug-Toint Conjugate Gradient (STCG) method [Ste83, Toi81], or the generalized Lanczos (GLM) method [GLRT99, ZS18] are also quite popular in practice.

Finding ℓ_∞-norm model minimizer

In this section, we assume that the trust-region is defined by the ℓ_∞-norm. The trust-region algorithm then obtains a search direction $\boldsymbol{s}$ by solving the following constrained quadratic minimization problem:

$$\begin{aligned} \underset{\boldsymbol{s}\in\mathbb{R}^n}{\text{minimize}} \quad & q(\boldsymbol{s}) := \langle \boldsymbol{g}, \boldsymbol{s}\rangle + \frac{1}{2}\langle \boldsymbol{s}, \mathbf{B}\boldsymbol{s}\rangle, \\ \text{subject to} \quad & \hat{\mathbf{l}} \leq \boldsymbol{s} \leq \hat{\mathbf{u}}, \end{aligned} \tag{1.8}$$

where variable bounds $\hat{\mathbf{l}}, \hat{\mathbf{u}}$ result from an intersection of the current trust-region $\mathcal{B}_i$ with the feasible set $\mathcal{F}$. Thus, the k-th component of $\hat{\mathbf{l}}$ and $\hat{\mathbf{u}}$ is obtained as

$$(\hat{\mathbf{l}})_k = \max[-\Delta, (\mathbf{l} - \boldsymbol{x}_i)_k], \qquad (\hat{\mathbf{u}})_k = \min[\Delta, (\mathbf{u} - \boldsymbol{x}_i)_k],$$

where $\boldsymbol{x}_i$ denotes solution at i-th iterate.

Solving the trust-region subproblem (1.8), defined by the ℓ_∞-norm, exactly is more challenging than solving the trust-region subproblem (1.7), defined by the ℓ_2-norm. In particular, the minimization problem (1.8) is non-deterministic polynomial-time (NP)-hard, when $\mathbf{B}$ is indefinite [CGT00]. However, as stated above, the convergence theory of

trust-region methods does not require the exact solution of (1.8). This is very convenient, as there are many algorithms, which are efficient in obtaining an approximate minimizer of (1.8), for instance gradient projection methods [NW06, MT91], alternate minimization [HK94], active-set methods [Won11, Ulb11], or interior point methods [PW00]. For large scale problems, it is desirable to use variants of Krylov subspace methods, such as Prolongated Truncated Conjugate Gradient (PTCG) method [Yua00], or Modified Proportioning with Gradient Projections (MPGRP) method [DD07, Dos09]. Alternatively, one can also employ the monotone multigrid method [Kor94a], originally developed to solve obstacle and contact problems. In the end, we also note the variant of constrained successive coordinate minimization (SCM) method, which is shown to be especially effective when used within multilevel trust-region methods [GMTW08].

1.2.2 Implementation and numerical considerations

In this section, we discuss some practicalities and numerical considerations required for the efficient and numerically stable implementation of the trust-region method, Algorithm 1.

Choice of algorithmic constants and trust-region update rules

An Algorithm 1 requires particular values to prescribed for the parameters $\eta_1, \eta_2, \gamma_1, \gamma_2$. The parameters η_1, η_2 determine the quality of the trial point and are typically set to $\eta_1 = 0.05$ and $\eta_2 = 0.9$, see References [CGT00, CGT13] for reported empirical evidence.

The parameters γ_1, γ_2, and the trust-region updating rules are very important, as they prescribe the maximum length of the search direction. The update rule provided by Algorithm 3, with $\gamma_1 = 0.25$ and $\gamma_2 = 2.0$, is common in practice. More advanced update rules, such as the interpolation/extrapolation algorithm, can be found in References [DS96, Aue93].

Initialization of trust-region radius

The trust-region method, Algorithm 1, starts with some initial trust-region radius Δ_0. The value of Δ_0 should be chosen such that it captures the region where the model m_0 is an adequate representation of our objective function. Typically, we initialize Δ_0 using some heuristics, such as

$$\Delta_0 = 1, \qquad \text{or} \qquad \Delta_0 = \frac{1}{10}\|\boldsymbol{g}_0\|, \qquad \text{or} \qquad \Delta_0 = \frac{\|\boldsymbol{g}_0\|^2}{|\langle \boldsymbol{g}_0, \mathbf{B}_0 \boldsymbol{g}_0 \rangle|}. \tag{1.9}$$

The first two options were used in Reference [GHM80] and require almost no computational cost. The third option was suggested for quasi-Newton trust-region methods [Pow68], where the choice of the initial Hessian $\mathbf{B}_0$ may be imprecise. In this case,

the value of Δ_0 is chosen as the length of the quadratic model minimizer along the steepest descent direction.

Despite the fact that the initialization techniques (1.9) are fairly simple, they are quite effective in practice [CGT00]. A more sophisticated strategy, which determines the largest possible Δ_0 such that the model prediction is sufficiently close to the value of the true objective function, can be found in Reference [Sar97].

Evaluation of predicted reduction and the trust-region ratio

Numerical evaluation of the predicted reduction and trust-region ratio often suffers from cancelation errors [CGT00]. A numerically stable approach for computing the predicted reduction is to express the difference $m_i(\boldsymbol{x}_i) - m_i(\boldsymbol{x}_i + \boldsymbol{s}_i)$ explicitly as:

$$m_i(\boldsymbol{x}_i) - m_i(\boldsymbol{x}_i + \boldsymbol{s}_i) := -\left(\langle \boldsymbol{g}_i, \boldsymbol{s}_i \rangle + \frac{1}{2} \langle \boldsymbol{s}_i, \mathbf{B}_i \boldsymbol{s}_i \rangle \right). \tag{1.10}$$

Please note that certain trust-region subproblem solvers, such as the STCG, or the GLM methods, allow for the evaluation of (1.10) at negligible cost from already known quantities.

Following [CGT00], a numerically stable evaluation of the trust-region ratio ρ_i can be performed as follows:

$$\rho_i = \begin{cases} 1, & \text{if} \quad m(\boldsymbol{x} + \boldsymbol{s}) = f(\boldsymbol{x} + \boldsymbol{s}) \quad \text{or} \quad (|\delta f_i| < \epsilon \quad \text{and} \quad |\delta m_i| < \epsilon), \\ \frac{\delta f_i}{\delta m_i}, & \text{otherwise}, \end{cases}$$

where

$$\delta f_i = f(\boldsymbol{x}_i + \boldsymbol{s}_i) - f(\boldsymbol{x}_i) - (10 \boldsymbol{\varepsilon}_M \max(1, |f_i|)),$$
$$\delta m_i = m_i(\boldsymbol{x}_i + \boldsymbol{s}_i) - m_i(\boldsymbol{x}_i) - (10 \boldsymbol{\varepsilon}_M \max(1, |f_i|)).$$

Thus, if the difference in function value and model decrease is relatively small (a multiple of the machine precision $\boldsymbol{\varepsilon}_M$), we set the value of ρ_i to 1.

1.3 Quasi-Newton trust-region method

For many real-life applications, the exact Hessian $\nabla^2 f(\boldsymbol{x}_i)$ is not known, or it is impractical to obtain. Secant methods provide a framework for approximating the Hessian using the curvature information collected between two subsequent iterations. At iteration $i+1$, the Hessian approximation $\mathbf{B}_{i+1} \in \mathbb{R}^{n \times n}$ is constructed such that the following secant equation is satisfied:

$$\mathbf{B}_{i+1} \boldsymbol{s}_i = \mathbf{y}_i. \tag{1.11}$$

Here, the secant pair $\{\boldsymbol{s}_i, \mathbf{y}_i\}$ consists of the search direction $\boldsymbol{s}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$ and the variation of the gradient along this direction, given as $\mathbf{y}_i = \boldsymbol{g}_{i+1} - \boldsymbol{g}_i$.

The equation (1.11) imposes only n linear conditions in order to determine $n(n+1)/2$ entries of the matrix $\mathbf{B}_{i+1}$ (assuming the symmetry of $\mathbf{B}_{i+1}$). Therefore, in order to determine $\mathbf{B}_{i+1}$ uniquely, we need to impose an additional set of conditions. The particular choice of those conditions gives rise to different Hessian approximation schemes. We consider only two well-known and widely-used Hessian approximation schemes, namely the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [Bro70, Fle70, Gol70, Sha70] and the symmetric-rank-one (SR1) method [Bro67, Dav68, FM90]. For alternatives, such as the Davidon-Fletcher-Powell (DFP) method, we refer the reader to Reference [EM17] and citations therein.

The majority of secant methods construct the Hessian approximations $\mathbf{B}_{i+1}$ in a recursive manner. For instance, given a symmetric positive definite (SPD) Hessian approximation $\mathbf{B}_i$ at the i-th iteration, the BFGS method constructs $\mathbf{B}_{i+1}$ as follows:

$$\mathbf{B}_{i+1} = \mathbf{B}_i - \frac{\mathbf{B}_i \boldsymbol{s}_i \boldsymbol{s}_i^T \mathbf{B}_i}{\langle \boldsymbol{s}_i, \mathbf{B}_i \boldsymbol{s}_i \rangle} + \frac{\mathbf{y}_i \mathbf{y}_i^T}{\langle \mathbf{y}_i, \boldsymbol{s}_i \rangle}. \tag{1.12}$$

The obtained matrix $\mathbf{B}_{i+1}$ is guaranteed to be also SPD, if the curvature condition, $\langle \mathbf{y}_i, \boldsymbol{s}_i \rangle > 0$ is satisfied [LN89]. Due to the symmetric positive definiteness of $\mathbf{B}_{i+1}$, the BFGS formula is very popular in practice. Especially, it is well-suited for line-search globalization strategies, which are typically unable to exploit directions of negative curvature [NW06].

In the context of non-convex minimization problems, the true Hessian is not always positive definite. In this case, the SPD Hessian approximation obtained by the BFGS method does not approximate the actual Hessian well. The SR1 method does not enforce the Hessian approximations to be positive definite and, therefore, it allows for a more accurate Hessian approximation. This can be explored particularly well within the trust-region framework, which can take advantage of the directions of negative curvature. Given the symmetric matrix $\mathbf{B}_i$, the SR1 method constructs symmetric $\mathbf{B}_{i+1}$ as follows:

$$\mathbf{B}_{i+1} = \mathbf{B}_i - \frac{(\mathbf{y}_i - \mathbf{B}_i \boldsymbol{s}_i)(\mathbf{y}_i - \mathbf{B}_i \boldsymbol{s}_i)^T}{\boldsymbol{s}_i^T (\mathbf{y}_i - \mathbf{B}_i \boldsymbol{s}_i)}, \tag{1.13}$$

where we assume $\boldsymbol{s}_i^T (\mathbf{y}_i - \mathbf{B}_i \boldsymbol{s}_i) \neq 0$.

Remark 2. *It is often more desirable to work directly with an approximation of the inverse* $(\mathbf{B}_i)^{-1}$ *than with* $\mathbf{B}_i$. *The approximation of inverse matrix* $(\mathbf{B}_i)^{-1}$ *can be obtained from* (1.12) *or* (1.13) *using the Sherman-Morrison-Woodbury formula [Hag89].*

1.3.1 Limited-memory approach

The naive implementation of update formulas (1.12) and (1.13) gives rise to a dense Hessian approximation $\mathbf{B}_i$. This makes the use of secant methods prohibitive, especially in the context of large-scale problems. An efficient and memory friendly alternative

can be obtained using limited-memory secant methods. These methods consider memory $\{\mathbf{S}_i, \mathbf{Y}_i\}$ of $M \ll n$ secant pairs and use them to recursively evaluate Hessian approximation update rules. Employing update rules (1.12) or (1.13) gives rise to the limited-memory BFGS (L-BFGS) method or to the limited-memory SR1 (L-SR1) method, respectively. The memory $\{\mathbf{S}_i, \mathbf{Y}_i\}$ is usually filled in by collecting the curvature information over the M most recent iterations. Thus, $\{\mathbf{S}_i, \mathbf{Y}_i\}$ is constructed as

$$\mathbf{S}_i = [\mathbf{s}_{i-m}, \ldots, \mathbf{s}_{i-1}] \in \mathbb{R}^{n\times m} \qquad \text{and} \qquad \mathbf{Y}_i = [\mathbf{y}_{i-m}, \ldots, \mathbf{y}_{i-1}] \in \mathbb{R}^{n\times m},$$

where $m = i$, for all $i < M$, otherwise $m = M$. Some of the recent developments propose to construct memory $\{\mathbf{S}_i, \mathbf{Y}_i\}$ using information obtained only at the current iteration, for example by sampling the random search directions [BJT19, GG18, BHNS16, GGR16] or by filtering the secant pair using multilevel hierarchy [GMT12].

Using the L-BFGS or L-SR1 Hessian approximation schemes, the trust-region method can be implemented in a matrix-free spirit. The solution of the trust-region subproblem and an evaluation of the trust-region ratio (1.6) is then performed without forming the matrix $\mathbf{B}_i$ explicitly. In the case of the L-BFGS method, the matrix-products involving $\mathbf{B}_i$, or its inverse $(\mathbf{B}_i)^{-1}$, can be performed using the approach proposed in Reference [ML91] or two-loop recursion algorithm developed in Reference [Noc80], respectively. In the case of the L-SR1 method, the recursive algorithm provided in Reference [EM17] can be used to evaluate products involving both $\mathbf{B}_i$ and $(\mathbf{B}_i)^{-1}$.

The aforementioned methods for evaluating the products involving $\mathbf{B}_i$ or $(\mathbf{B}_i)^{-1}$ are well-suited for parallel processing environments. However, their design does not permit efficient implementation of optimization methods for large-scale constrained quadratic optimization problems [BLNZ95], as required for solving trust-region subproblems. To reduce the computational cost, Byrd et al. proposed a compact representation of limited-memory updates rules [BNS94]. The L-BFGS or the L-SR1 update rules can then be written in the following abstract form:

$$\mathbf{B}_{i+1} = \mathbf{B}_0 + \mathbf{\Psi}_i \mathbf{M}_i \mathbf{\Psi}^T, \tag{1.14}$$

where $\mathbf{M}_i \in \mathbb{R}^{M\times M}$ and $\mathbf{\Psi}_i \in \mathbb{R}^{n\times M}$. Here, the matrix $\mathbf{\Psi}_i$ is required to have a full rank. The matrix $\mathbf{B}_0 \in \mathbb{R}^{n\times n}$ is usually chosen as a scalar multiple of the identity matrix. The exact form of the matrices $\mathbf{\Psi}_i$ and $\mathbf{M}_i$ depends on the particular type of the Hessian approximation scheme. For example, in the case of the L-BFGS method, $\mathbf{\Psi}_i$ and $\mathbf{M}_i$ take on the following form:

$$\mathbf{\Psi}_i = \begin{bmatrix} \mathbf{B}_0\mathbf{S}_i & \mathbf{Y}_i \end{bmatrix} \qquad \text{and} \qquad \mathbf{M}_i = \begin{bmatrix} -\mathbf{S}_i^T\mathbf{B}_0\mathbf{S}_i & -\mathbf{L}_i \\ -\mathbf{L}_i^T & \mathbf{D}_i \end{bmatrix}^{-1}.$$

In the case of the L-SR1 method, $\mathbf{\Psi}_i$ and $\mathbf{M}_i$ are defined as

$$\mathbf{\Psi}_i = \mathbf{Y}_i - \mathbf{B}_0\mathbf{S}_i \qquad \text{and} \qquad \mathbf{M}_i = (\mathbf{D}_i + \mathbf{L}_i + \mathbf{L}_i^T - \mathbf{S}_i^T\mathbf{B}_0\mathbf{S}_i)^{-1}.$$

Here, the strictly lower diagonal matrix $\mathbf{L}_i$ and the diagonal matrix $\mathbf{D}_i$ are obtained by decomposing $\mathbf{S}_i^T\mathbf{Y}_i$ as $\mathbf{S}_i^T\mathbf{Y}_i = \mathbf{L}_i + \mathbf{D}_i + \mathbf{U}_i$, where $\mathbf{U}_i$ is a strictly upper triangular matrix.

Remark 3. *In the context of trust-region methods, it is important to update* $\mathbf{B}_i$*, even if the trial point is rejected. This is due to the fact, that a rejection might indicate that the current* $\mathbf{B}_i$ *is not an adequate approximation of the true Hessian* $\nabla^2 f(\boldsymbol{x}_i)$*.*

1.3.2 The solution of trust-region subproblem

In order to obtain an efficient matrix-free implementation of a quasi-Newton trust-region method, a careful design of the trust-region subproblem solver is required. Of course, many iterative methods mentioned in Section 1.2.1, such as the STCG method [CGT00], can be readily used. The only algorithmic modification is associated with replacing all matrix-vector products with an application of the approximate Hessian to the vector. Although this approach is simple, it is not desirable in practice as it makes quasi-Newton trust-region methods significantly more expensive than their line-search counterparts. In particular, line-search methods usually require only one application of the inverse Hessian approximation to a vector per nonlinear iteration, while obtaining the approximate solution of a single trust-region subproblem by an iterative method requires several (approximate) Hessian-vector products.

To decrease the computational cost, several specialized constrained quadratic minimization techniques were developed. The majority of them are designed for solving trust-region subproblems with a trust-region radius defined by the ℓ_2-norm. For example, Kaufman developed the double-Dogleg approach [Kau99], which utilizes the approximate Hessian and its inverse obtained by the L-BFGS method. Burke et al. [BWX08] proposed two variants of the Morè–Sorensen method, which take advantage of the L-BFGS scheme. Furthermore, an efficient implementation of the Morè-Sorensen method based on the unrolling formula for the L-BFGS method was designed in References [EM12, EM14]. Finally, the orthonormal basis (OBS) method, which utilizes the compact representation of quasi-Newton updates was proposed in References [BEM17, BGZY17]. The advantage of the OBS method is that it explores the structure of generic quasi-Newton update (1.14) and, therefore, it is suitable for both the L-BFGS and the L-SR1 Hessian approximation scheme.

In the context of the trust-region subproblems with point-wise constraints, a variant of the infeasible interior point method [SS97] was developed and utilized in Reference [XB07] and Reference [EK15], respectively. The method also utilizes a compact representation of quasi-Newton updates in order to efficiently solve arising trust-region subproblems.

1.3.3 Implementation and numerical considerations

A practical implementation of quasi-Newton trust-region methods requires several considerations. In particular, we need to find suitable matrix $\mathbf{B}_0$ required for the evaluation of (1.14) and ensure the numerical stability of the quasi-Newton updates.

Choice of initial matrix $\mathbf{B}_0$

The choice of a matrix $\mathbf{B}_0$ is crucial for obtaining well-scaled search directions. In practice, the matrix $\mathbf{B}_0$ is often selected as a multiple of the identity matrix, e.g., $\mathbf{B}_0 = \alpha_i \mathbf{I}$, where $\alpha_i \in \mathbb{R}^+$. For the L-BFGS method, it is conventional to choose α_i as

$$\alpha_i = \max\left[\frac{\mathbf{y}_i^T \mathbf{y}_i}{\boldsymbol{s}_i^T \mathbf{y}_i}, \epsilon_1\right]. \tag{1.15}$$

The clamping parameter $\epsilon_1 > 0$ in (1.15) prevents the Hessian approximations $\mathbf{B}_i$ from becoming nearly singular [BBEM19]. The value α_i can be alternatively obtained by solving an eigenvalue problem [RM18].

The choice of the suitable $\mathbf{B}_0$ for the L-SR1 method was recently investigated by Erway et al. [EGMO20], who pointed out that certain choices of $\mathbf{B}_0$ are responsible for almost all numerical instabilities related to the L-SR1 method. To overcome the difficulty, they proposed to initialize $\mathbf{B}_0$ as in (1.15), where α_i is obtained by solving the generalized eigenvalue problem.

Stable quasi-Newton updates

The L-BFGS method produces a positive-definite Hessian approximation $\mathbf{B}_i$ only if the curvature condition $\langle \mathbf{y}_i, \boldsymbol{s}_i \rangle > 0$ is satisfied. In practice, we can ensure that the curvature condition holds by incorporating a line-search strategy with Wolfe conditions [NW06]. Alternatively, we can skip updates by discarding all secant pairs $\{\boldsymbol{s}_i, \mathbf{y}_i\}$ for which the curvature condition becomes very small, i.e., if $\langle \boldsymbol{s}_i, \mathbf{y}_i \rangle < \epsilon_2$, where $\epsilon_2 \in (0, 1)$. However, skipping updates too often might degrade the performance of the overall method. An alternative strategy, based on the damped L-BFGS method [NW06, Pow85] can be applied instead.

The L-SR1 scheme becomes numerically unstable when the denominator of (1.13) becomes too small. As a remedy, we can skip updates if the following condition is satisfied:

$$|\langle \mathbf{y}_i - \mathbf{B}_i \boldsymbol{s}_i, \boldsymbol{s}_i \rangle| < \epsilon_3 \|\mathbf{y}_i - \mathbf{B}_i \boldsymbol{s}_i\| \|\boldsymbol{s}_i\|, \tag{1.16}$$

where $\epsilon_3 \in (0, 1)$. Interestingly, the condition (1.16) is often fulfilled when the curvature represented by $\mathbf{B}_i$ along $\boldsymbol{s}_i$ is already correct [CGT91].

In the end, we note that it is important to ensure that all algebra operations performed using a compact matrix representation of quasi-Newton updates are stable. To this aim, we recursively remove the first column of $\mathbf{Y}$ and $\mathbf{S}$ if

$$\boldsymbol{\Psi}^T \boldsymbol{\Psi} < 0,$$

or if a matrix $\mathbf{M}$ is ill-conditioned.

1.4 Conclusion

In this chapter, we discussed the basic trust-region method. We provided the algorithmic description of the method and discussed how to solve the arising trust-region subproblems. Furthermore, quasi-Newton trust-region methods were considered. Particular focus was given to limited-memory secant methods, i.e., L-BFGS and L-SR1, and their practical implementation.

Chapter 2

Recursive multilevel trust-region method

2.1 Introduction

Partial differential equations (PDEs) are ubiquitous in many scientific fields. Their discretization often leads to large-scale systems of equations. Solving such large-scale systems using direct methods, e.g., LU or Cholesky factorization is often infeasible due to time and memory constraints. For example, modern sparse direct solvers can perform factorization of a linear system with n unknowns using $\mathcal{O}(n^2)$ flops for three-dimensional problems [Saa03]. To ease the computational burden, iterative methods are often employed in practice. Unfortunately, the convergence rate of iterative methods often deteriorates with increasing problem size, i.e., the number of iterations required to reach the desired tolerance grows rapidly with the number of unknowns.

Multilevel methods are known to be optimal solvers for many problems, as their convergence rate is often independent of the problem size, e.g., for elliptic problems, and the number of required arithmetic operations grows proportionally with the number of unknowns. The main idea behind multilevel methods is to employ a hierarchy of auxiliary problems, which are then solved internally in order to speed up the solution process. A hierarchy of problems can be obtained naturally for many practical applications. For example, when solving discretized partial differential equations, the hierarchy of problems can be generated using meshes with different resolutions.

Multigrid methods were originally developed for solving linear elliptic boundary value problems. The origin of linear multilevel methods, often called multigrid, can be traced back to Fedorenko [Fed62, Fed64], and Bakhvalov [Bak66]. However, the method became popular a decade later, after the work of Brandt [Bra73, Bra76, Bra77] and Hackbusch [Hac76] demonstrated its actual potential and efficiency. Since then, the multigrid method has been extended and applied to many different problems, ranging from simple Poisson equations to problems in quantum chromodynamics [BMMR87].

Detailed information about the historical development of the linear multigrid method can be found in Reference [Bra94]. For a general introduction to multigrid methods, we refer the reader to the following monographs [BM+00, Hac85, TOS00, Wes95, BZ00].

Our aim is to solve non-convex optimization problems. The easiest approach for utilizing a multilevel method within a nonlinear framework is to employ an outer linearization scheme, such as Newton's method. The outer scheme then produces a sequence of linear systems of equations, which can be solved using a linear multigrid method. Due to its simplicity, this approach is extensively used in practice [BR82, DW98, DMS00, BHH06]. We remark that the global convergence of Newton's method is not guaranteed for generic nonlinear problems unless a suitable initial guess is chosen within a local neighborhood of a solution. As a consequence, the globalized variants of Newton's method, such as line-search [NW06] or trust-region [CGT00] methods; have to be employed in practice. Additionally, the arising linear systems of equations may not be positive definite, which can cause problems for standard linear multigrid to converge.

Instead of utilizing the multigrid method for arising linearized systems, the nonlinear multilevel methods address nonlinearity directly. Similarly to the multigrid, a hierarchy of auxiliary nonlinear problems is used to speed up a solution process of the original nonlinear problem. Historically, the first nonlinear multilevel method, called full approximation scheme (FAS), is developed by Brandt in Reference [Bra77]. The generalization of FAS, called nonlinear multigrid (NMG), was proposed subsequently by Hackbusch [Hac85]. Both methods, FAS and NMG, are designed to solve nonlinear systems of equations arising from the discretization of PDEs. In the context of nonlinear optimization problems considered in this work, this corresponds to applying FAS or NMG directly to the first-order criticality conditions, while neglecting information concerning the objective function. However, without knowing the behavior of the objective function, it is difficult to control the convergence behavior of a solution strategy. Indeed, FAS and NMG are generally not guaranteed to converge globally for nonlinear problems [Hac85, Reu87, Reu88].

In the interest of obtaining a globally convergent multilevel method, the MG/OPT method was introduced by Nash [Nas00]. The MG/OPT method is designed to solve unconstrained convex optimization problems and relies on two key aspects. Firstly, the first-order consistency between the fine and coarse-level optimization problems is imposed in the neighborhood of the current fine-level iterate. This ensures that a coarse-level correction is a direction of descent with respect to a fine level. Secondly, a line-search strategy is employed to guarantee global convergence. The MG/OPT method is designed from an optimization point of view, and it does not rely on the assumption that a multilevel hierarchy is obtained by discretizing the PDEs.

Since the introduction of MG/OPT, combining multilevel methods with line-search globalization strategies has become popular. Wen and Goldfarb propose a variant of the MG/OPT method that is also suitable for solving non-convex optimization problems [WG10]. A multilevel line-search method developed for solving convex infinite-

dimensional problems is introduced by Borzì and Schultz [BS09]. Frandi and Papini develop a coordinate search multilevel line-search method as an alternative to the derivative-free optimization techniques [FP14]. Multilevel line-search methods suitable for optimization of non-smooth composite functions are proposed in References [PLRR14, HPZ16, Par17]. The FAS-like methods united with line-search globalization strategy are studied using the framework of subspace correction methods in References [TX98, TX02, CHW19].

Instead of using a line-search method, Gratton, Sartenaer, and Toint propose to employ a trust-region globalization strategy within the nonlinear multilevel framework [GST08]. The resulting method, called recursive multilevel trust-region (RMTR), is also designed from the optimization point of view and it does not rely on the assumption that a multilevel hierarchy is obtained by discretizing the PDEs. Similarly to the MG/OPT method, the first-order consistency between the fine and coarse-level optimization problems is required. Contrary to the MG/OPT method, the size of coarse-level corrections is regulated by a fine-level trust-region radius and coarse-level corrections must obey quality control before being accepted by the fine level. A comprehensive analysis, which shows a convergence of the RMTR method to the first-order critical points even for non-convex objective functions, is carried out in Reference [GST08]. Note that the proposed RMTR method relies on the assumption that the trust-region is defined by the ℓ_2-norm.

The RMTR method was later extended to solve optimization problems with point-wise constraints [GMTW08] . This particular variant of the RMTR method utilizes a trust-region defined by the ℓ_∞-norm. In this way, the intersection between trust-region and variable bounds can be performed easily, in a component-wise manner. Using the ℓ_∞-norm to define the trust-region also allows for a more accurate representation of the fine-level trust-region radius on coarser levels. Indeed, a numerical study reported in Reference [GMS+10] indicates that employing the ℓ_∞-norm instead of the ℓ_2-norm gives rise to lower convergence rates of the RMTR method when applied to unconstrained problems.

Both aforementioned variants of the RMTR method were further extended by several authors. For example, Gross and Krause broaden the convergence theory using weaker assumptions on the differentiability of the objective function [GK09b, GK09c]. Karasözen expand the method to the derivative-free framework [Kar15]. Ulbrich and Ziems couple the RMTR method with the adaptive mesh refinement framework [ZU11, UZ17]. Kragel develops a combination of the RMTR method with proper orthogonal decomposition (POD) methods [Kra05]. The additive variant of the RMTR method, termed as additive preconditioned trust-region method (APTS), is proposed by Gross and Krause [GK09a]. The APTS method utilizes the domain decomposition framework, which makes it particularly well-suited for high-performance computing environments.

By design, the RMTR method does not pose restrictions on the way in which the multilevel hierarchy is created. We can construct a multilevel hierarchy by discretizing the same partial differential equations using meshes with different resolutions. Alternatively,

we can employ some low-fidelity approximations of the original optimization problem. The latter is conceptually similar to trust-region methods developed for tackling multiple fidelities, such as the first-order approximation and model management optimization (AMMO) method [Ale96, Lew96, AL01], or the trust-region proper orthogonal decomposition (TRPOD) [Fah00] method. Interestingly, these methods also ensure convergence to first-order critical points of the high-fidelity objective function by enforcing first-order consistency between the high and low-fidelity objective functions.

2.2 The recursive multilevel trust-region method

The RMTR method is designed for solving large-scale constrained non-convex optimization problems. In this chapter, we introduce the nonlinear multilevel framework and provide an algorithmic description of the RMTR method. For the additional details and the convergence theory of the RMTR method, we refer the interested reader to References [GST08, GMTW08, GK09b]. We also point out, that the discussion provided in this chapter serves as a basis for the development of novel variants of the RMTR methods presented in Part II.

2.2.1 Nonlinear multilevel framework

Multilevel solution strategies take into account a hierarchy of levels. Each level $l = 1, \dots, L$, is associated with three transfer operators, a level-dependent objective function, and a the level-dependent feasible set.

Transfer operators

The transfer of the data between subsequent levels of the multilevel hierarchy is ensured by the following transfer operators:

- The prolongation operator $\mathbf{I}_l^{l+1} : \mathbb{R}^{n^l} \to \mathbb{R}^{n^{l+1}}$ is designed to transfer primal variables, such as the corrections, from level l to level $l+1$.
- The restriction operator $\mathbf{R}_{l+1}^l : \mathbb{R}^{n^{l+1}} \to \mathbb{R}^{n^l}$ transfers dual variables, e.g., gradients, from level $l+1$ to level l. In this work, the operator $\mathbf{R}_{l+1}^l$ is defined as the adjoint of $\mathbf{I}_l^{l+1}$, i.e., $\mathbf{R}_{l+1}^l := (\mathbf{I}_l^{l+1})^T$.
- The projection operator $\mathbf{P}_{l+1}^l : \mathbb{R}^{n^{l+1}} \to \mathbb{R}^{n^l}$ transfers primal variables, e.g., the current iterate, from level $l+1$ to level l.

Level-dependent objective functions

Each level l is associated with a level-dependent objective function $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$. Throughout this work, we assume that for each $l = 1, \dots, L-1$, a function h^l is computationally less expensive to minimize than h^{l+1} and that $n^{l+1} \geq n^l$. In addition, we assume

that $h^L := f$. We aim to construct the level-dependent objective function h^l, such that its minimization provides a good search direction on the level $l+1$.

The construction of level-dependent objective functions $\{h^l\}_{l=1}^L$ relies on the knowledge of some auxiliary objective functions $\{f^l\}_{l=1}$, where $f^L := h^L := f$. We refer to the functions $\{f^l\}_{l=1}^{L-1}$ as low-cost functions and assume that each f^l approximates f, but that it is computationally cheaper to minimize than f. There are several approaches to acquire a set $\{f^l\}_{l=1}^L$. In this work, we consider two following approaches:

- **Multi-resolution approach:** A hierarchy of low-cost functions $\{f^l\}_{l=1}^{L-1}$ is constructed using less accurate discretizations of the same infinite dimensional problem. This approach is commonly used in various scientific fields, often, where differential equations are used to describe a certain physical phenomenon.
- **Multi-fidelity approach:** A hierarchy of low-cost functions $\{f^l\}_{l=1}^{L-1}$ is constructed using structurally cheaper approximations of the objective function f. The multi-fidelity approach is commonly used in the field of fluid dynamics [AL01], or cardiac electrophysiology [QPK19].

We note that the multi-resolution and the multi-fidelity approaches can also be combined to improve the overall efficiency of a multilevel method.

Level-dependent feasible set

We associate each level l with a level-dependent feasible set $\mathcal{F}^l$. On the finest level, we assume that $\mathcal{F}^L := \mathcal{F}$. On all levels $l < L$, the level-dependent feasible set $\mathcal{F}^l$ is defined as

$$\mathcal{F}^l := \{\boldsymbol{x}^l \in \mathbb{R}^{n^l} \mid \mathbf{l}^l \leq \boldsymbol{x}^l \leq \mathbf{u}^l\}, \tag{2.1}$$

where $\mathbf{l}^l, \mathbf{u}^l \in (\mathbb{R} \cup \{-\infty\} \cup \{\infty\})^{n^l}$. As we will see in Section 2.5, the role of the feasible set $\mathcal{F}^l$ is two-fold. On one hand, it ensures that the sequence of iterates produced by the RMTR method remains feasible with respect to the variable bounds, which are defined by the feasible set $\mathcal{F}$. On the other hand, $\mathcal{F}^l$ also imposes a restriction on the size of all corrections taken on a given level l.

We discuss how to construct transfer operators, level-dependent objective functions and level-dependent feasible sets in detail in Sections 2.4, 2.3, and 2.5, respectively. Additionally, Part II provides details on how to construct level-dependent quantities for three specific applications from the field of fracture mechanics, cloth animation, and machine learning.

2.2.2 The RMTR algorithm

We consider RMTR method in the form of a V-cycle, as shown in Figure 2.1. We use subscripts to denote an iteration number and superscripts to denote a level. For instance, the symbol $\boldsymbol{x}_i^l$ denotes the solution vector on level l at iteration i.

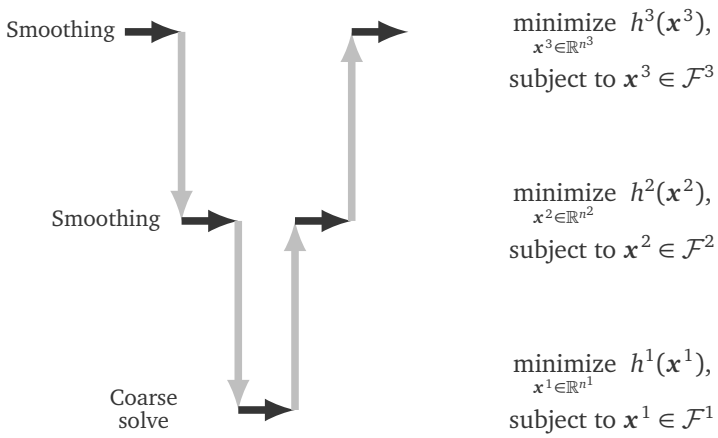

Figure 2.1: Example of V-cycle with three levels.

The V-cycle starts on the finest level, $l = L$, with the initial iterate $\boldsymbol{x}_0^L$, and passes through all levels of the multilevel hierarchy until the coarsest level, i.e., $l = 1$, is reached. On each level, the algorithm performs a pre-smoothing step in order to improve the iterate $\boldsymbol{x}_0^l$. This is done by minimizing the level-dependent objective function h^l as follows:

$$\begin{aligned} &\underset{\boldsymbol{s}^l \in \mathbb{R}^{n^l}}{\text{minimize}} \quad h^l(\boldsymbol{x}_0^l + \boldsymbol{s}^l), \\ &\text{subject to} \quad \boldsymbol{x}_0^l + \boldsymbol{s}^l \in \mathcal{F}^l, \end{aligned} \tag{2.2}$$

where $\mathcal{F}^l$ is level-dependent feasible set, as defined in Section 2.2.1. The minimization on a given level is performed approximately by employing μ_1 iterations of the trust-region method, Algorithm 1. The updated iterate $\boldsymbol{x}_{\mu_1}^l := \boldsymbol{x}_0^l + \boldsymbol{s}^l$ is then used to initialize the solution vector on the subsequent coarser level, thus $\boldsymbol{x}_0^{l-1} := \mathbf{P}_l^{l-1}\boldsymbol{x}_{\mu_1}^l$. In addition, we initialize trust-region radius on level $l-1$ as $\Delta_0^{l-1} = \Delta_{\mu_1}^l$. After Δ_0^{l-1} and $\boldsymbol{x}_0^{l-1}$ are initialized, we can construct level-dependent objective function h^{l-1} and feasible set $\mathcal{F}^{l-1}$ using techniques described in Section 2.4, and 2.5, respectively. This process is repeated recursively until the coarsest level, $l = 1$, is reached.

Once the coarsest level is reached, we again approximately minimize (2.2). The minimization is carried out using μ^1 steps of the trust-region method and produces an updated coarse-level iterate $\boldsymbol{x}_{\mu^1}^1$. We note that the minimization of (2.2) on the coarsest level is usually carried out more accurately than on all other levels, by performing more steps of the trust-region method, or by solving the trust-region subproblems more accurately.

After obtaining the updated iterate $\boldsymbol{x}_{\mu^1}^1$, the RMTR algorithm initiates the upward phase of the V-cycle. An upward phase is associated with the return to the finest level of the multilevel hierarchy while passing through all intermediate levels. Starting on the coarsest level, we compute the coarse-level correction $\boldsymbol{s}_*^{l-1}$ as the difference between the

initial iterate $\boldsymbol{x}_0^{l-1}$, and the final iterate $\boldsymbol{x}_*^{l-1}$, i.e., $\boldsymbol{s}_*^{l-1} = \boldsymbol{x}_*^{l-1} - \boldsymbol{x}_0^{l-1}$. This correction is then prolongated to the subsequent finer level as

$$\boldsymbol{s}_{\mu_1+1}^l := \mathbf{I}_{l-1}^l(\boldsymbol{s}_*^{l-1}) := \mathbf{I}_{l-1}^l(\boldsymbol{x}_*^{l-1} - \boldsymbol{x}_0^{l-1}).$$

As it is common for trust-region methods, the prolongated coarse-level correction $\boldsymbol{s}_{\mu_1+1}^l$, can not be accepted immediately, but only if it provides a decrease in the fine-level objective function; thus only if

$$h^l(\boldsymbol{x}_{\mu_1} + \boldsymbol{s}_{\mu_1+1}^l) < h^l(\boldsymbol{x}_{\mu_1}).$$

For this reason, we define the multilevel trust-region ratio $\rho_{\mu_1+1}^l$ as

$$\rho_{\mu_1+1}^l := \frac{h^l(\boldsymbol{x}_{\mu_1}^l) - h^l(\boldsymbol{x}_{\mu_1}^l + \boldsymbol{s}_{\mu_1+1}^l)}{h^{l-1}(\boldsymbol{x}_0^{l-1}) - h^{l-1}(\boldsymbol{x}_*^{l-1})} = \frac{\text{fine-level reduction}}{\text{coarse-level reduction}}. \tag{2.3}$$

A positive value of $\rho_{\mu_1+1}^l$ implies a decrease in the fine-level objective function h^l. Therefore, it is safe to accept correction $\boldsymbol{s}_{\mu_1+1}^l$ and enlarge the trust-region radius. In contrast, small or negative values of $\rho_{\mu_1+1}^l$ suggest that there is no good agreement between the fine and the coarse-level models. Therefore, the prolongated coarse-level correction $\boldsymbol{s}_{\mu_1+1}^l$ has to be rejected and the trust-region radius must be reduced in size. To this end, the RMTR algorithm performs μ_2 post-smoothing steps in order to improve the current iterate on a given level l. This process is again repeated on every level of the multilevel hierarchy until we reach the finest level L.

Algorithm 4 summarizes the outlined process. Please note that Algorithm 4 makes use of the algorithms presented in Section 1.2.

2.2.3 Nonlinear smoothing and coarse-level solve

On each level l, the RMTR method requires an approximate solution of the level-dependent minimization problem (2.2). To this aim, we employ the trust-region method, Algorithm 1. As explained in Section 1.2, at each iteration i, the trust-region method obtains a correction $\boldsymbol{s}_i^l$ by minimizing the model m_i^l, given as

$$\begin{aligned} &\underset{\boldsymbol{s}_i^l \in \mathbb{R}^{n^l}}{\text{minimize}} \quad m_i^l(\boldsymbol{x}_i^l) := h^l(\boldsymbol{x}_i^l) + \langle \nabla h^l(\boldsymbol{x}_i^l),\ \boldsymbol{s}_i^l \rangle + \frac{1}{2}\langle \boldsymbol{s}_i^l, \mathbf{B}_i{}^l \boldsymbol{s}_i^l \rangle, \\ &\text{subject to} \quad \boldsymbol{x}_i^l + \boldsymbol{s}_i^l \in \mathcal{W}_i^l, \end{aligned} \tag{2.4}$$

where $\mathbf{B}_i{}^l$ denotes the Hessian obtained on level l at iteration i. A symbol $\mathcal{W}_i^l$ denotes a working set, defined as an intersection of level-dependent feasible set $\mathcal{F}^l$ with the current trust-region radius $\mathcal{B}_i^l := \{\boldsymbol{x}_i^l + \boldsymbol{s}_i^l \in \mathbb{R}^{n^l} \mid \|\boldsymbol{s}_i^l\|_p \leq \Delta_i^l\}$, e.g., $\mathcal{W}_i^l := \mathcal{F}^l \cap \mathcal{B}_i^l$.

Algorithm 4 RMTR(l, h^l, $\boldsymbol{x}_0^l$, $\mathcal{F}^l$, Δ_0^l)

Require: $l \in \mathbb{N}$, $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$, $\boldsymbol{x}_0^l \in \mathbb{R}^{n^l}$, $\mathcal{F}^l$, $\Delta_0^l \in \mathbb{R}$
Constants: μ_1, μ_2, $\mu^1 \in \mathbb{N}$

```
[x^l_{μ1}, Δ^l_{μ1}] = Basic_TR(h^l, x^l_0, F^l, Δ^l_0, μ1)        ▷ Pre-smooth/ Call Algorithm 1
Construct F^{l-1} and h^{l-1}        ▷ Initialize coarse-level feasible set and objective function

if l == 2 then
    [x^l_*, __] = Basic_TR(h^{l-1}, P^{l-1}_l x^l_{μ1}, F^{l-1}, Δ^l_{μ1}, μ^1)        ▷ Solve coarse-level problem
else
    [x^{l-1}_*, __]=RMTR(l-1, h^{l-1}, P^{l-1}_l x^l_{μ1}, F^{l-1}, Δ^l_{μ1})        ▷ Call RMTR recursively
end if

s^l_{μ1+1} = I^l_{l-1}(x^{l-1}_{μ*} - P^{l-1}_l x^l_{μ1})        ▷ Prolongate coarse-level correction

ρ^l_{μ1+1} = (h^l(x^l_{μ1}) - h^l(x^l_{μ1} + s^l_{μ1+1})) / (h^{l-1}(P^{l-1}_l x^l_{μ1}) - h^{l-1}(x^{l-1}_*))        ▷ Evaluate multilevel TR ratio

[x^l_{μ1+1}, Δ^l_{μ1+1}] = Convergence_control(ρ^l_{μ1+1}, x^l_{μ1}, s^l_{μ1+1}, Δ^l_{μ1})        ▷ Call Algorithm 2

[x^l_*, Δ^l_*] = Basic_TR(h^l, x^l_{μ1+1}, F^l, Δ^l_{μ1+1}, μ2)        ▷ Post-smooth/ Call Algorithm 1
return x^l_*, Δ^l_*
```

Solution of the trust-region subproblem

The trust-region subproblems can be solved approximately, as long as the obtained minimizer of (2.4) satisfies the sufficient decrease condition (SDC), see Section 1.2.1. The choice of trust-region subproblem solver usually varies on different levels, especially in the case of the multi-resolution variant of the RMTR method. This is motivated by two complementary observations known from the linear multigrid. On one hand, some iterative schemes, such as Jacobi or Gauss-Seidel method, are efficient in reducing high-frequency/oscillatory components of the error, while they are ineffective in eliminating the low-frequency/smooth components of the error. We refer to these types of iterative methods as smoothers. On the other hand, smooth functions on a fine level may appear oscillatory on coarser levels. As a consequence, we can reduce different components of the error on the different levels, by employing only a few steps of smoothing. The remaining components of the error are subsequently reduced by solving coarse-level problems more accurately. This is computationally feasible, as the number of unknowns associated with the coarsest level is usually low.

In the context of the RMTR method, considered here, the algorithmic choice of the smoother and the coarsest-level subproblem solver is not as trivial as in the case of standard linear multigrid methods. This is due to the fact, that the arising Hessian matrices might not be positive definite. Furthermore, we have to ensure that the constraint in (2.4) is not violated and that the (SDC) condition is satisfied.

Numerical evidence indicates that variants of the successive-coordinate minimization (SCM) method constitute good smoothers [BM+00, GMS+10, GST06]. However,

the SCM methods are inherently of serial nature and can not be parallelized easily. We can implement the SCM method in parallel by embedding it into a domain-decomposition framework. This gives rise to a hybrid additive-multiplicative solution strategy [BFKY11]. Unfortunately, the effectiveness of the hybrid SCM method deteriorates with an increasing number of processors [ABHT03], which has also a negative impact on the overall convergence of the RMTR method [ZKN+20]. We can improve the smoothing properties of the hybrid SCM method by adding an extra term to the diagonal of the Hessian [BFKY11]. This extra term takes into account the off-processor components of the Hessian and increases the diagonal dominance of the matrix. However, to the best of our knowledge, a design of a parallel SCM method, that is stable with respect to a number of processors, is still the subject of research. In contrast to the SCM method, Krylov subspace methods are well-suited for parallel processing environments. Hence, it is often desirable to employ Conjugate Gradient-based methods as smoothers, e.g., the STCG method [Ste83, Toi81] for trust-region subproblems in the ℓ_2-norm or the PTCG method [Yua00] for trust-region subproblems in the ℓ_∞-norm.

On the coarsest level, $l = 1$, we aim to solve trust-region subproblems more accurately, for instance by Morè-Sorensen method, if the trust-region is defined by the ℓ_2-norm, or by the interior-point method, if the trust-region is defined by the ℓ_∞-norm. Here, we point out that the trust-region subproblem solvers employed on the coarsest level do not have to be specially designed for parallel processing environments. This is due to the fact that they are usually executed in a sequential manner, i.e., using a single processor, in order to reduce the communication cost.

2.3 Multilevel hierarchy and transfer operators

Transfer of data between subsequent levels of a multilevel hierarchy is realized by three transfer operators. In this section, we review the necessary and desirable properties of all three transfer operators. Furthermore, we demonstrate how to construct transfer operators in practice by means of two examples. The first example is related to the multi-resolution variant of the RMTR method, while the second example considers the multi-fidelity variant of the RMTR method.

2.3.1 Properties of transfer operators

Prolongation and restriction operators

As it is standard in multilevel methods, the prolongation operators $\{\mathbf{I}_l^{l+1}\}_{l=1}^{L-1}$ are assumed to have full column rank. The restriction operators $\{\mathbf{R}_{l+1}^{l}\}_{l=1}^{L-1}$ are assumed to satisfy the following relationship $\mathbf{R}_{l+1}^{l} := \sigma(\mathbf{I}_l^{l+1})^T$, where $\sigma > 0$. For simplicity, we adopt $\sigma = 1$ throughout this thesis and note that different values of σ require more careful rescaling of certain multilevel quantities, such as multilevel trust-region ratio, defined by (2.3).

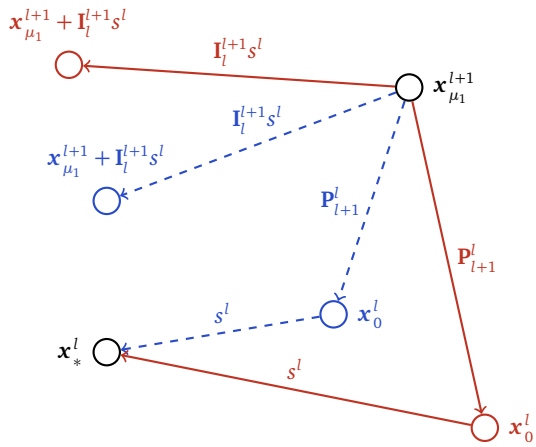

Figure 2.2: Impact of the initial coarse-level iterate $\boldsymbol{x}_0^l$ on the resulting coarse-level correction $\boldsymbol{s}^l$. Different choices of projection operator $\mathbf{P}_{l+1}^l$ give rise to various $\boldsymbol{x}_0^l$. Even if the coarse level converges to the same iterate $\boldsymbol{x}_*^l$, the coarse-level corrections $\boldsymbol{s}^l$ differ. Consequently, assigning $\mathbf{I}_l^{l+1}\boldsymbol{s}^l$ to the current fine-level iterate $\boldsymbol{x}_{\mu_1}^{l+1}$ produces distinct fine-level trial points $\boldsymbol{x}_{\mu_1}^{l+1} + \mathbf{I}_l^{l+1} s^l$.

It is often beneficial to create prolongation operators, such that

$$\sum_{k=1,\dots,n^l} (\mathbf{I}_l^{l+1})_{jk} = 1, \quad \forall j = 1,\dots,n^{l+1}, \tag{2.5}$$

for all $l = 1,\dots,L-1$. Thus, we enforce the row sum of $\mathbf{I}_l^{l+1}$ to be equal to one, which guarantees that constant functions are preserved by the transfer.

Projection operator

The RMTR method employs a projection operator in order to construct an initial guess on a level l, given the iterate $\boldsymbol{x}_{\mu_1}^{l+1}$ on level $l+1$. It has been shown that the choice of projection operator impacts the convergence behavior of the RMTR method [GK09b, Gro09]. This is due to the fact that a quantity which gets transferred to the fine level is a coarse-level correction, defined as

$$\boldsymbol{s}^l = \boldsymbol{x}_*^l - \boldsymbol{x}_0^l.$$

The definition of $\boldsymbol{s}^l$ depends on the initial guess $\boldsymbol{x}_0^l := \mathbf{P}_{l+1}^l \boldsymbol{x}_{\mu_1}^{l+1}$, obtained by applying a projection operator $\mathbf{P}_{l+1}^l$. Thus, the usage of different projection operators leads to different coarse-level corrections and, therefore, to different fine-level trial points. As a consequence, employing the projection operator with poor approximation properties might slow down the overall convergence of the multilevel method. A graphical representation of this phenomenon is illustrated in Figure 2.2.

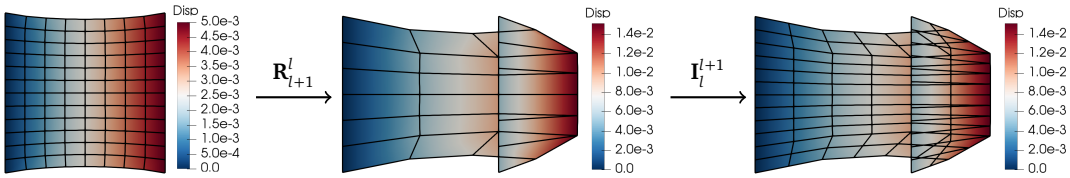

Figure 2.3: Example from continuum mechanics; a linear elastic material model discretized with the first-order Lagrange finite elements. The left side of boundary is held fixed in both components, while we prescribe $\text{Disp}_y = 5 \times 10^3$ on the right side. *Left:* The fine-level solution (displacement field). *Middle:* The coarse-level solution obtained by applying a restriction operator $\mathbf{R}^l_{l+1}$. *Right:* Prolongation of a coarse-level displacement to the fine level.

The majority of the literature on nonlinear multilevel methods employs the restriction operator $\mathbf{R}^l_{l+1}$ in order to transfer the iterates to the coarser levels. This is not ideal, because the restriction operator is a dual operator [Hac85], designed to transfer the dual quantities, such as gradients. However, we note that certain scaled variants of the restriction operator are reported to work well in practice [GST08].

Ideally, we search for a projection operator $\mathbf{P}^l_{l+1} \in \mathbb{R}^{n^l \times n^{l+1}}$, which satisfies the following relation:

$$\boldsymbol{x}^l = \mathbf{P}^l_{l+1} \underbrace{(\mathbf{I}^{l+1}_l \boldsymbol{x}^l)}_{\boldsymbol{x}^{l+1}}. \tag{2.6}$$

The restriction operator, $\mathbf{R}^l_{l+1} := (\mathbf{I}^{l+1}_l)^T$, does not typically satisfy the condition (2.6), since it tends to add artificial values to the transferred vector, as demonstrated by Figure 2.3. An operator $\mathbf{P}^l_{l+1}$ that meets requirement (2.6) can be found by solving the following least-square minimization problem:

$$\boldsymbol{x}^l = \underset{\boldsymbol{x}^l}{\text{minimize}} \; \|\boldsymbol{x}^{l+1} - \mathbf{I}^{l+1}_l \boldsymbol{x}^l\|^2. \tag{2.7}$$

This gives rise to

$$\boldsymbol{x}^l = ((\mathbf{I}^{l+1}_l)^T \mathbf{I}^{l+1}_l)^{-1} (\mathbf{I}^{l+1}_l)^T \boldsymbol{x}^{l+1}, \tag{2.8}$$

thus the unique operator $\mathbf{P}^l_{l+1}$ is obtained as

$$\mathbf{P}^l_{l+1} = ((\mathbf{I}^{l+1}_l)^T \mathbf{I}^{l+1}_l)^{-1} (\mathbf{I}^{l+1}_l)^T, \tag{2.9}$$

i.e., as the Moore–Penrose pseudo-inverse of $\mathbf{I}^{l+1}_l$. Unfortunately, the expression (2.8) requires the solution of a linear system, which makes the algorithm computationally expensive. Moreover, the operator $\mathbf{P}^l_{l+1}$ is a dense matrix, even though the matrix $\mathbf{I}^{l+1}_l$ is sparse.

2.3.2 Construction of transfer operators

Below, we demonstrate how to construct the transfer operators for the multi-resolution and the multi-fidelity variants of the RMTR method. The example presented for the multi-resolution variant considers a scenario, where the first-order finite element method is used to discretize some infinite-dimensional problem. The example presented for the multi-fidelity variant comes from the field of machine learning.

Multi-resolution approach

Let us assume that the finite element method (FEM) is used to discretize a PDE defined on the domain $\Omega \subset \mathbb{R}^d$, where $d \in \mathbb{N}$. The discretized problem is posed in the finite-dimensional space $\mathcal{X}$. The solution vector $\boldsymbol{x} \in \mathbb{R}^n$ of the discrete problem contains the coefficients of finite element functions. The multilevel hierarchy utilized by the RMTR method is based on the hierarchy of nested finite element spaces

$$\mathcal{X}^1 \subset \mathcal{X}^2 \subset \ldots \subset \mathcal{X}^L.$$

These FEM spaces are usually constructed using a hierarchy of nested meshes $\{\mathcal{T}\}_{l=1}^{L}$, obtained by uniformly refining the initial coarse-level mesh $\mathcal{T}^1$. Each FEM space $\mathcal{X}^l$ is spanned by the set of basis functions, denoted by $\{N_p\}_{p=1}^{n^l}$. Furthermore, we assume a coordinate isomorphism

$$\Phi^l : \mathbb{R}^{n^l} \to \mathcal{X}^l, \qquad \Phi^l(\boldsymbol{x}) := \sum_{p=1}^{n^l} x_p N_p^l.$$

Remark 4. *Approaches for constructing a hierarchy of nested FEM spaces based on non-nested meshes can be found in Reference [Dic10].*

Prolongation and restriction operators The prolongation operator between two subsequent levels of multilevel hierarchy is obtained as

$$\mathbf{I}_l^{l+1}\boldsymbol{x}^l = (\Phi^{l+1})^{-1}(\tilde{I}_l^{l+1}\Phi^l(\boldsymbol{x}^l)), \quad \text{for all } \boldsymbol{x}^l \in \mathbb{R}^{n^l}, \tag{2.10}$$

where $\tilde{I}_l^{l+1} : \mathcal{X}^l \to \mathcal{X}^{l+1}$ is natural embedding of $\mathcal{X}^l$ into $\mathcal{X}^{l+1}$. In other words, the non-zero entries of the matrix $\mathbf{I}_l^{l+1}$ are obtained as the coefficients of the unique linear combination representing the basis of $\mathcal{X}^l$ with respect to the basis of $\mathcal{X}^{l+1}$. As stated previously, the restriction operator $\mathbf{R}_{l+1}^l$ is simply chosen as $\mathbf{R}_{l+1}^l := (\mathbf{I}_l^{l+1})^T$.

Projection operator Although we can evaluate the projection operator $\mathbf{P}_{l+1}^l$ directly using formula (2.9), we prefer to use a computationally cheaper alternative. To this aim, we impose the identity relation (2.6) as

$$\Phi^l\boldsymbol{x}^l = \Phi^l\mathbf{P}_{l+1}^l\mathbf{I}_l^{l+1}\boldsymbol{x}^l, \qquad \text{for all } \boldsymbol{x}^l \in \mathbb{R}^{n^l}. \tag{2.11}$$

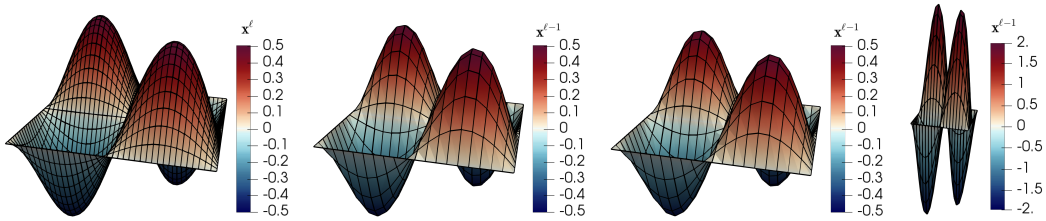

Figure 2.4: Transfer of fine-level solution to coarse level. Example considers Poisson problem discretized using uniform structured meshes and the first-order Lagrange finite elements. From left to right: fine-level iterate $\boldsymbol{x}^{l+1}$; coarse-level iterate $\boldsymbol{x}^l = \mathbf{P}^l_{l+1}\boldsymbol{x}^{l+1}$, where $\mathbf{P}^l_{l+1}$ is defined by (2.9); coarse-level iterate $\boldsymbol{x}^l = \tilde{\mathbf{P}}^l_{l+1}\boldsymbol{x}^{l+1}$, where $\tilde{\mathbf{P}}^l_{l+1}$ is defined by (2.14); coarse-level iterate $\boldsymbol{x}^l = \mathbf{R}^l_{l+1}\boldsymbol{x}^{l+1}$, where $\mathbf{R}^l_{l+1} = (\mathbf{I}^{l+1}_l)^T$.

Similarly to (2.7), we can solve least-square minimization problem using (2.11) and obtain $\mathbf{P}^l_{l+1}$ in the following form:

$$\mathbf{P}^l_{l+1} = ((\Phi^{l+1}\mathbf{I}^{l+1}_l)^T\Phi^{l+1}\mathbf{I}^{l+1}_l)^{-1}(\Phi^{l+1}\mathbf{I}^{l+1}_l)^T\Phi^{l+1}. \tag{2.12}$$

Here, we identify $(\Phi^l)^T\Phi^l$ as a mass matrix $\mathbf{M}^l \in \mathbb{R}^{n^l\times n^l}$ with entries

$$(\mathbf{M}^l)_{ij} = (N^l_i, N^l_j)_{L^2(\Omega)}.$$

This allows us to reformulate (2.12) as follows:

$$\begin{aligned}\mathbf{P}^l_{l+1} &= ((\mathbf{I}^{l+1}_l)^T\mathbf{M}^{l+1}\mathbf{I}^{l+1}_l)^{-1}(\mathbf{I}^{l+1}_l)^T\mathbf{M}^{l+1},\\ \mathbf{P}^l_{l+1} &= (\mathbf{R}^l_{l+1}\mathbf{M}^{l+1}\mathbf{I}^{l+1}_l)^{-1}\mathbf{R}^l_{l+1}\mathbf{M}^{l+1}.\end{aligned}$$

Recognizing that $\mathbf{R}^l_{l+1}\mathbf{M}^{l+1}\mathbf{I}^{l+1}_l$ is a restriction of the mass matrix $\mathbf{M}^{l+1}$ to the level l, we can now express $\mathbf{P}^l_{l+1}$ as

$$\mathbf{P}^l_{l+1} = (\mathbf{M}^l)^{-1}\mathbf{R}^l_{l+1}\mathbf{M}^{l+1}. \tag{2.13}$$

Since a mass matrix $\mathbf{M}^l$ is usually well-conditioned, the operator $\mathbf{P}^l_{l+1}$ can be applied to a vector by approximating the inverse of $\mathbf{M}^l$ with a few steps of some linear iterative method, e.g., Conjugate Gradient method.

As a cheaper alternative, we can approximate the mass matrix $\mathbf{M}^l$ by lumped mass matrix $\tilde{\mathbf{M}}^l$, given as

$$(\tilde{\mathbf{M}}^l)_{ij} = \begin{cases}\sum_{k=1}^{n^l}(\mathbf{M}^l)_{ik}, & \text{if } i = j,\\ 0, & \text{otherwise.}\end{cases}$$

Since $\tilde{\mathbf{M}}^l$ is a diagonal matrix, its inverse is also diagonal. This gives rise to a sparse and computationally cheap approximate projection operator

$$\tilde{\mathbf{P}}^l_{l+1} = (\tilde{\mathbf{M}}^l)^{-1}\mathbf{R}^l_{l+1}\mathbf{M}^{l+1}. \tag{2.14}$$

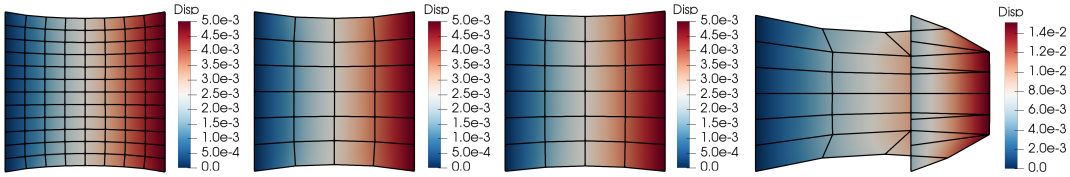

Figure 2.5: Transfer of fine-level solution to coarse level. Example considers linear elasticity discretized using structured but not uniform meshes and the first-order Lagrange finite elements. From left to right: fine-level iterate $\boldsymbol{x}^{l+1}$; coarse-level iterate $\boldsymbol{x}^l = \mathbf{P}^l_{l+1}\boldsymbol{x}^{l+1}$, where $\mathbf{P}^l_{l+1}$ is defined by (2.9); coarse-level iterate $\boldsymbol{x}^l = \tilde{\mathbf{P}}^l_{l+1}\boldsymbol{x}^{l+1}$, where $\tilde{\mathbf{P}}^l_{l+1}$ is defined by (2.14); coarse-level iterate $\boldsymbol{x}^l = \mathbf{R}^l_{l+1}\boldsymbol{x}^{l+1}$, where $\mathbf{R}^l_{l+1} = (\mathbf{I}^{l+1}_l)^T$.

Remark 5. *The properties of the lumped mass matrix vary depending on the type and order of the finite elements.*

Figure 2.4 illustrates the approximation properties of three different transfer operators, namely $\mathbf{P}^l_{l+1}$ given by (2.9), $\tilde{\mathbf{P}}^l_{l+1}$ given by (2.14), and $\mathbf{R}^l_{l+1} = (\mathbf{I}^{l+1}_l)^T$, when used to transfer iterate $\boldsymbol{x}^{l+1}$ to level l. In this example, we assume that the meshes used for a FEM discretization are uniform and structured. As we can see, the approximation properties of the projection operator $\mathbf{P}^l_{l+1}$ given by (2.9) and the approximate projection operator $\tilde{\mathbf{P}}^l_{l+1}$ given by (2.14) are almost identical. In contrast, the coarse-level iterate $\boldsymbol{x}^l$ obtained using restriction operator $\mathbf{R}^l_{l+1}$ is badly scaled. In particular, the components of the vector $\mathbf{R}^l_{l+1}\boldsymbol{x}^{l+1}$ are four times larger than components of the vector $\mathbf{P}^l_{l+1}\boldsymbol{x}^{l+1}$. Thus, if we scale $\mathbf{R}^l_{l+1}$ by factor of $1/4$, we obtain a reasonable approximation of the projection operator $\mathbf{P}^l_{l+1}$ given by (2.9). This corresponds to common observations from practice, which suggest the usage of scaled variants of $\mathbf{R}^l_{l+1}$ in order to transfer iterates to a coarse level. If we examine formula (2.13) more closely for uniform structured meshes, we notice that the application of $\mathbf{M}^{l+1}$ and $(\mathbf{M}^l)^{-1}$ to $\mathbf{R}^l_{l+1}$ scales all components of matrix $\mathbf{R}^l_{l+1}$ by a constant factor, e.g., $1/4$ for uniformly refined structured meshes in 2D.

However, if the meshes are not uniform and structured, then the scaling obtained by (2.13) or (2.14) can not be expressed by a single value. The application of the restriction operator $\mathbf{R}^l_{l+1}$ to the iterate $\boldsymbol{x}^{l+1}$ causes bad scaling in addition to distortion. This is especially problematic for vectorial and coupled problems, such as linear elasticity, where the solution vector/displacement field is obtained as a linear combination of displacements in all spatial dimensions, see Figure 2.5.

Multi-fidelity approach

Let us assume that a hierarchy of auxiliary low-cost functions $\{f^l\}_{l=1}^L$ is created using structurally cheaper approximations of f. Furthermore, we assume that on every level l, the function f^l is defined as $f^l : \mathbb{R}^{n^l} \to \mathbb{R}$, where $n^l := n^L$. Since the number of variables n^l is held fixed, the computational domain remains the same and the transfer operators become the identity - even in their algebraic forms. As a consequence, the practical

implementation of the RMTR method is simplified, compared to the traditional multi-resolution multilevel minimization approach.

Example 2.3.1. This example is concerned with a supervised learning task. We consider the dataset $\mathcal{D}^L = \{o_j, y_j\}_{j=1}^{p^L}$, which consists of p^L samples. Each j-th sample is defined by the feature vector $o_j \in \mathbb{R}^{d_o}$ and the label vector $y_j \in \mathbb{R}^{d_y}$. We are interested in finding a prediction function $f_m : \mathbb{R}^{n^L} \times \mathbb{R}^{d_o} \to \mathbb{R}$, parametrized by $\boldsymbol{x} \in \mathbb{R}^{n^L}$, that takes input o_j and outputs $f_m(\boldsymbol{x}, o_j)$ that hopefully identifies y_j.

We can find suitable parameters $\boldsymbol{x}$ of a prediction function f_m, by minimizing the following objective function

$$f^L(\boldsymbol{x}) := \frac{1}{p^L} \sum_{j=1}^{p^L} \ell(f_m(\boldsymbol{x}, o_j), y_j), \tag{2.15}$$

where $\ell : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$ denotes a loss function, which measures the misfit between the prediction given by our model $f_m(\boldsymbol{x}, o_j)$ and the actual observation y_j.

Following Reference [BKK20], the low-cost approximation f^{L-1} of the objective function (2.15) can be obtained by considering only a small subset of samples provided by the dataset $\mathcal{D}^L$. Let the subset $\mathcal{D}^{L-1}$ consist of $p^{L-1} < p^L$ samples, then we can define the low-cost function $f^{L-1} : \mathbb{R}^n \to \mathbb{R}$ as follows:

$$f^{L-1}(\boldsymbol{x}) := \frac{1}{p^{L-1}} \sum_{j=1}^{p^{L-1}} \ell(f_m(\boldsymbol{x}, o_j), y_j).$$

Although, both functions f^{L-1} and f^L utilize same parameters $\boldsymbol{x} \in \mathbb{R}^{n^L}$, the function f^{L-1} is computationally cheaper to evaluate than function f^L as it considers smaller amount of samples.

2.4 Level-dependent objective functions

On each level l, the RMTR method minimizes some nonlinear level-dependent objective function h^l. The result of this minimization, the coarse-level correction, is expected to provide a decrease in the value of the objective function on the subsequent finer level. The overall efficiency of the multilevel method relies on the quality of this prolongated coarse-level correction. The level-dependent objective functions $\{h^l\}_{l=1}^L$ are constructed during each V-cycle using some auxiliary functions $\{f^l\}_{l=1}^L$, which represent low-cost approximations of the original objective function f. The model h^L, defined on the finest level L is always identified with the original objective function f, i.e., $h^L := f^L := f$.

In this section, we review the assumptions posed on the level-dependent objective functions, such that the global convergence properties of the RMTR method [GST08, GMTW08, GK09b] are preserved. Then, we discuss three approaches that allow for

the construction of the level-dependent objective functions with desirable properties. Throughout the following, we use subscripts to denote a particular form of the level-dependent function. For instance, the function h^l_{add} denotes a level-dependent objective function constructed on a level l using the additive approach. The subscripts are omitted, if any form of function h^l can be used.

2.4.1 Properties of level-dependent objective functions

The model $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$, defined on level l has to be constructed such that the following conditions hold.

- Given initial guess $\boldsymbol{x}^l_0$ on level l, we assume that $\boldsymbol{x}^l_0 \in \mathcal{F}^l$ and that the level set

$$\mathscr{L}^l = \{\boldsymbol{x}^l \in \mathcal{F}^l \mid h^l(\boldsymbol{x}^l) \leq h^l(\boldsymbol{x}^l_0)\}$$

  is non-empty and compact. Here, symbol $\mathcal{F}^l$ denotes the level-dependent feasible set, see Section 2.5 for details.

- We assume that each function h^l is continuously differentiable on $\mathscr{L}^l$. Moreover, there exist constants $C_g > 0$ and $C_B > 0$, such that

$$\begin{aligned} \|\nabla h^l(\boldsymbol{x}^l)\| &\leq C_g, \qquad \text{for all } \boldsymbol{x}^l \in \mathscr{L}^l, \\ \|\mathbf{B}^l(\boldsymbol{x}^l)\| &\leq C_B, \qquad \text{for all } \boldsymbol{x}^l \in \mathscr{L}^l, \end{aligned}$$

  where $\mathbf{B}^l(\boldsymbol{x}^l)$ is either Hessian $\nabla^2 h^l(\boldsymbol{x}^l)$ or its approximation on the level l given the iterate $\boldsymbol{x}^l$.

- For all levels $l < L$, the function h^l has to be built such that it is locally coherent with h^{l+1}. Thus, given current fine-level iterate $\boldsymbol{x}^{l+1}_{\mu_1}$ and the coarse-level initial guess $\boldsymbol{x}^l_0 := \mathbf{P}^{l+1}_l \boldsymbol{x}^{l+1}_{\mu_1}$, the function h^l is assumed to satisfy the first-order consistency condition at $\boldsymbol{x}^l_0$ given as

$$\nabla h^l(\boldsymbol{x}^l_0) = \mathbf{R}^l_{l+1} \nabla h^{l+1}(\boldsymbol{x}^{l+1}_{\mu_1}). \tag{2.16}$$

Since the model h^l is built using low-cost function f^l, it is necessary that the function f^l also fulfils certain assumptions, such as being differentiable, bounded from below and having uniformly bounded Hessians.

2.4.2 An additive approach

Multilevel methods often construct coarse-level model $h^l_{\text{add}} : \mathbb{R}^{n^l} \to \mathbb{R}$ in an additive manner. This is achieved by correcting the low-cost function f^l associated with level l as follows:

$$h^l_{\text{add}}(\boldsymbol{x}^l) = f^l(\boldsymbol{x}^l) + \gamma^l(\boldsymbol{x}^l), \tag{2.17}$$

where $\gamma^l : \mathbb{R}^{n^l} \to \mathbb{R}$ denotes the additive correction function, defined as

$$\gamma^l(\boldsymbol{x}^l) := h^{l+1}(\mathbf{I}_l^{l+1}\boldsymbol{x}^l) - f^l(\boldsymbol{x}^l). \tag{2.18}$$

The correction function γ^l accounts for the difference between the value of the function f^l and the model h^{l+1}, defined on the subsequent finer level.

Unfortunately, the evaluation of the correction function γ^l at the coarse-level iterate $\boldsymbol{x}^l$ requires an evaluation of the fine-level model h^{l+1} at $\mathbf{I}_l^{l+1}\boldsymbol{x}^l$, see (2.18). As a result, working with coarse-level model h^l_{add} is computationally even more expensive than working with the fine-level model h^{l+1}. To ease the computational burden, we evaluate the correction function γ^l exactly only at the initial coarse-level iterate $\boldsymbol{x}_0^l$. Thus, we impose

$$\gamma^l(\boldsymbol{x}_0^l) := h^{l+1}(\boldsymbol{x}_{\mu_1}^{l+1}) - f^l(\boldsymbol{x}_0^l),$$

only at $\boldsymbol{x}_0^l := \mathbf{P}_l^{l+1}\boldsymbol{x}_{\mu_1}^{l+1}$. For any other coarse-level iterate $\boldsymbol{x}^l$, we use approximate correction function obtained by means of the Taylor approximation. The polynomial order used for the Taylor approximation then defines different variants of the level-dependent functions.

The first-order consistency

Several nonlinear multilevel schemes, such as FAS [Bra77], NMG [Hac85], or MG/OPT [Nas00], employ a first-order Taylor approximation of the additive correction function γ^l. The approximate correction function $\tilde{\gamma}_1^l : \mathbb{R}^{n^l} \to \mathbb{R}$ is defined around $\boldsymbol{x}_0^l$ as

$$\tilde{\gamma}_1^l(\boldsymbol{x}^l) = \gamma^l(\boldsymbol{x}_0^l) + \langle \nabla\gamma^l(\boldsymbol{x}_0^l),\ \boldsymbol{s}^l\rangle,$$

where $\boldsymbol{s}^l = \boldsymbol{x}^l - \boldsymbol{x}_0^l$. If we replace γ^l with $\tilde{\gamma}^l$ in (2.17), we obtain the first-order coherent level-dependent objective function $h^l_{\text{add\_1}} : \mathbb{R}^{n^l} \to \mathbb{R}$, defined as

$$h^l_{\text{add\_1}}(\boldsymbol{x}^l) := \underbrace{f^l(\boldsymbol{x}^l)}_{\text{low-cost model}} + \underbrace{h^{l+1}(\boldsymbol{x}_{\mu_1}^{l+1}) - f^l(\boldsymbol{x}_0^l)}_{0^{\text{th}}\text{-order coupling}} + \underbrace{\langle \delta\boldsymbol{g}^l_{\text{add}}(\boldsymbol{x}_0^l), \boldsymbol{s}^l\rangle}_{1^{\text{st}}\text{-order coupling}}, \tag{2.19}$$

where

$$\delta\boldsymbol{g}^l_{\text{add}}(\boldsymbol{x}_0^l) := \mathbf{R}_{l+1}^l\boldsymbol{g}_{\mu_1}^{l+1} - \nabla f^l(\boldsymbol{x}_0^l). \tag{2.20}$$

The first term in (2.19) denotes a low-cost approximation of the original objective function f, associated with level l. The second term expresses the difference between the value of a fine-level and a coarse-level objective function at $\boldsymbol{x}_{\mu_1}^{l+1}$ and $\boldsymbol{x}_0^l$, respectively. The presence of this term enforces zeroth-order consistency between h^{l+1} and $h^l_{\text{add\_1}}$ at $\boldsymbol{x}_{\mu_1}^{l+1}$ and $\boldsymbol{x}_0^l$, respectively, i.e., $h^l_{\text{add\_1}}(\boldsymbol{x}_0^l) = h^{l+1}(\boldsymbol{x}_{\mu_1}^{l+1})$. The third term in (2.19) consists of $\delta\boldsymbol{g}^l_{\text{add}}$ quantity, which expresses the difference between the restricted gradient

obtained after pre-smoothing on level $l+1$ and the initial gradient at level l (evaluated at $\boldsymbol{x}_0^l = \mathbf{P}_{l+1}^l \boldsymbol{x}_{\mu_1}^{l+1}$). The presence of this term enforces the relation

$$\boldsymbol{g}_0^l = \nabla h_{\text{add\_1}}^l(\boldsymbol{x}_0^l) = \mathbf{R}_{l+1}^l \boldsymbol{g}_{\mu_1}^{l+1}. \tag{2.21}$$

As a consequence, the first step of the minimization process on level l will be performed in the direction of the negative gradient restricted from the level $l+1$.

The multilevel method configured with model (2.19) is known to yield fast convergence, even with a poor initial guess, under the assumption that $h_{\text{add\_1}}^l$ provides a good approximation of the error after pre-smoothing. If the coarse-level model $h_{\text{add\_1}}^l$ has poor approximation properties, then the level l might produce an inadequate correction with respect to the level $l+1$. This results in slow convergence of a multilevel method, even if the initial error is very low and the nonlinearity is weak [YD06].

At the end, we point out that the zeroth-order consistency term, $h^{l+1}(\boldsymbol{x}_{\mu_1}^{l+1}) - f^l(\boldsymbol{x}_0^l)$, in (2.19), does not effect an evaluation of the derivatives of the function $h_{\text{add\_1}}^l$. Therefore, it is often neglected in practice.

The second-order consistency

We can approximate the correction function γ^l more precisely using a second-order Taylor approximation $\tilde{\gamma}_2^l : \mathbb{R}^{n^l} \to \mathbb{R}$ defined as

$$\tilde{\gamma}_2^l(\boldsymbol{x}^l) = \gamma^l(\boldsymbol{x}_0^l) + \langle \nabla \gamma^l(\boldsymbol{x}_0^l),\ \boldsymbol{s}^l \rangle + \frac{1}{2} \langle \boldsymbol{s}^l\ \nabla^2 \gamma^l(\boldsymbol{x}_0^l),\ \boldsymbol{s}^l \rangle. \tag{2.22}$$

Plugging (2.22) into (2.17) gives rise to the level-dependent objective function $h_{\text{add\_2}}^l : \mathbb{R}^{n^l} \to \mathbb{R}$ of the following form:

$$h_{\text{add\_2}}^l(\boldsymbol{x}^l) := \underbrace{f^l(\boldsymbol{x}^l)}_{\text{low-cost model}} + \underbrace{\langle \delta \boldsymbol{g}_{\text{add}}^l(\boldsymbol{x}_0^l), \boldsymbol{s}^l \rangle}_{1^{\text{st}}\text{-order coupling}} + \underbrace{\frac{1}{2} \langle \boldsymbol{s}^l, \delta \mathbf{B}_{\text{add}}^l(\boldsymbol{x}_0^l) \boldsymbol{s}^l \rangle}_{2^{\text{nd}}\text{-order coupling}}, \tag{2.23}$$

where $\delta \boldsymbol{g}_{\text{add}}^l$ is defined as in (2.20). The quantity $\delta \mathbf{B}_{\text{add}}^l$ expresses the difference between restricted fine-level Hessian and the initial coarse-level Hessian, i.e.,

$$\delta \mathbf{B}_{\text{add}}^l(\boldsymbol{x}_0^l) := \mathbf{R}_{l+1}^l \mathbf{B}_{\mu_1}^{l+1} \mathbf{I}_l^{l+1} - \mathbf{B}_0^l. \tag{2.24}$$

The minimization of model (2.23) produces the first coarse-level correction, which coincides with a restricted Newton step (if Δ^l is sufficiently large and the optimization problem is unconstrained). We note that all subsequent coarse-level corrections involve knowledge acquired by evaluating the low-cost function f^l.

The model (2.23) was used in a multilevel context for the first time by Yavneh and Dardyk in Reference [YD06]. Yavneh and Dardyk also demonstrated that the multilevel method configured with model (2.23) is guaranteed to achieve at least as good and

usually better convergence rate than the multilevel method, which employs the first-order consistent model (2.19) or Galerkin model (2.25), discussed below.

Despite its good approximation properties, the second-order consistency approach is computationally more expensive than the first-order consistency approach. This is due to the fact, that the model (2.23) requires an explicit representation of matrix $\delta \mathbf{B}^l_{\text{add}}$. In addition, the products involving the matrix $\delta \mathbf{B}^l_{\text{add}}$ are used for evaluation of the function h^l, as well as its gradient, and Hessian.

Remark 6. *Following standard practice, the model* (2.23) *neglects the zeroth-order consistency term.*

Galerkin approximation

Exploiting ideas from the linear multigrid, a level-dependent objective function $h^l_{\text{add\_G}} : \mathbb{R}^{n^l} \to \mathbb{R}$ can be obtained as

$$h^l_{\text{add\_G}}(\boldsymbol{x}^l) := \langle \mathbf{R}^l_{l+1} \boldsymbol{g}^{l+1}_{\mu_1}, \boldsymbol{s}^l \rangle + \frac{1}{2} \langle \boldsymbol{s}^l, (\mathbf{R}^l_{l+1} \mathbf{B}^{l+1}_{\mu_1} \mathbf{I}^{l+1}_l) \boldsymbol{s}^l \rangle, \tag{2.25}$$

where $\mathbf{R}^l_{l+1} \boldsymbol{g}^{l+1}_{\mu_1}$ and $\mathbf{R}^l_{l+1} \mathbf{B}^{l+1}_{\mu_1} \mathbf{I}^{l+1}_l$ represent the gradient and the Hessian restricted from level $l+1$, respectively. In the unconstrained convex case, the minimization of model (2.25) within the RMTR method is equivalent to one iteration of Galerkin linear multigrid to the associated Newton equation.

The approximation properties and the quality of the model (2.25) are determined by the properties of the Hessian and the iterate on the finest level. If the finest-level iterate is in the local neighborhood of a solution, then the model (2.25) is expected to approximate h^{l+1} well. However, the first/second-order consistent models usually allow for better convergence rates, for generic nonlinear optimization problems [YD06].

The model (2.25) can be obtained from the second-order consistent model (2.23), by setting $f^l(\boldsymbol{x}^l) := 0$. In comparison to previously discussed approaches, the model (2.25) allows for a simplified implementation of the RMTR algorithm. This is due to the fact, that the knowledge of the low-cost function f^l is not required in (2.25). In the end, we also note that the model (2.25) was also investigated within the line-search multilevel framework in Reference [HKP19].

2.4.3 A multiplicative approach

Trust-region methods that exploit multiple fidelities often construct level-dependent objective functions using the multiplicative approach. In this case, the low-cost approximation f^l associated with level l is made consistent with the model h^{l+1} as follows:

$$h^l_{\text{mult}}(\boldsymbol{x}^l) = \beta^l(\boldsymbol{x}^l) f^l(\boldsymbol{x}^l), \tag{2.26}$$

where $\beta^l : \mathbb{R}^{n^l} \to \mathbb{R}$ is a multiplicative correction function, given by

$$\beta^l(\boldsymbol{x}^l) := \frac{h^{l+1}(\mathbf{I}_l^{l+1}\boldsymbol{x}^l)}{f^l(\boldsymbol{x}^l)}, \tag{2.27}$$

where $f^l(\boldsymbol{x}^l) \neq 0$.

Similarly to the additive case, evaluating the correction function β^l precisely at all coarse-level iterates is computationally expensive. Therefore, we impose (2.27) only at $\boldsymbol{x}_0^l$, i.e.,

$$\beta^l(\boldsymbol{x}_0^l) := \frac{h^{l+1}(\boldsymbol{x}_{\mu_1}^{l+1})}{f^l(\boldsymbol{x}_0^l)},$$

where $f^l(\boldsymbol{x}_0^l) \neq 0$. At any other coarse-level iterate $\boldsymbol{x}^l$, we employ approximation of β^l. The approximation is again constructed by means of a Taylor approximation.

The major shortcoming of the multiplicative approach is its numerical instability as the value of $f^l(\boldsymbol{x}^l)$ in (2.27) approaches zero. Marduel et al. [MTT02] proposed to overcome this difficulty by adding a constant $\kappa > 0$ to the definition of β^l. A stablized model h^l is then constructed as

$$h^l_{\text{mult}}(\boldsymbol{x}^l) = \beta^l(\boldsymbol{x}^l)(f^l(\boldsymbol{x}^l) + \kappa) - \kappa,$$

where

$$\beta^l(\boldsymbol{x}^l) = \frac{h^{l+1}(\mathbf{I}_l^{l+1}\boldsymbol{x}^l) + \kappa}{f^l(\boldsymbol{x}^l) + \kappa}.$$

The first-order consistency

We can approximate the correction function β^l by means of its first-order Taylor approximation $\tilde{\beta}_1^l : \mathbb{R}^{n^l} \to \mathbb{R}$, defined around $\boldsymbol{x}_0^l$ as follows:

$$\tilde{\beta}_1^l(\boldsymbol{x}^l) = \beta^l(\boldsymbol{x}_0^l) + \langle \nabla\beta(\boldsymbol{x}^l),\ \boldsymbol{s}^l \rangle. \tag{2.28}$$

The first-order consistent multiplicative level-dependent model $h^l_{\text{mult\_1}} : \mathbb{R}^{n^l} \to \mathbb{R}$ is obtained by replacing β^l with $\tilde{\beta}_1^l$ in (2.26), which gives rise to

$$h^l_{\text{mult\_1}}(\boldsymbol{x}^l) := \underbrace{\tilde{\beta}_1^l(\boldsymbol{x}^l)}_{1^{\text{st}}\text{-order coupling}} \underbrace{f^l(\boldsymbol{x}^l).}_{\text{low-cost model}} \tag{2.29}$$

The numerical evaluation of $\tilde{\beta}_1^l$ amounts to

$$\tilde{\beta}_1^l(\boldsymbol{x}^l) := \frac{h^{l+1}(\boldsymbol{x}_{\mu_1}^{l+1})}{f^l(\boldsymbol{x}_0^l)} + \langle \delta \boldsymbol{g}^l_{\text{mult}}(\boldsymbol{x}_0^l), \boldsymbol{s}^l \rangle,$$

where $\delta \boldsymbol{g}^l_{\text{mult}}$ is given by

$$\delta \boldsymbol{g}^l_{mult}(\boldsymbol{x}^l_0) := \frac{1}{f^l(\boldsymbol{x}^l_0)}\big(\mathbf{R}^l_{l+1}\nabla h^{l+1}(\boldsymbol{x}^{l+1}_{\mu_1})\big) - \frac{h^{l+1}(\boldsymbol{x}^{l+1}_{\mu_1})}{(f^l(\boldsymbol{x}^l_0))^2}\nabla f^l(\boldsymbol{x}^l_0).$$

Straightforward calculations show, that model $h^l_{\text{mult\_1}}$ defined by (2.29) is zeroth- and first-order consistent with h^{l+1} at $\boldsymbol{x}^l_0$ and $\boldsymbol{x}^{l+1}_{\mu_1}$, respectively. Hence, it fulfills the assumption (2.16) and therefore it can be used within the RMTR method.

From a computational point of view, additive and multiplicative first-order approaches are comparable. However, the behavior of the multiplicative model is very different compared to its additive counterpart. This is due to the fact that the multiplication of f^l with correction function $\tilde{\beta}^l$ can introduce new minima. For instance, let us suppose that a low-cost function f^l is represented by a second-order polynomial. The multiplication with a correction term $\tilde{\beta}^l_1$ increases the order of the polynomial, i.e., we obtain a model h^l, which is quartic and has, in general, more minima than quadratic function f^l.

The second-order consistency

Similarly to the additive approach, we can also utilize the second-order Taylor approximation of the correction function β^l to define the model h^l. Following References [GPR+04, EGC04], the level-dependent objective function $h^l_{\text{mult\_2}} : \mathbb{R}^{n^l} \to \mathbb{R}$ now takes the form

$$h^l_{\text{mult\_2}}(\boldsymbol{x}^l) := \underbrace{\tilde{\beta}^l_2(\boldsymbol{x}^l)}_{2^{\text{nd}}\text{-order coupling}} \quad \underbrace{f^l(\boldsymbol{x}^l)}_{\text{low-cost model}}, \tag{2.30}$$

where $\tilde{\beta}^l_2(\boldsymbol{x}^l)$ is given by

$$\tilde{\beta}^l_2(\boldsymbol{x}^l) := \tilde{\beta}^l_1(\boldsymbol{x}^l)\langle \boldsymbol{s}^l, \delta\mathbf{B}^l_{\text{mult}}(\boldsymbol{x}^l_0)\, \boldsymbol{s}^l\rangle,$$

where

$$\begin{aligned}\delta\mathbf{B}^l_{\text{mult}}(\boldsymbol{x}^l_0) :=& \frac{1}{f^l(\boldsymbol{x}^l_0)}\big(\mathbf{R}^l_{l+1}\mathbf{B}^{l+1}_{\mu_1}\mathbf{I}^{l+1}_l\big) - \frac{h^{l+1}(\boldsymbol{x}^l_{\mu_1})}{(f^l(\boldsymbol{x}^l_0))^2}\mathbf{B}^l_0 + \frac{2h^{l+1}(\boldsymbol{x}^{l+1}_{\mu_1})}{(f^l(\boldsymbol{x}^l_0))^3}\nabla f^l(\boldsymbol{x}^l_0)(\nabla f^l(\boldsymbol{x}^l_0))^T \\ &- \frac{1}{(f^l(\boldsymbol{x}^l_0))^2}\nabla f^l(\boldsymbol{x}^l_0)\big(\mathbf{R}^l_{l+1}\nabla h^{l+1}(\boldsymbol{x}^{l+1}_{\mu_1})\big)^T \\ &+ \mathbf{R}^l_{l+1}\nabla h^{l+1}(\boldsymbol{x}^{l+1}_{\mu_1})(\nabla f^l(\boldsymbol{x}^l_0))^T.\end{aligned} \tag{2.31}$$

The second-order multiplicative correction scheme is computationally more expensive than the second-order additive correction scheme. This is due to the fact that the evaluation of (2.31) consists of outer products, which produce a dense matrix. We can reduce memory requirements associated with storing $\delta\mathbf{B}^l_{\text{mult}}$ explicitly, by applying $\delta\mathbf{B}^l_{\text{mult}}(\boldsymbol{x}^l_0)$ directly to a tensor, when necessary.

2.4.4 A hybrid approach

Both, additive and multiplicative approaches have their merits and drawbacks. The additive approach adds new terms to the low-cost function f^l. This can be interpreted as uniform translation (zeroth-order), rotation (first-order), and bending (second-order) of the function graph, see also Figure 2.6. In the multiplicative approach, the low-cost function f^l is multiplied with a linear (first-order) or a quadratic (second-order) function, which usually introduces more skewing [FGB18]. This may not be desirable if the objective function f and its low-cost approximation f^l are in good agreement, at least locally. However, when functions are not in good agreement, then additional skewing can be beneficial. For example, if the polynomial order of fine-level function is higher than its low-cost approximation, then the multiplicative model usually performs better than its additive counterpart.

In general, we do not know a priori, whether the additive or the multiplicative model is more suitable, given our optimization problem. The hybrid approach, proposed by Gano et al. [GRS05], effectively combines both additive and multiplicative correction schemes. In this case, a level-dependent objective function $h_{\text{mix}} : \mathbb{R}^{n^l} \to \mathbb{R}$ is defined as a convex combination of additive h^l_{add} and multiplicative h^l_{mult} model, thus as

$$h^l_{\text{mix}}(\boldsymbol{x}^l) := w\, h^l_{\text{add}}(\boldsymbol{x}^l) + (1-w)\, h^l_{\text{mult}}(\boldsymbol{x}^l). \tag{2.32}$$

A weighting parameter $w \in [0,1]$ in (2.32) ensures that the local first-order consistency properties of h^l_{add} and h^l_{mult} are retained. If $w = 0$, or $w = 1$, then model h^l_{mix} reduces to the multiplicative or to the additive model, respectively.

It is important to chose weighting parameter w in (2.32), such that the approximation properties of the model h^l_{mix} are maximized. Eldred et al. [EGC04] proposed to select w automatically by matching fine-level and coarse-level function value at previously evaluated fine-level iterate, denoted by $\boldsymbol{x}^{l+1}_p$. The weighting parameter w is obtained using current iterate $\boldsymbol{x}^l$ on a level l as follows:

$$w = \frac{h^{l+1}(\boldsymbol{x}^{l+1}_p) - h^l_{\text{mult}}(\boldsymbol{x}^l)}{h^l_{\text{add}}(\boldsymbol{x}^l) - h^l_{\text{mult}}(\boldsymbol{x}^l)}. \tag{2.33}$$

From computational point of view, the ratio (2.33) is cheap to evaluate, as the fine-level function value $h^{l+1}(\boldsymbol{x}^{l+1}_p)$ is readily available from the previous trust-region step on level $l+1$ (successful or unsuccessful). More advanced method for estimating the value of w can be found in Reference [FGB17].

Example 2.4.1. Let us consider the objective function $f^L : \mathbb{R} \to \mathbb{R}$ of the following form:

$$f^L(x) = (0.5x-3)^2 \sin(1.3x+1) + \exp(-0.25x),$$

and its low-cost approximation $f^{L-1} : \mathbb{R} \to \mathbb{R}$ defined as

$$f^{L-1}(x) = (0.75x-4)^2 - 7.$$

The function f^L has two minima at $x = 2.56$ and $x = 8.18$, while f^{L-1} has only one minimum at $x = 5.13$. Although the minimum of f^{L-1} and f^L do not coincide, f^{L-1} can be used to find a minimum of f^L, if it is utilized to construct model h^{L-1}, which is at least first-order consistent with f^L. Figures 2.6 and 2.7 illustrate coarse-level models h^{L-1}_{add}, h^{L-1}_{mult}, h^{L-1}_{mix} obtained using additive, multiplicative and hybrid approaches, respectively. Models illustrated on Figures 2.6 and 2.7 are built around different iterate x^L_0. As we can see, the multiplicative corrections schemes introduce more skewing than additive schemes. We also observe that second-order schemes tend to approximate f^L more accurately than the first-order schemes. We also observe, that using hybrid functions can be beneficial, as the automatic choice of weighing parameter in (2.33), helps to decide which scheme is more appropriate, given x^l_0. The presented example employed $x^{l+1}_p = x^{l+1}_{\mu_1} - 0.5$.

2.5 Multilevel treatment of constraints

In this section, we discuss how to construct level-dependent feasible sets $\{\mathcal{F}^l\}^L_{l=1}$. As we will see, the construction of level-feasible sets $\{\mathcal{F}^l\}^L_{l=1}$ requires a careful treatment of the trust-region and the variable bounds on all levels of the multilevel hierarchy. We discuss how to treat the trust-region and the variable bounds for both multi-resolution and multi-fidelity variants of the RMTR method.

2.5.1 Properties of level-dependent feasible sets

Recall (2.1), the minimization on a given level l is performed with respect to a feasible set $\mathcal{F}^l$, defined as

$$\mathcal{F}^l := \{\boldsymbol{x}^l \in \mathbb{R}^{n^l} \mid \mathbf{l}^l \leq \boldsymbol{x}^l \leq \mathbf{u}^l\}, \tag{2.34}$$

where $\mathbf{l}^l, \mathbf{u}^l \in (\mathbb{R} \cup \{-\infty\} \cup \{\infty\})^{n^l}$. By design, the feasible set $\mathcal{F}^L$ on the finest level coincides with the feasible set $\mathcal{F}$ of our minimization problem (1.1), i.e., $\mathcal{F}^L := \mathcal{F}$, and $\mathbf{l}^L := \mathbf{l}$, $\mathbf{u}^L := \mathbf{u}$. On all other levels, the set $\mathcal{F}^l$ has to be constructed, such that it satisfies the following conditions.

- Prolongated coarse-level correction $\boldsymbol{s}^l$ produces a feasible trial point on level $l+1$, such that

$$\boldsymbol{x}^{l+1}_{\mu_1} + \mathbf{I}^{l+1}_l \boldsymbol{s}^l \in \mathcal{F}^{l+1}, \tag{2.35}$$

  holds for $\boldsymbol{s}^l = \boldsymbol{x}^l_* - \boldsymbol{x}^l_0$, where $\boldsymbol{x}^l_* \in \mathcal{F}^l$.
- The size of prolongated coarse-level correction remains bounded by the current trust-region radius on level l, i.e.,

$$\|\mathbf{I}^l_{l-1} \boldsymbol{s}^{l-1}\|_p \leq \Delta^l_{\mu_1}. \tag{2.36}$$

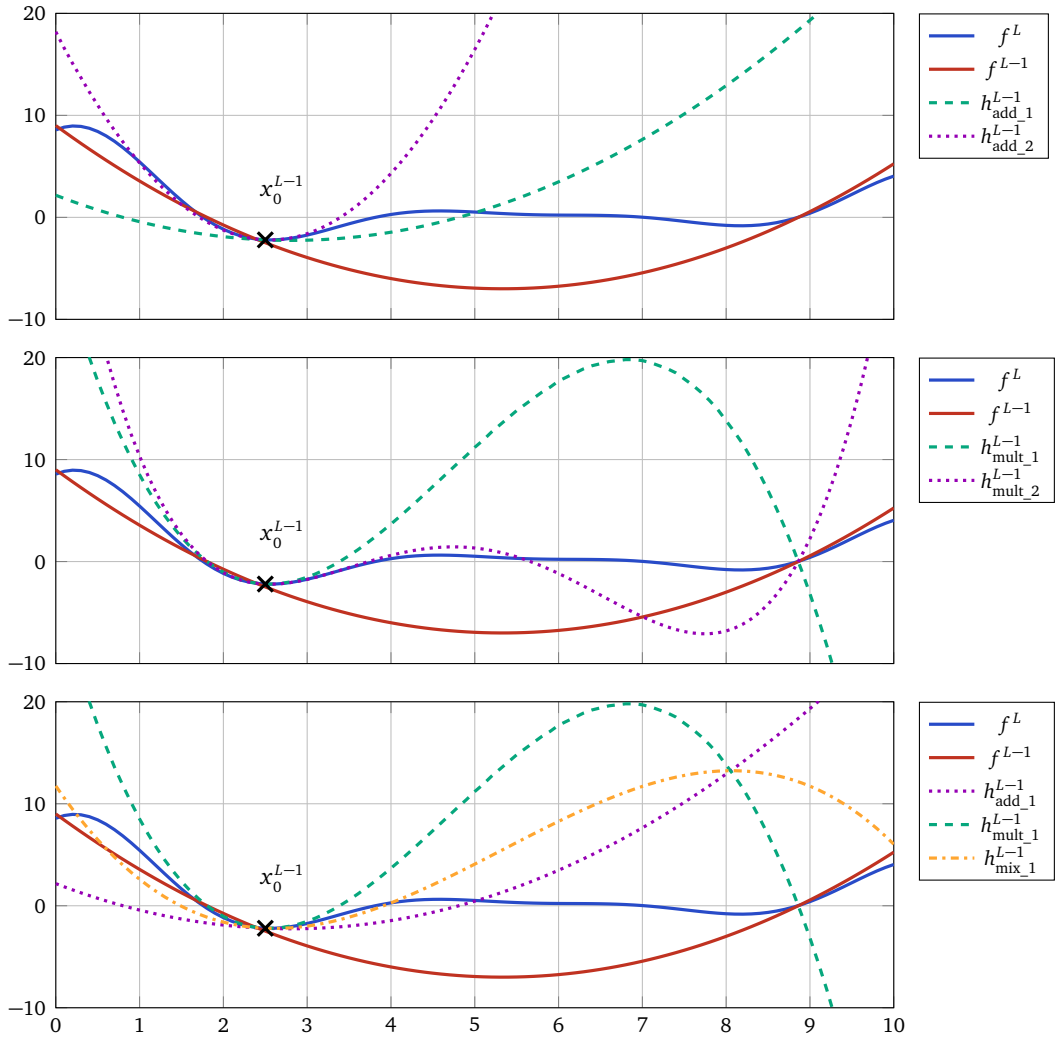

Figure 2.6: Coarse-level objective functions h^{L-1} constructed around point $x_0^{L-1} = 2.5$. The original/finest-level objective function f^L is represented by blue color, while its low-cost approximation f^{L-1} by red color. *Top:* Additive approach using the first-order (dashed line) and the second-order (dotted line) order consistency. *Middle:* Multiplicative approach using the first-order (dashed line) and the second-order (dotted line) consistency. *Bottom:* The first-order hybrid approach (dash-dotted line) using a convex combination of additive (dotted line) and multiplicative (dashed line) coarse-level objective functions.

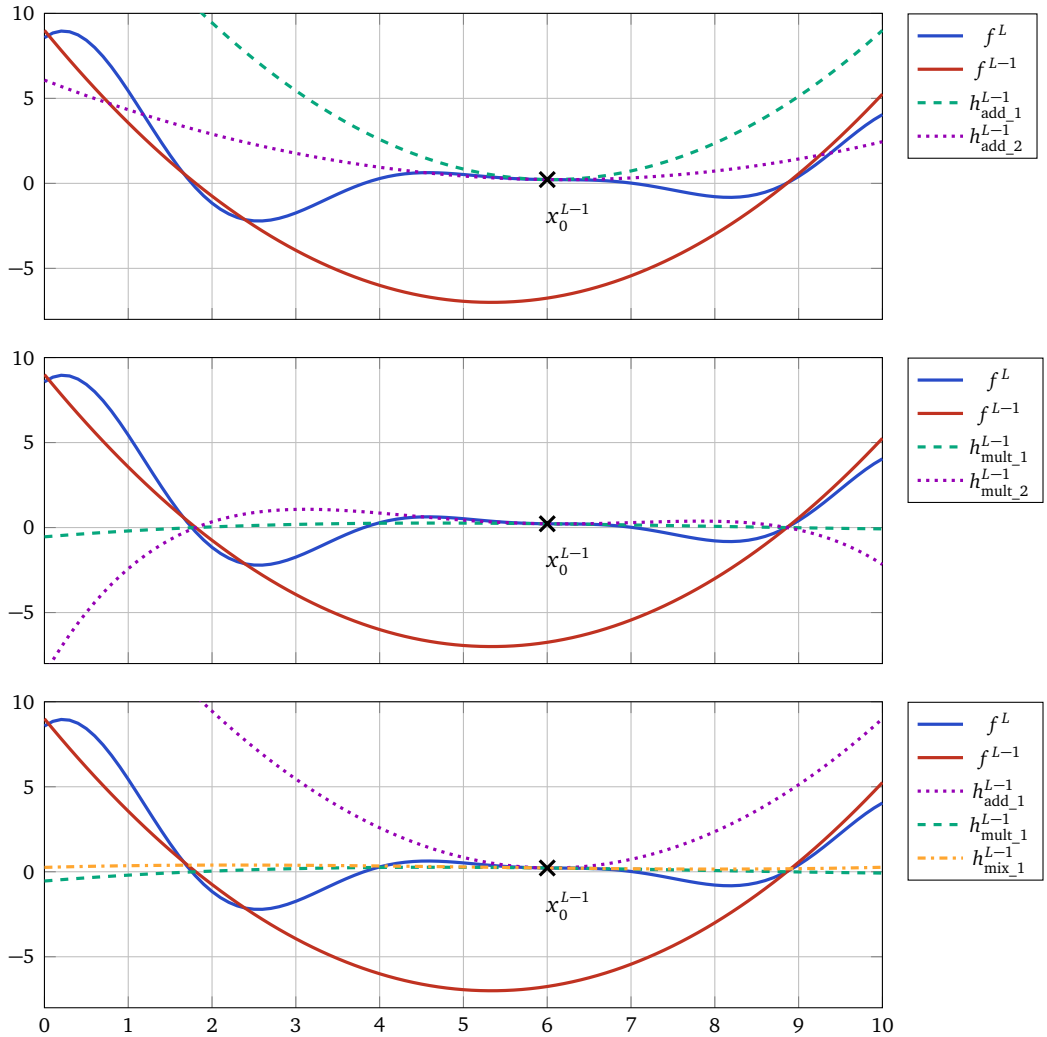

Figure 2.7: Coarse-level objective functions h^{L-1} constructed around point $x_0^{L-1} = 6.0$. The original/finest-level objective function f^L is represented by blue color, while its low-cost approximation f^{L-1} by red color. *Top:* Additive approach using the first-order (dashed line) and the second-order (dotted line) order consistency. *Middle:* Multiplicative approach using the first-order (dashed line) and the second-order (dotted line) consistency. *Bottom:* The first-order hybrid approach (dash-dotted line) using a convex combination of additive (dotted line) and multiplicative (dashed line) coarse-level objective functions.

2.5.2 Construction of level-dependent feasible sets

The construction of the level-dependent feasible sets $\{\mathcal{F}^l\}_{l=1}^L$ relies heavily on the properties of the employed transfer operators. As a consequence, the construction of the level-dependent feasible sets $\{\mathcal{F}^l\}_{l=1}^L$ has to be performed differently for the multi-resolution and the multi-fidelity variants of the RMTR method.

Multi-fidelity approach

Let us consider the multi-fidelity variant of the RMTR method. In this case, all three transfer operators (prolongation, restriction, projection) are defined to be identity. Therefore, the k-th component of vectors $\mathbf{l}^l$ and $\mathbf{u}^l$ from (2.34) can be obtained as follows:

$$(\mathbf{l}^l)_k := \max[(\mathbf{l}^{l+1})_k, (\boldsymbol{x}_{\mu_1}^{l+1})_k - \Delta_{\mu_1}^{l+1}], \quad (\mathbf{u}^l)_k := \min[(\mathbf{u}^{l+1})_k, (\boldsymbol{x}_{\mu_1}^{l+1})_k + \Delta_{\mu_1}^{l+1}]. \tag{2.37}$$

Thus, the bounds $\mathbf{l}^l, \mathbf{u}^l$ are constructed by intersecting $\mathbf{l}^{l+1}$ or $\mathbf{u}^{l+1}$ with the trust-region bounds defined on level $l+1$ after μ_1 pre-smoothing steps.

Multi-resolution approach

Let us consider the multi-resolution variant of the RMTR method. In this case, the definition of $\mathcal{F}^l$ is more involved than for the multi-fidelity variant of the RMTR method. More precisely, we define $\mathcal{F}^l$ for all $l < L$ as follows:

$$\mathcal{F}^l := \mathcal{L}^l \cap \mathcal{S}^l. \tag{2.38}$$

The set $\mathcal{L}^l$, cf. (2.43), defines the lower and upper bounds associated with each level l, such that (2.35) is satisfied. This is usually done by projecting the variable bounds associated with a level $l+1$ to a level l. In addition, the set $\mathcal{S}^l$, cf. (2.40), keeps track of the trust-region bounds projected from a level $l+1$ to a level l, such that the requirement (2.36) is satisfied. We note that the construction of the set $\mathcal{S}^l$ differs depending on the norm, which is used to define trust-region. If the trust-region is defined by the ℓ_∞-norm, then the construction of $\mathcal{S}^l$ can be performed in a component-wise manner. If the trust-region is defined by the ℓ_2-norm, then the numerical evaluation of $\mathcal{S}^l$ and consequently $\mathcal{F}^l$ is more challenging. For this reason, we employ the RMTR method with the trust-region defined by the ℓ_2-norm only for unconstrained problems.

As we will see in Section 2.5.3, and 2.5.4, the definition of the sets $\mathcal{S}^l$ and $\mathcal{L}^l$ depend on the iterate $\boldsymbol{x}_{\mu_1}^{l+1}$ obtained after pre-smoothing on level $l+1$. Therefore, the set $\mathcal{F}^l$ has to be also re-evaluated every time the algorithm enters a given level l. By design, we can employ different constraint projection rules for the trust-region and the variable bounds. This is desirable, as the bounded violation of the trust-region constraints is permitted by the RMTR convergence theory [GMTW08]. However, the violation of

variable bounds is not allowed. From an implementation point of view, using different projection rules for the trust-region and the variable bounds require higher memory and tedious bookkeeping, as we have to work with five types of constraints (defined by the sets $\mathcal{B}^l, \mathcal{S}^l, \mathcal{L}^l, \mathcal{F}^l, \mathcal{W}^l$) on all levels.

2.5.3 Multilevel treatment of trust-region constraints

In this section, we review methods for projecting the trust-region bounds to coarser levels. Our discussion is divided into two parts, depending on the norm that defines the trust-region radius.

Trust-region constraints generated by the ℓ_2-norm

Let us assume that the model problem (1.1) is unconstrained and that the trust-region is defined by the ℓ_2-norm. In this particular case, it is not necessary to construct the set $\mathcal{S}^l$ and therefore also the set $\mathcal{F}^l$ explicitly. Instead, we can ensure that the requirement (2.35) is satisfied by reformulating the level-dependent minimization problem (2.2) as follows:

$$\begin{aligned} \underset{\boldsymbol{s}^l \in \mathbb{R}^{n^l}}{\text{minimize}} \quad & h^l(\boldsymbol{x}_0^l + \boldsymbol{s}^l), \\ \text{subject to} \quad & \|\boldsymbol{s}^l\|_l \leq \Delta_{\mu_1}^{l+1}, \end{aligned} \tag{2.39}$$

where $\|\cdot\|_l$ is the level-dependent norm defined by

$$\|\boldsymbol{s}_i^l\|_l := \begin{cases} \|\mathbf{I}_{L-1}^L \cdots \mathbf{I}_l^{l+1} \boldsymbol{s}_i^l\|, & \text{if} \quad l < L, \\ \|\boldsymbol{s}_i^l\|, & \text{if} \quad l = L. \end{cases}$$

The constraint in (2.39) guarantees that the prolongated coarse-level correction does not violate the current fine-level trust-region radius $\Delta_{\mu_1}^{l+1}$.

We can incorporate the constraint from (2.39) explicitly into the trust-region radius update rules. As demonstrated by Algorithm 5, the new trust-region radius, Δ_i^l, on a given level l and iterate i, is obtained in two steps. First, we compute the intermediate radius Δ_+^l, by following standard trust-region update rules. In the second step, we determine a threshold δ_i^l on the current coarse-level correction $\boldsymbol{s}_i^{l-1}$, such that $\|\mathbf{I}_{l-1}^l \boldsymbol{s}^{l-1}\| \leq \Delta_{\mu_1}^l$. Finally, $\min[\Delta_+^l, \delta_i^l]$ defines a new trust-region radius Δ_{i+1}^l. Figure 2.8 on the left illustrates the process. Note, we assume $\Delta_{\mu_1}^{l+1} = \infty$ for $l = L$.

Trust-region constraints generated by the ℓ_∞-norm

Let us assume that the trust-region is defined by the ℓ_∞-norm. Following Reference [GMTW08], we briefly review a technique for projecting the trust-region bounds to the coarser levels. The set $\mathcal{S}^l$, which contains trust-region bounds inherited from a finer level, is defined as

$$\mathcal{S}^l := \{\boldsymbol{x}^l \in \mathbb{R}^{n^l} \mid \mathbf{tl}^l \leq \boldsymbol{x}^l \leq \mathbf{tu}^l\}, \tag{2.40}$$

Algorithm 5 Radius_update

Require: $l \in \mathbb{N}, \rho_i,\ \Delta_i^l,\ \Delta_{\mu_1}^{l+1} \in \mathbb{R}$
Constants: $\eta_1, \eta_2, \gamma_1, \gamma_2 \in \mathbb{R}$, where $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$

$$\Delta_+^l = \begin{cases} \gamma_1 \Delta_i^l, & \rho_i < \eta_1, \\ \Delta_i^l, & \rho_i \in [\eta_1, \eta_2], \\ \gamma_2 \Delta_i^l, & \rho_i > \eta_2 \end{cases}$$

$$\delta_i^l = \Delta_{\mu_1}^{l+1} - \|\boldsymbol{x}_i^l - \boldsymbol{x}_0^l\|_l$$

$$\Delta_{i+1}^l = \min[\Delta_+^l,\ \delta_i^l]$$

return Δ_{i+1}^l

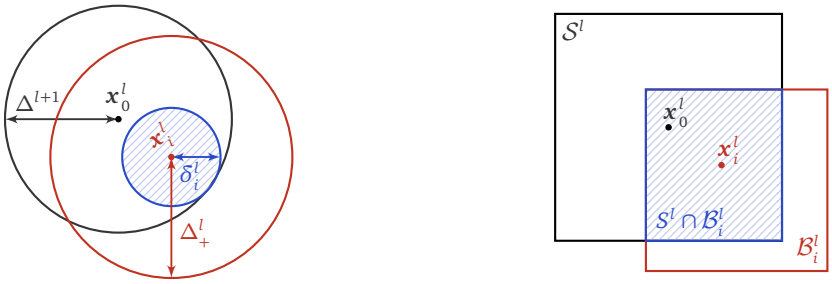

Figure 2.8: Preservation of the fine-level trust-region radius. *Left:* The trust-region radius defined by the ℓ_2-norm. *Right:* The trust-region radius defined by the ℓ_∞-norm.

where

$$\mathbf{tl}^l = (\mathbf{P}_{l+1}^l)^+ \mathbf{ttl}^{l+1} + (\mathbf{P}_{l+1}^l)^- \mathbf{ttu}^{l+1}, \qquad \mathbf{tu}^l = (\mathbf{P}_{l+1}^l)^+ \mathbf{ttu}^{l+1} + (\mathbf{P}_{l+1}^l)^- \mathbf{ttl}^{l+1}. \tag{2.41}$$

Here, the matrices $(\mathbf{P}_{l+1}^l)^+, (\mathbf{P}_{l+1}^l)^- \in \mathbb{R}^{n^l \times n^{l+1}}$ are obtained by additive decomposition of the projection operator $\mathbf{P}_{l+1}^l$, i.e., $\mathbf{P}_{l+1}^l = (\mathbf{P}_{l+1}^l)^+ + (\mathbf{P}_{l+1}^l)^-$, such that $(\mathbf{P}_{l+1}^l)^+$ contains only positive and $(\mathbf{P}_{l+1}^l)^-$ contains only negative elements of $\mathbf{P}_{l+1}^l$. The vectors $\mathbf{ttl}^{l+1}, \mathbf{ttu}^{l+1} \in \mathbb{R}^{n^{l+1}}$ from (2.41) are defined as

$$\begin{aligned} (\mathbf{ttl}^{l+1})_k &:= \max[(\mathbf{tl}^{l+1})_k, (\boldsymbol{x}_{\mu_1}^{l+1})_k - \Delta_{\mu_1}^{l+1}], \\ (\mathbf{ttu}^{l+1})_k &:= \min[(\mathbf{tu}^{l+1})_k, (\boldsymbol{x}_{\mu_1}^{l+1})_k + \Delta_{\mu_1}^{l+1}], \end{aligned} \tag{2.42}$$

where $\mathbf{tl}^{l+1}, \mathbf{tu}^{l+1}$ denote the bounds defining the set $\mathcal{S}^{l+1}$ associated with level $l+1$. The symbol $\Delta_{\mu_1}^{l+1}$ in (2.42) denotes the current trust-region radius on level $l+1$, thus obtained after μ_1 pre-smoothing steps.

Remark 7. *We assume, that all components of the lower bound vector* $\mathbf{tl}^{L+1}$ *are equal to* $-\infty$*, while components of the upper bound vector* $\mathbf{tu}^{L+1}$ *are set to* ∞*.*

The rules for projecting the trust-region bounds defined by (2.41) allow for coarse-level corrections, that slightly violate the requirement (2.36). At the same time, the rules

from (2.41) are less restrictive than constraint projection rules that will be presented in Section 2.5.4 for variable bounds. Therefore, they allow for larger coarse-level corrections. We remark, that one can apply any rules presented in Section 2.5.4 to project the trust-region bounds. If we employ the same projection rules for the trust-region and variable bounds, then the set $\mathcal{F}^l$ can be formed without explicitly forming $\mathcal{L}^l$ and $\mathcal{S}^l$, for all $l = 1, \ldots, L-1$.

2.5.4 Multilevel treatment of variable bounds

In this section, we describe how to treat variable bounds in the multilevel framework. This is usually not trivial, as the coarse levels are often not capable of resolving the finest-level constraints well, especially if the constraints are oscillatory [KK01]. The initial attempts to incorporate the constraints into the multilevel framework are associated with solving constrained quadratic minimization problems, see for example References [Man84, ML84, BC83, HM83, Hop87]. The devised methods propose several projection rules for constructing the coarse-level constraints, such that prolongated coarse-level correction does not violate the variable bounds on the finest level. Unfortunately, the proposed projection rules provide quite a narrow approximation of the finest-level constraints. As a consequence, the resulting multilevel methods converge significantly slower than standard linear multigrid, when applied to the unconstrained problems. In order to speed up the convergence, Kornhuber proposes an active-set multigrid method [Kor94a]. The developed method, called truncated monotone multigrid, employs a truncated basis approach and recovers the convergence rate of the unconstrained multigrid once the exact active-set is detected [HK94, KY94].

In the context of nonlinear multilevel methods, very few existing algorithms can be employed to solve bound constraint optimization problems. Vallejos proposes a gradient projection algorithm within MG/OPT framework [Val10]. Two variants of the constrained line-search multilevel method for convex optimization problems are also proposed by Kočvara and Mohammed [KM16]. These variants utilize constraint projection rules developed in Reference [HM83] and a variant of the active-set method from Reference [Kor94a]. Gratton et al. propose a variant of the RMTR method for bound constraint non-convex minimization problems [GMTW08] by extending the constraint projection rules developed in Reference [GM90].

In the remaining part of this section, we discuss how to construct a level-dependent feasible set

$$\mathcal{L}^l := \{\boldsymbol{x}^l \in \mathbb{R}^{n^l} \mid \mathbf{vl}^l \leq \boldsymbol{x}^l \leq \mathbf{vu}^l\}, \tag{2.43}$$

where the level-dependent variable bounds $\mathbf{vl}^l, \mathbf{vu}^l \in (\mathbb{R} \cup \{-\infty\} \cup \{\infty\})^{n^l}$ are constructed in a recursive manner. This is achieved using the information about the current

fine-level iterate $\boldsymbol{x}_{\mu}^{l+1}$, and bounds $\mathbf{vl}^{l+1}, \mathbf{vu}^{l+1}$, such that the condition

$$\boldsymbol{x}_{\mu_1}^{l+1} + \mathbf{I}_l^{l+1}(\underbrace{\boldsymbol{x}^l - \mathbf{P}_{l+1}^l \boldsymbol{x}_{\mu_1}^{l+1}}_{\boldsymbol{s}^l}) \in \mathcal{L}^{l+1}$$

holds for all $\boldsymbol{x}^l \in \mathcal{L}^l$. The discussion presented in this section includes the constraint projection rules proposed originally for the RMTR method in Reference [GMTW08]. In addition, we also discuss how to adapt the active-set strategy from Reference [Kor94a] within the RMTR framework.

Remark 8. *We assume that the set $\mathcal{L}^L$ associated with the finest level is the same as the feasible set $\mathcal{F}$ defining the variable bounds of the minimization problem (1.1), i.e., $\mathbf{vl}^L := \mathbf{l}$, and $\mathbf{vu}^L := \mathbf{u}$.*

Projection of the variable bounds

Following References [GM90, GMTW08], we can construct lower bound $\mathbf{vl}^l$ and upper bound $\mathbf{vu}^l$ by defining the level-dependent feasible set $\mathcal{L}^l$ in a component-wise manner. The k-th components of $\mathbf{vl}^l$ and $\mathbf{vu}^l$ are then defined as

$$(\mathbf{vl}^l)_k := (\boldsymbol{x}_0^l)_k + \frac{1}{\tau^l} \max_{j=1,\dots,n^{l+1}} \begin{cases} (\mathbf{vl}^{l+1})_j - (\boldsymbol{x}_{\mu_1}^{l+1})_j, & \text{if } (\mathbf{I}_l^{l+1})_{jk} > 0, \\ (\boldsymbol{x}_{\mu_1}^{l+1})_j - (\mathbf{vu}^{l+1})_j, & \text{if } (\mathbf{I}_l^{l+1})_{jk} < 0, \\ -\infty, & \text{otherwise}, \end{cases} \tag{2.44}$$

and

$$(\mathbf{vu}^l)_k := (\boldsymbol{x}_0^l)_k + \frac{1}{\tau^l} \min_{j=1,\dots,n^{l+1}} \begin{cases} (\mathbf{vu}^{l+1})_j - (\boldsymbol{x}_{\mu_1}^{l+1})_j, & \text{if } (\mathbf{I}_l^{l+1})_{jk} > 0, \\ (\boldsymbol{x}_{\mu_1}^{l+1})_j - (\mathbf{vl}^{l+1})_j, & \text{if } (\mathbf{I}_l^{l+1})_{jk} < 0, \\ \infty, & \text{otherwise}, \end{cases} \tag{2.45}$$

where $\tau^l := \|\mathbf{I}_l^{l+1}\|_\infty$.

Intuitively, the k-th component of the coarse-level variable bound is constructed in two steps. As a first step, we evaluate the maximal distance of the current fine-level iterate $\boldsymbol{x}_{\mu_1}^{l+1}$ from the given fine-level constraint. As a second step, we define the coarse-level variable bound, by preventing the initial coarse-level iterate $\boldsymbol{x}_0^l$ from moving further away than this maximal distance. It is important to notice that the maximal distance is evaluated with respect to all j-th components of the fine-level bound, which are connected with the k-th component of the coarse-level iterate by means of the transfer operator $\mathbf{I}_l^{l+1}$, e.g., $(\mathbf{I}_l^{l+1})_{jk} \neq 0$. This process is demonstrated in Figure 2.9.

The projection rules (2.44) and (2.45) are computationally cheap to evaluate, as they are defined in component-wise manner. However, they provide quite a narrow approximation of the feasible set [Kor94b], which often causes deterioration of the convergence

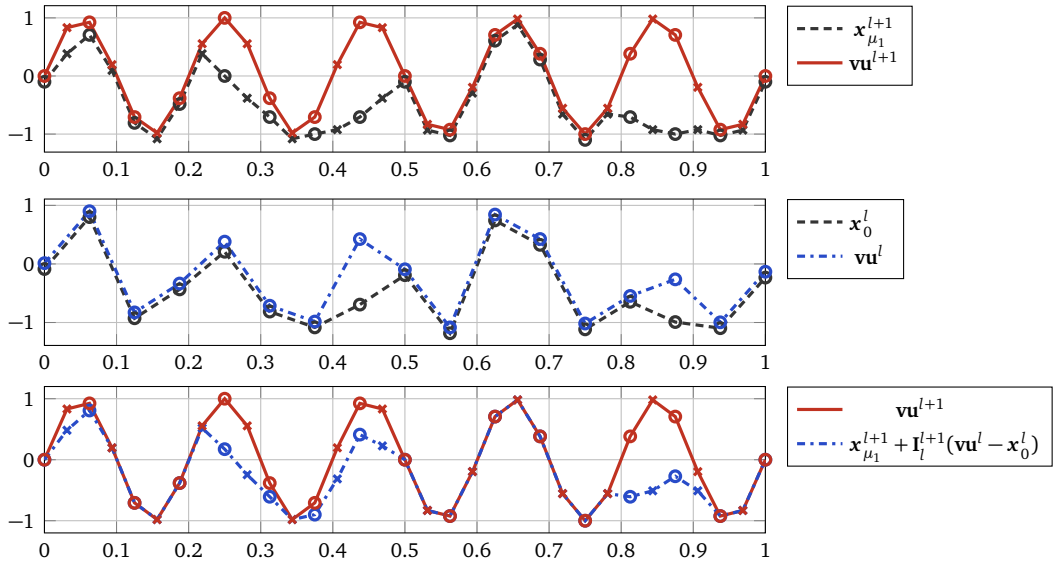

Figure 2.9: Projection of variable bounds for a one-dimensional example. *Top:* A segment of a fine-level upper bound $\mathbf{vu}^{l+1}$ and current fine-level approximate solution $x_{\mu_1}^{l+1}$. *Middle:* Initial coarse-level iterate $x_0^l = \mathbf{P}_{l+1}^l x_{\mu_1}^{l+1}$ and coarse-level upper bound $\mathbf{vu}^l$ obtained using (2.45). *Bottom:* Comparison between the fine-level upper bound $\mathbf{vu}^{l+1}$ and the iterate $x_{\mu_1}^{l+1} + \mathbf{I}_l^{l+1}(\mathbf{vu}^l - x_0^l)$ obtained by assigning the maximal coarse-level correction, given by $\mathbf{vu}^l - x_0^l$. As we can see, the coarse-level upper bound $\mathbf{vu}^l$ is overly restrictive.

rate of the RMTR method. A more advanced technique for constructing the variable bounds can be found in Reference [Kor97, Section 3.1.3]. Unfortunately, this technique is well suited only for the optimization problems with only one type of variable bound (either upper bound or lower bound), therefore, we do not explore it further in this work.

Remark 9. *Evaluation of projection rules* (2.45) *and* (2.44) *in parallel requires special care. This is due to the fact that components of a fine-level constraint required for the construction of a particular component of a coarse-level constraint do not have to belong to the same processor. We can overcome this difficulty by utilizing vectors with ghost elements.*

Example 2.5.1. The constraint projection rules (2.45) and (2.44) appear quite technical at the first glance. However, their design is quite intuitive within the finite element framework. Recall Section 2.3.2, we assume that the FEM method is used to discretize some PDE defined on the domain $\Omega \subset \mathbb{R}^d$, where $d \in \mathbb{N}$. Each level l of the multilevel hierarchy is then associated with mesh $\mathcal{T}^l$ and finite-dimensional space $\mathcal{X}^l$. We assume that the meshes $\{\mathcal{T}^l\}_{l=1}^{l=L}$ and spaces $\{\mathcal{X}^l\}_{l=1}^{l=L}$ are nested. Each FEM space $\mathcal{X}^l$ is spanned by the nodal basis functions $\{N_k^l\}_{k\in\mathcal{N}^l}$, where $\mathcal{N}^l$ denotes a set of interior nodes

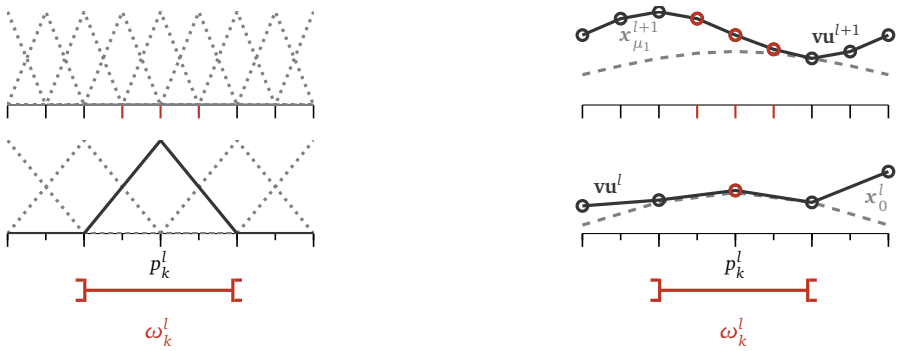

Figure 2.10: One dimensional example of the first-order Lagrange finite elements. Support of basis function N_k^l associated with the node p_k^l defines which components of fine-level upper bound $\mathbf{vu}^{l+1}$ and the solution vector $\boldsymbol{x}^{l+1}$ influence creation of k-th component of the coarse-level upper bound $\mathbf{vu}^l$.

of mesh $\mathcal{T}^l$. The support of a given basis function N_k^l is defined as

$$\omega_k^l = \overline{\{x \in \Omega \mid N_k^l(x) \neq 0\}}.$$

In case of first-order Lagrange finite elements, the support of basis function N_k^l is local and restricted only to the elements in the neighborhood of the k-th node of the mesh $\mathcal{T}^l$, see also Figure 2.10 on the left.

The prolongation operators $\{\mathbf{I}_l^{l+1}\}_{l=1}^{L-1}$ are defined as matrix representation of natural embedding of $\mathcal{X}^l$ into $\mathcal{X}^{l+1}$, see also (2.10). Given this particular settings, all components of $\{\mathbf{I}_l^{l+1}\}_{l=1}^{L-1}$ are non-negative. Thus, the k-th components of $\mathbf{vl}^l$ and $\mathbf{vu}^l$, which are given by (2.44) and (2.45) satisfy

$$\begin{aligned}(\mathbf{vl}^l)_k &:= (\boldsymbol{x}_0^l)_k + \max_{j \in \mathcal{N}^{l+1} \cap \omega_k^l} [(\mathbf{vl}^{l+1} - \boldsymbol{x}_{\mu_1}^{l+1})_j], \\ (\mathbf{vu}^l)_k &:= (\boldsymbol{x}_0^l)_k + \min_{j \in \mathcal{N}^{l+1} \cap \omega_k^l} [(\mathbf{vu}^{l+1} - \boldsymbol{x}_{\mu_1}^{l+1})_j].\end{aligned} \tag{2.46}$$

In other words, the support of the basis function N_k^l associated with k-th node of the mesh $\mathcal{T}^l$ determines, which fine-level components of lower bound $\mathbf{vl}^{l+1}$ and upper bound $\mathbf{vu}^{l+1}$ have to be taken into account while constructing coarse-level variable bounds. Figure 2.10 on the right illustrates this process for a simple one-dimensional example.

Active-set approach

In this section, we present an active-set method for handling the variable bounds. Similarly to Algorithm 4, the presented active-set RMTR method has also a form of V-cycle.

During each V-cycle, the algorithm enters a level l and performs μ_1 pre-smoothing steps. After pre-smoothing is performed, the algorithm detects an active-set

$$\mathcal{A}^l := \{k \in \{1, \dots, n^l\} \mid (\mathbf{vl}^l)_k = (\boldsymbol{x}^l_{\mu_1})_k \text{ or } (\mathbf{vu}^l)_k = (\boldsymbol{x}^l_{\mu_1})_k\}, \tag{2.47}$$

where $\mathbf{vl}^l, \mathbf{vu}^l$ denote lower and upper bounds that define the set $\mathcal{L}^l$, c.f. (2.43). The components of the solution vector $\boldsymbol{x}^l_{\mu_1}$, which belong to the active set $\mathcal{A}^l$, are then held fixed and cannot be altered by the coarser levels. To this end, the level-dependent objective functions $\{h^a\}_{a=1}^{l-1}$ have to be constructed such that their minimization yields coarse-level corrections, which fulfill this requirement.

The construction of suitable level-dependent objective functions depends on the way in which low-cost objective functions $\{f^l\}_{l=1}^L$ are obtained. Here, we restrict ourselves to the same setting as considered in Section 2.3.2, and Example 2.5.1. Thus, each level l of the multilevel hierarchy is associated with mesh $\mathcal{T}^l$ and with the finite-dimensional space $\mathcal{X}^l$, spanned by the basis functions $\{N^l_k\}_{k\in\mathcal{N}^l}$, where $\mathcal{N}^l$ denotes a set of interior nodes of the mesh $\mathcal{T}^l$. In this particular case, each low-cost function f^l is obtained by discretizing an underlying infinite dimensional problem using the basis functions spanning space $\mathcal{X}^l$.

Given these FEM settings, we can employ a truncated basis method [Kor94a] to construct suitable coarse-level models. The truncated basis method constructs truncated FEM spaces $\{\widetilde{\mathcal{X}}^l\}_{l=1}^{L-1}$ by exploiting the fact that the basis functions on level l, can be written as a linear combination of basis functions on level $l+1$, thus as

$$N^l_k = \sum_{p=1}^{n^{l+1}} (\mathbf{I}^{l+1}_l)_{pk} N^{l+1}_p.$$

Each truncated FEM space $\widetilde{\mathcal{X}}^l$ is spanned by truncated basis functions $\{\widetilde{N}^l_k\}_{k\in\mathcal{N}^l}$. Each truncated basis function $\widetilde{N}^l_k$ on level l is constructed, such that its value is zero at all active nodes of finer levels. More precisely, we can construct truncated basis functions in a recursive manner as

$$\widetilde{N}^l_k = \sum_{p=1}^{n^{l+1}} (\widetilde{\mathbf{I}}^{l+1}_l)_{pk} \widetilde{N}^{l+1}_p, \tag{2.48}$$

where $\widetilde{\mathbf{I}}^{l+1}_l$ is truncated prolongation operator defined as

$$(\widetilde{\mathbf{I}}^{l+1}_l)_{pk} = \begin{cases} 0, & \text{if } p \in \mathcal{A}^{l+1}, \\ (\mathbf{I}^{l+1}_l)_{pk}, & \text{otherwise.} \end{cases} \tag{2.49}$$

The truncated prolongation operator $\widetilde{\mathbf{I}}^{l+1}_l$ is obtained from the prolongation operator $\mathbf{I}^{l+1}_l$ by setting p-th row of $\mathbf{I}^{l+1}_l$ to zero, for all $p \in \mathcal{A}^{l+1}$. The use of truncated operator $\widetilde{\mathbf{I}}^{l+1}_l$ in (2.48) removes contributions of basis functions associated with active nodes on level $l+1$, defined by the active set $\mathcal{A}^{l+1}$. Figure 2.11 demonstrates a construction of truncated basis functions for one-dimensional example.

Remark 10. *We construct truncated basis functions* $\{\widetilde{N}_k^{L-1}\}_{k\in\mathcal{N}^{L-1}}$ *on level* $L-1$ *using formula* (2.48) *and the basis functions* $\{N_k^L\}_{k\in\mathcal{N}^L}$ *associated with the finite element space* $\mathcal{X}^L$.

Remark 11. *If all active-sets* $\{\mathcal{A}^l\}_{l=1}^L$ *are empty, then* $\widetilde{\mathcal{X}}^l = \mathcal{X}^l$, *for all* $l = 1,\dots,L$.

Once the truncated coarse-level FEM spaces $\{\widetilde{\mathcal{X}}^l\}_{l=1}^{L-1}$ are created, we can use them to construct level-dependent objective functions and feasible sets. The construction of each the level-dependent feasible set $\mathcal{L}^l$ can be performed using projection rules defined by (2.46). We note that the support of basis functions spanning $\widetilde{\mathcal{X}}^l$ is different from the support of the basis functions spanning $\mathcal{X}^l$. As a consequence, fewer components of a fine-level variable bound are taken into account by (2.46), which leads to less restrictive coarse-level variable bounds. Figure 2.12 demonstrates the difference between coarse-level constraints obtained with and without an active-set strategy.

The construction of level-dependent objective functions using truncated basis functions is not trivial. This is due to the fact that the low-cost functions $\{f^l\}_{l=1}^{L-1}$ were obtained using basis functions spanning spaces $\{\mathcal{X}^l\}_{l=1}^{L-1}$. As a consequence, we have to reconstruct each low-cost function f^l by rediscretizing the underlying infinite dimensional problem using truncated basis functions. Once, the rediscretization is performed and function f^l is created, we can build level-dependent objective function h^l using techniques described in Section 2.4. Here, we point out that the first/second-order coupling terms, such as $\delta\boldsymbol{g}^l$, or $\delta\mathbf{B}^l$ have to be obtained using the truncated transfer operator $\widetilde{\mathbf{I}}_l^{l+1}$, given by (2.49).

The construction of level-dependent objective functions as described above is a tedious task in practice. This is due to the fact that the truncated FEM space $\widetilde{\mathcal{X}}^l$ are altered every time any of the active sets $\{\mathcal{A}^a\}_{a=l+1}^L$ is changed. As a consequence, the function f^l has to be reconstructed fairly often. In addition, we note that the truncated bases are not piecewise linear anymore. Therefore, more sophisticated and expensive quadrature formulas are required in order to evaluate f^l and its derivatives with reasonable accuracy. This makes the truncated basis method impractical, as unconventional modifications to existing FEM software packages have to be made.

Practical algorithm using coarse-level models of Galerkin type A practical implementation of the truncated active-set RMTR method is a challenging task. However, it can be implemented in a simple manner, if the level-dependent objective functions of Galerkin type are employed, recall Section 2.4.2. In this particular case, we are not required to construct spaces $\{\widetilde{\mathcal{X}}^l\}_{l=1}^{L-1}$ explicitly. It suffices to assemble truncated prolongation operators $\{\widetilde{\mathbf{I}}_l^{l+1}\}_{l=1}^{L-1}$ using formula (2.49), given active-sets $\{\mathcal{A}^l\}_{l=2}^L$.

Then, we can create level-dependent objective function $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$ on each level $l < L$ as follows:

$$h^l(\boldsymbol{x}^l) := \langle(\widetilde{\mathbf{I}}_l^{l+1})^T \boldsymbol{g}_{\mu_1}^{l+1}, \boldsymbol{s}^l\rangle + \frac{1}{2}\langle \boldsymbol{s}^l, (\widetilde{\mathbf{I}}_l^{l+1})^T \mathbf{B}_{\mu_1}^{l+1} \widetilde{\mathbf{I}}_l^{l+1})\boldsymbol{s}^l\rangle, \tag{2.50}$$

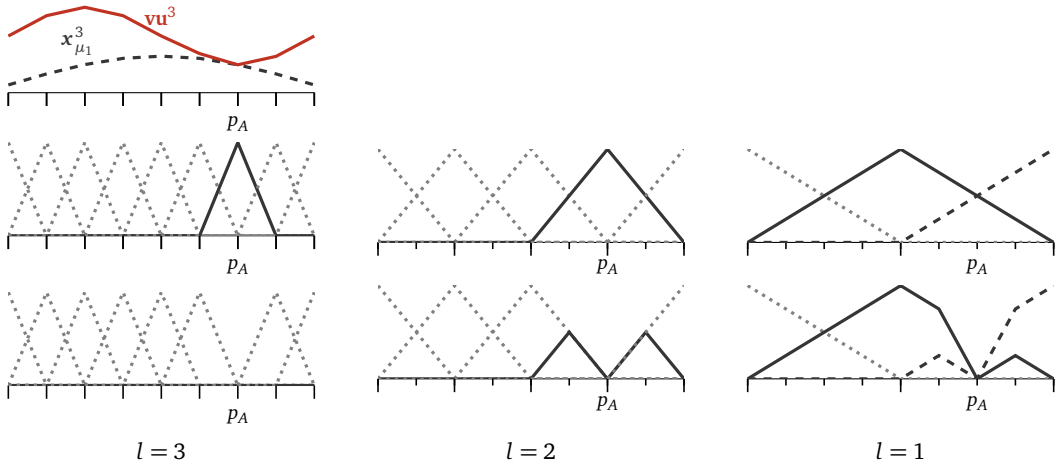

Figure 2.11: *Top:* A segment of a fine-level upper bound $\mathbf{vu}^3$ and current fine-level approximate solution $\boldsymbol{x}^3_{\mu_1}$. The active node associated with the finest level is denoted by the symbol p_A. *Middle:* Nodal basis functions associated with $\mathcal{X}^3$, $\mathcal{X}^2$, and $\mathcal{X}^1$ (from left to right). *Bottom:* Truncated basis functions associated with $\widetilde{\mathcal{X}}^3$, $\widetilde{\mathcal{X}}^2$, and $\widetilde{\mathcal{X}}^1$ (from left to right).

where we used $\widetilde{\mathbf{I}}_l^{l+1}$ to restrict gradient $\boldsymbol{g}^{l+1}_{\mu_1}$ and Hessian $\mathbf{B}^{l+1}_{\mu_1}$ from level $l+1$ to level l. The application of the truncated transfer operator $\widetilde{\mathbf{I}}_l^{l+1}$ in (2.50) removes the components of fine-level gradient/Hessian associated with the active set $\mathcal{A}^{l+1}$. The algorithmic description of the active-set variant of the RMTR method can be found in Algorithm 6. We note, the truncated prolongation operator $\widetilde{\mathbf{I}}_l^{l+1}$ is also used to transfer the correction from level l to $l+1$. This ensures that the components of $\boldsymbol{x}^{l+1}_{\mu_1}$, which were active, do not get modified by the prolongated coarse-level correction.

Example 2.5.2. This example demonstrates the convergence behavior of the RMTR method with and without an active-set strategy by means of two nonlinear constrained minimization problems, namely Membrane, defined in Equation (A.1), and Combustion, defined in Equation (A.2). Both problems are discretized on a two-dimensional uniform grid, using first-order Lagrange finite elements. The performed study considers RMTR configured with six levels and Galerkin type of level-dependent objective functions.

As we can observe from Figure 2.13, using an active-set approach is beneficial, as it allows for significantly faster convergence. We also see that the standard approach, without an active-set strategy, is more effective during the first few V-cycles, but once the exact active-set is detected, the active-set variant of the RMTR method accelerates and converges faster.

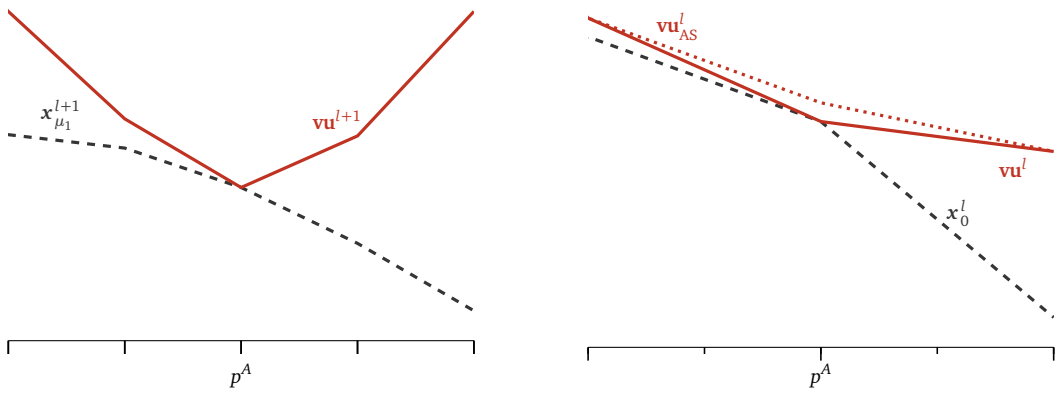

Figure 2.12: *Left:* A segment of a fine-level upper bound $\mathbf{vu}^{l+1}$ and current fine-level approximate solution $\boldsymbol{x}_{\mu_1}^{l+1}$. The active node associated with the finest level is denoted by the symbol p^A. *Right:* Coarse-level upper bounds $\mathbf{vu}_{\text{AS}}^{l}$ and $\mathbf{vu}^{l}$ obtained using (2.46), with and without an active-set strategy, respectively.

2.6 Implementation and numerical considerations

In this section, we discuss some practicalities, which can be used to improve the numerical performance of the RMTR method, described in Algorithm 4. We provide only the implementation details and numerical considerations related to the multilevel nature of the RMTR algorithm. The discussion related to the implementation of the trust-region based algorithms can be found in Section 1.2.2. Furthermore, we point out References [GMS+10, GST06], which provide many helpful insights regarding the implementation of the RMTR method. A general discussion about the factors affecting the performance of multilevel minimization methods can be found in References [NL11, LN06b].

Recursion pattern The convergence speed of nonlinear optimization methods can often be enhanced by providing a good initial guess. In terms of the RMTR method, we can obtain a good initial guess by combining a V-cycle algorithm with the nested iteration [BM+00], which gives rise to a nonlinear F-cycle.

We also note that other forms of recursion, such as W-cycle [BM+00], or free-form recursion [GMS+10] can also be used.

Termination of the recursion Minimization on levels $l < L$ is not always fruitful and might lead to meaningless computations [Gra06, GMTW08, WG10]. For instance, if the gradient of the fine-level objective function $\nabla h^l(\boldsymbol{x}_{\mu_1}^l)$ lies in the null-space of the restriction operator $\mathbf{R}_l^{l-1}$, then the current coarse-level iterate, $\boldsymbol{x}_0^{l-1} := \mathbf{P}_l^{l-1}\boldsymbol{x}_{\mu_1}^l$, appears to be stationary point for h^{l-1}, while $\boldsymbol{x}_{\mu_1}^l$ is not for h^l. Therefore, we abort the recursion

Algorithm 6 RMTR_active_set(l, h^l, $\boldsymbol{x}_0^l$, $\mathcal{F}^l$, Δ_0^l)

Require: $l \in \mathbb{N}$, $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$, $\boldsymbol{x}_0^l \in \mathbb{R}^{n^l}$, $\mathcal{F}^l$, $\Delta_0^l \in \mathbb{R}$
Constants: μ_1, μ_2, $\mu^1 \in \mathbb{N}$
$[\boldsymbol{x}_{\mu_1}^l, \Delta_{\mu_1}^l] = \text{Basic\_TR}(h^l, \boldsymbol{x}_0^l, \mathcal{F}^l, \Delta_0^l, \mu_1)$ ▷ Pre-smooth/ Call Algorithm 1

Construct active-set $\mathcal{A}^l$ by means of (2.47)
Construct $\widetilde{\mathbf{I}}_{l-1}^l$ by means of (2.49) using $\mathbf{I}_{l-1}^l$ and $\mathcal{A}^l$ ▷ Truncate transfer operators
Construct h^{l-1} by means of (2.50) using $\widetilde{\mathbf{I}}_{l-1}^l$ ▷ Initialize coarse-level obj. function
Construct $\mathcal{F}^{l-1}$ by means of (2.38) using (2.46) ▷ Initialize coarse-level feasible set

if $l == 2$ **then**
  $[\boldsymbol{x}_*^l, \_\_] = \text{Basic\_TR}(h^{l-1}, \mathbf{P}_l^{l-1}\boldsymbol{x}_{\mu_1}^l, \mathcal{F}^{l-1}, \Delta_{\mu_1}^l, \mu^1)$ ▷ Solve coarse-level problem
else
  $[\boldsymbol{x}_*^{l-1}, \_\_]$=RMTR_active_set($l-1, h^{l-1}, \mathbf{P}_l^{l-1}\boldsymbol{x}_{\mu_1}^l, \mathcal{F}^{l-1}, \Delta_{\mu_1}^l$) ▷ Call RMTR recursively
end if

$\boldsymbol{s}_{\mu_1+1}^l = \widetilde{\mathbf{I}}_{l-1}^l(\boldsymbol{x}_{\mu_*}^{l-1} - \mathbf{P}_l^{l-1}\boldsymbol{x}_{\mu_1}^l)$ ▷ Prolongate coarse-level correction

$\rho_{\mu_1+1}^l = \frac{h^l(\boldsymbol{x}_{\mu_1}^l) - h^l(\boldsymbol{x}_{\mu_1}^l + \boldsymbol{s}_{\mu_1+1}^l)}{h^{l-1}(\mathbf{P}_l^{l-1}\boldsymbol{x}_{\mu_1}^l) - h^{l-1}(\boldsymbol{x}_*^{l-1})}$ ▷ Compute multilevel TR ratio

$[\boldsymbol{x}_{\mu_1+1}^l, \Delta_{\mu_1+1}^l] = \text{Convergence\_control}(\rho_{\mu_1+1}^l, \boldsymbol{x}_{\mu_1}^l, \boldsymbol{s}_{\mu_1+1}^l, \Delta_{\mu_1}^l)$ ▷ Call Algorithm 2

$[\boldsymbol{x}_*^l, \Delta_*^l] = \text{Basic\_TR}(h^l, \boldsymbol{x}_{\mu_1+1}^l, \mathcal{F}^l, \Delta_{\mu_1+1}^l, \mu_2)$ ▷ Post-smooth/Call Algorithm 1
return $\boldsymbol{x}_*^l$, Δ_*^l

if

$$\mathcal{E}^{l-1}(\boldsymbol{x}_0^{l-1}) < \epsilon_4,$$

where $\epsilon_4 \approx 0$. Here, the symbol $\mathcal{E}^{l-1}$ denotes the first-order criticality measure, as defined by (1.3), with respect to level-dependent feasible set $\mathcal{F}^{l-1}$.

Gradient evaluation and lagging strategies For some problems, the evaluation of a gradient can be very expensive, e.g., in PDE-constrained optimization, when the underlying PDE is time-dependent. To ease the computational burden, we can create a level-dependent objective function h^l using inexact fine-level gradients. For instance, we can employ lagged gradient in order to evaluate the $\delta\boldsymbol{g}^l$ term in (2.20), when constructing level-dependent objective h^l.

The usage of inexact gradients is theoretically supported in the literature. For instance, authors of Reference [LN13] demonstrate, that errors introduced by the evaluation of the gradient on level $l+1$ have only a mild effect on the construction of level-dependent objective function h^l. This is due to the fact that the errors are often damped away by the application of the restriction operator.

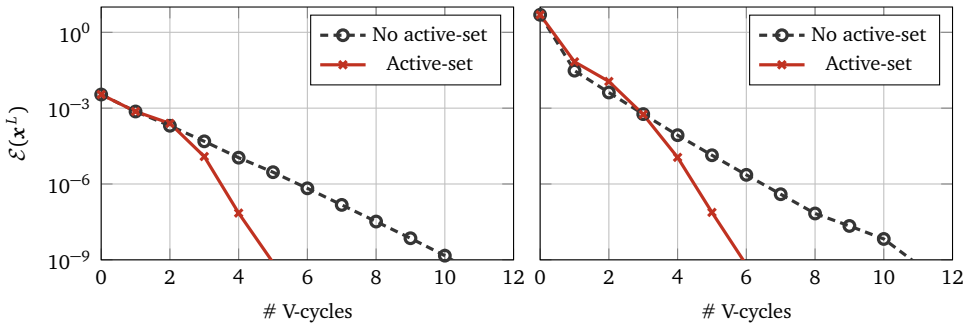

Figure 2.13: The convergence of the RMTR method with (red color) and without (blue color) an active-set strategy. RMTR setup with six levels, Galerkin type of level-dependent functions, and one pre/post-smoothing step. The trust-region subproblems were solved using one iteration of the successive coordinate minimization method. *Left:* Membrane example. *Right:* Combustion example.

Hessian evaluation and lagging strategies The most time-consuming part of the RMTR method is often the evaluation of the Hessian. We can decrease the number of Hessian evaluations by incorporating the Hessian lagging strategies into our implementation. In particular, second-order consistency approaches can employ restricted fine-level Hessian evaluated during the pre-smoothing step in order to evaluate $\delta\mathbf{B}^l$, given by (2.24) or (2.31). Furthermore, we can also skip the Hessian evaluation while performing post-smoothing steps, and employ the Hessian assembled during the pre-smoothing step. In this way, the RMTR method requires only one Hessian evaluation on each level of multilevel hierarchy per V-cycle. We note that these modifications slightly worsen the convergence rate of the RMTR method, but offer 20−30% speed up in terms of the computational time on average [ZKN+20].

2.7 Conclusion

In this chapter, we described the RMTR method in detail. First, the nonlinear multilevel framework was introduced and a generic description of the RMTR algorithm was provided. Then, we identified the three main building blocks of the algorithm and discussed how to construct them in practice. More precisely, we reviewed how to obtain the multilevel hierarchy and transfer operators. We also presented several techniques, which can be employed in order to construct coarse level models. In the end, we discussed how to preserve trust-region and variable bounds in multilevel settings. Furthermore, we proposed an active-set variant of the RMTR method. The resulting method is designed for optimization problems, that arise from the finite element discretization of partial differential equations, and it utilizes the truncated basis functions. The efficiency of the proposed method was demonstrated using two numerical examples.

Chapter 3

UTOPIA: A C++ library of nonlinear multilevel methods

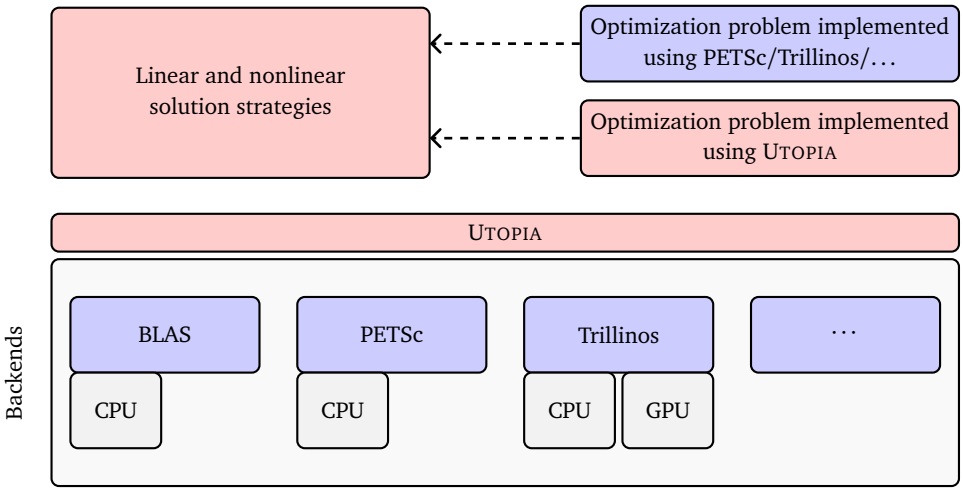

Figure 3.1: Conceptual design of the UTOPIA library. The solution strategies are implemented using the UTOPIA application programming interface and therefore they can use any backend.

We implement solution strategies presented in Chapters 1 and 2 as a part of the open-source C++ library UTOPIA [ZKN+16]. UTOPIA is an embedded domain-specific language (eDSL), designed to make implementations for parallel computing as transparent as possible. UTOPIA enables us to write complex scientific code by using expressions similar to MATLAB [MAT10], whilst it hides the complexity of parallelization and machine-specific optimizations in different backends. As UTOPIA separates the model and the actual linear algebra computations, the generic solution strategies built on top of UTOPIA can be used with any backend. For instance, our implementation of the trust-region algorithm is the

```
Scalar eval_rho(const Vector &s, const Vector &g, const Vector &Bs){
      Scalar rho = dot(g, s) + 0.5*dot(s, Bs);
      return -1.0*rho;
}
```

Figure 3.2: Code snippet, which can be used to evaluate a trust-region ratio using UTOPIA frontend.

same for PETSc [BAA+14] CPU and for the TRILINOS [HBH+05] CPU/GPU backend. The choice of backend is left to the user.

At the moment, UTOPIA provides a uniform interface to the linear algebra libraries BLAS, PETSc and the TRILINOS module Tpetra, see also Figure 3.1. Our implementation of the trust-region methods targets CPU architectures and has been optimized for PETSc backend. In particular, we reduce the overhead of the frontend by exploiting functionalities of the PETSc backend as much as possible. This is achieved by taking advantage of static polymorphism so that no performance-overhead associated with virtual tables is introduced. The particular routines are then specialized by utilizing partial/full specializations. This is beneficial, as we can perform all specializations without checking types at runtime. For example, we can implement evaluation of the trust-region ratio (1.10) using UTOPIA frontend as illustrated in Figure 3.2. The naive frontend evaluation of this expression in parallel necessitates two reductions, associated with the evaluation of two dot products. However, we can specialize the expression, such that it maps to the backend routine, which performs only one reduction by synchronizing dot product evaluations.

3.1 Software design of trust-region methods

Our implementation of trust-region methods follows the design depicted in Figure 3.3. We implement the trust-region algorithm in four separate classes. These classes inherit from `TrustRegionBase` (TR) class, which provides common methods and fields. An actual trust-region algorithm is then implemented inside of following classes:

- `TR_2` (TR_2) / `MF_TR_2` (Matrix-free TR_2) An implementation/ matrix-free implementation of the trust-region method with trust-region radius defined by the ℓ_2-norm. The implemented algorithm is suitable only for unconstrained optimization problems.
- `TR_inf` (TR_∞)/ `MF_TR_inf` (Matrix-free TR_∞) An implementation/ matrix-free implementation of the trust-region method with trust-region radius defined by the ℓ_∞-norm. The implemented algorithm is suitable for constrained and unconstrained optimization problems.

The trust-region methods are designed and implemented, such that they can be integrated with a wide range of existing user application codes without sacrificing the performance. This is possible as UTOPIA provides a balance between abstraction and

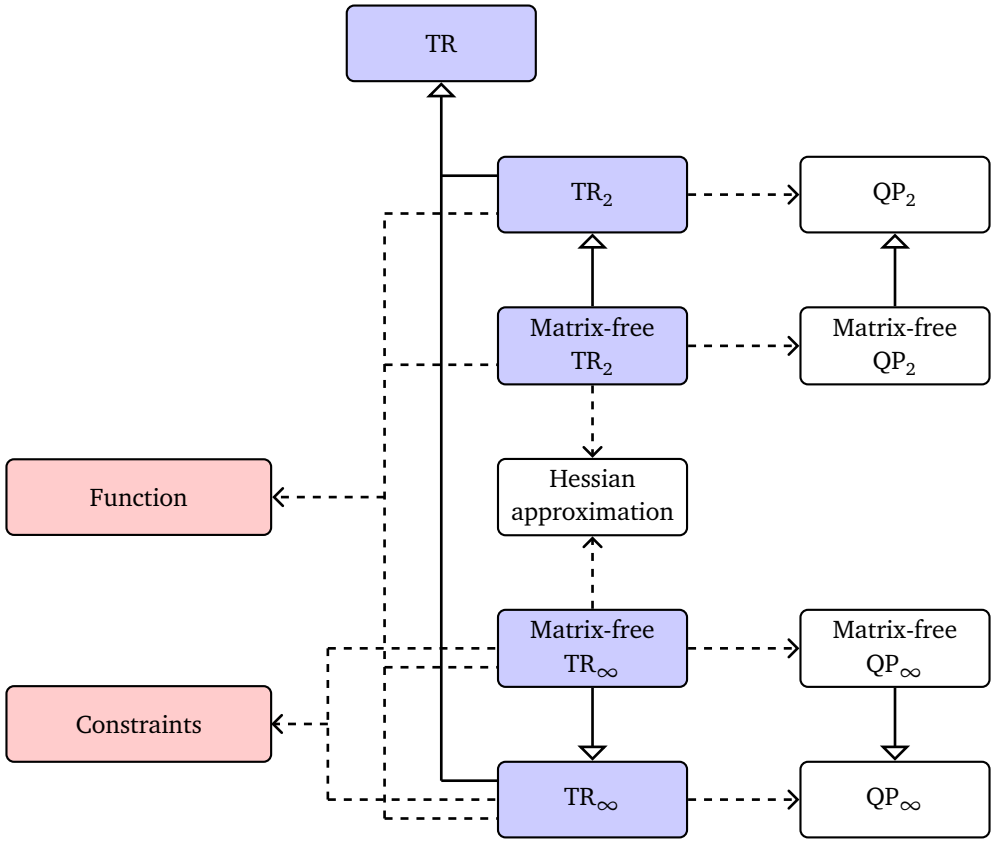

Figure 3.3: Class diagram of single-level trust-region software. Dashed arrows represent the association and solid arrows represent inheritance. Blue color denotes the implementations of the TR method. Red color symbolizes user-provided context. White color indicates objects which can be customized by the user.

low-level data access. As a consequence, the user application, e.g. FEM simulation implemented using an external library, can utilize the trust-region method implemented on top of UTOPIA. Interoperability is ensured by enforcing uniform interfaces between optimization problem context and UTOPIA solvers. In particular, an implementation of the optimization problem (objective function, and its derivatives) has to follow interfaces provided by the class `Function` (Function), see also Figure B.1 for details. In the case of constrained optimization problems, the class `BoxConstraints` (Constraints), has to be used to encapsulate the description of lower and upper bounds.

The modular design of the trust-region solvers allows users to customize the solution strategy, such that it suits their application. More precisely, users can choose which trust-region subproblem solver to use. In the case of matrix-free implementations, a choice of suitable Hessian approximation scheme can be selected. The routines, which mimic the application of Hessian and its inverse to a vector have to be imposed by follow-

```
RMTR_inf<Matrix, Vector, TRBoundsGR_VRBoundsGM<Matrix, Vector>,
                         SECOND_ORDER_ADDITIVE> rmtr_inf(n_levels);

RMTR_2<Matrix, Vector,  MF_FIRST_ORDER_MULTIPLICATIVE> rmtr2(n_levels);
```

Figure 3.4: Instantiation of the `RMTR_inf` and `RMTR_2` classes. The `RMTR_inf` class is setup with second-order additive level-dependent functions and uses rules (2.41) and (2.44) to project trust-region and variable bounds to coarse levels, respectively. The `RMTR_2` class is setup with first-order multiplicative level-dependent objective functions, optimized for matrix-free implementation.

ing interfaces provided by the `HessianApproximation` (Hessian approximation) class. UTOPIA currently provides the implementation of the L-BFGS and L-SR1 secant methods. Figure B.2 demonstrates an example of solving an optimization problem using `MF_TR_2` class. In this particular example, the preconditioned STCG method is used to solve the resulting trust-region subproblems, while the L-BFGS method is used to approximate the Hessian.

Remark 12. *Matrix-free implementations do not necessarily rely on limited-memory secant methods. Indeed, any technique, which evaluates the application of a Hessian to a vector can be used.*

Trust-region subproblem solvers and static polymorphism

We developed several trust-region subproblem solvers directly using UTOPIA. Hence, they can be used in conjunction with any backend. However, we also aim to reuse as many existing implementations as possible by interfacing external solvers, provided for instance by PETSC's KSP or TAO. We utilize multiple implementations of the same subproblem solver by exploiting meta-programming facilities of the C++ language. The particular implementation is then invoked at the compile time. For instance, the `SteihaugToint` class may provide different implementations when specialized for PETSC or for TRILINOS backend.

We note that only a few implementations of trust-region subproblem solvers are currently available within PETSC or TRILINOS library. Table B.1 provides a brief overview of the subproblem solvers currently available within the UTOPIA library.

3.2 Software design of multilevel trust-region methods

Our implementation of the RMTR methods follows the design shown in Figure 3.5. We implement the RMTR method in four separate classes:

- `RMTR_2` (RMTR_2) / `MF_RMTR_2` (Matrix free RMTR_2) An implementation/ matrix-free implementation of the RMTR method, with TR radius defined by the ℓ_2-norm. Implementation is suitable for unconstrained optimization problems.

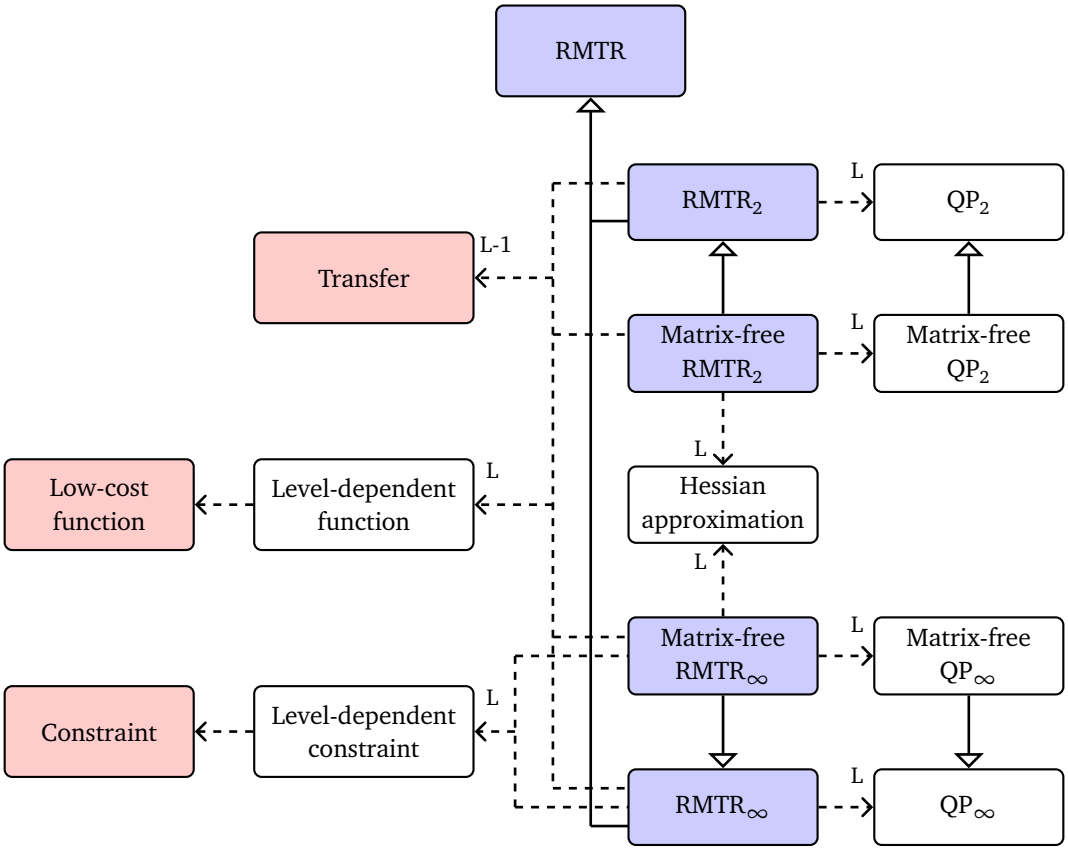

Figure 3.5: Class diagram of multilevel trust-region software. Dashed arrows represent the association and solid arrows represent inheritance. Blue color denotes the implementations of the RMTR method. Red color symbolizes user-provided context, while white color depicts objects, which can be customized by the user.

- `RMTR_inf` (RMTR$_\infty$) / `MF_RMTR_inf` (Matrix-free RMTR$_\infty$) An implementation/ matrix-free implementation of the RMTR method, with TR radius defined by the ℓ_∞-norm. Implementation is suitable for constrained and unconstrained optimization problems.

All the aforementioned classes inherit from `RMTRBase` (RMTR) class, which provides common functionalities and fields. In order to obtain a desirable variant of the RMTR method, all RMTR classes have to be specialized by the user and configured at the compile time. Specialization is necessary in order to determine which type of level-dependent objective functions should be employed. In the case of RMTR with a trust-region radius defined by the ℓ_∞-norm, it is also necessary to specify how to treat constraints in multilevel settings. Given these specializations, the RMTR then employs routines, which were developed and optimized specifically for a given configuration. Figure 3.4 demonstrates an example of instantiation of the `RMTR_2` and `RMTR_inf` class. Tables B.2 and B.3

```
auto qp_solver = std::make_shared<utopia::MPGRP<Matrix, Vector>>();
const SizeType p = 2;
auto tr_strategy_coarse = std::make_shared<RedundantQPSolver<Matrix, Vector>>(qp_solver, p);
```

Figure 3.6: Code snippet, which can be used to instantiate the redundant trust-region subproblem solver, with p redundant solves of the MPGRP method.

provide summary of currently available specializations for defining the level-dependent objective functions and multilevel treatment of constraints, respectively.

The RMTR is also implemented, such that it can be used by external application codes. Interoperability is ensured by enforcing uniform interfaces between the RMTR and optimization problem context. More precisely, the user has to provide an implementation of the low-cost objective functions (including the finest level). Each low-cost objective function has to follow common interface provided by the class `Function` (Low-cost function). In addition, knowledge about transfer operators is required. Here, the interfaces provided by the class `Transfer` (Transfer) must be followed. In the case of constrained optimization problems, the variable bounds of the finest level have to be supplied by means of class `BoxConstraints` (Constraints).

Besides, the user can specify which trust-region subproblem solver and Hessian approximation scheme (if applicable) should be used on each level of the multilevel hierarchy. To achieve good scaling properties of the RMTR method, it is often necessary to reduce the communication cost of the coarsest level. We can reduce the communication cost required while solving the trust-region subproblems by introducing a `Redundant` solver class. This class replicates and solves trust-region subproblems on p subgroups of processors. For instance, if we execute code with 10 MPI processes, `Redundant` class creates and solves p trust-region subproblems. Each subproblem will be solved on $10/p$ processes. Figure 3.6 demonstrates an example of setting up `Redundant` MPGRP solver.

The communication cost of the coarser levels could be further decreased by reducing the number of processors used by the assembly routines. We plan to explore this possibility in future work.

3.3 Conclusion

In this chapter, we provided a brief overview of the open-source C++ library UTOPIA. In particular, we discussed a design of the trust-region module, which incorporates an implementation of the single-level and multi-level trust-region methods described in Chapters 1 and 2. A summary of currently available functionalities was provided and the user interfaces were described. In the end, we also demonstrated how to solve the custom optimization problem using UTOPIA trust-region and multilevel trust-region solvers.

Part II

Applications

Chapter 4

Phase-field fracture simulations

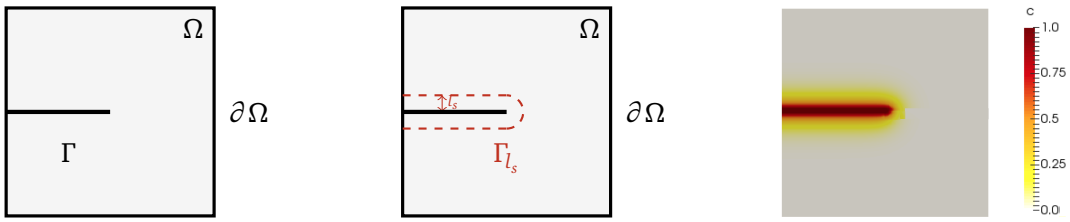

Figure 4.1: *Left:* Computational domain Ω with internal discontinuity/fracture Γ. *Middle:* Phase-field regularization of fracture surface, denoted by Γ_{l_s}. Size of the regularized zone is controlled by the length-scale parameter l_s. *Right:* Result of phase-field fracture simulation. Damage parameter c indicates fracture zone.

Predicting damage, crack and fragmentation patterns is a long-lasting challenge in computational mechanics. The phase-field approach to fracture addresses this task in an elegant way and has, therefore, become very popular. The basic idea behind the phase-field approach is to regularize sharp crack interfaces in the structure by means of smeared, volumetric crack zones. This removes the need to represent discontinuities explicitly and, therefore, avoids the requirement of the remeshing. The volumetric approximation of a crack surface is based on the Ambrosio-Torelli functional [AT90], which employs an auxiliary damage variable. This damage variable acts as an indicator, which marks the state of the material from intact to fully broken. The transition between the broken and intact parts of the domain is modeled in a diffuse manner. The size of the transition zone is controlled by a length-scale parameter.

Several approaches to phase-field fracture can be found in the literature. In the physics community, evolution equations are derived by adapting the phase transition formalism of Landau and Ginzburg [LL80]. In the mechanics community, phase-field fracture approaches are based on the variational formulation of the brittle fracture

The content of this chapter is extracted from published Manuscript [KK20b].

developed by Francfort and Marigo [FM98], which provides an extension to classical Griffiths theory of fracture [Gri21]. The first numerical implementation of the variational phase-field approach is presented by Bourdin et al. in Reference [BFM00]. Miehe et al. [MWH10, MHW10] modify the underlying mathematical model by introducing a rate-independent formulation that ensures the local growth of the damage variable. Their formulation also introduces an anisotropic energy functional, which allows for the degradation of the tensile energy with increased damage. Those modifications result in a thermodynamically consistent phase-field formulation of brittle fracture. Since then the phase-field approach has become popular and was extended in various directions, such as dynamics [BLR11, SKM14], large deformations [DPLM07, HGO+17, HW14], cohesive fracture [VMDBV14, CFI16], shells and plates [AMS+14, ADL16], and used within several multiphysics applications [BKKW17, HM17, AA12].

An important feature of any phase-field fracture model is to ensure crack irreversibility [HA09, Gia05], which introduces a lower bound on the phase-field variable. The resulting constraint can be treated directly in the solution strategy, for example by using active set [FJS98], or semi-smooth Newton methods [Ulb11]. In order to simplify the solution process, alternatives, based on modified energy, or weak formulations, are widely considered in the literature. For example, Bourdin et al. [BFM00] transforms inequality constraints into equality constraints by using a penalization term from [AFMP15]. A strategy based on augmented Lagrangian penalization is introduced by Wheeler et al. [WWW14]. Miehe et al. [MHW10] handle the irreversibility by introducing a so-called history variable, which replaces the maximal tensile energy accumulated within the loading process. Disposing of the inequality constraints usually causes deterioration of the conditioning of the Hessian matrix [NW06], or leads to inconsistencies between the original energy functional and the modified weak form.

In this work, the phase-field formulation of fracture is based on the variational framework. The fracture problem is modeled using an energy functional, whose minimizer corresponds to the associated solution. Algorithmically, the minimization process can be realized by using monolithic or staggered solution schemes. In the monolithic approach, the energy functional is minimized simultaneously for displacements and phase-field [VMDBV14]. This is numerically challenging, as the energy functional of the coupled problem is non-convex. In contrast, using a staggered solution scheme, the energy functional is minimized individually for displacements and phase-field. This gives rise to two convex minimization problems, which makes staggered solution scheme very robust and popular in the engineering community. However, it has been shown in Reference [GL16], that if convergent, the monolithic scheme can be faster than the staggered scheme since the staggered scheme tends to underestimate crack speed. Therefore, it is an important and challenging task to design efficient and reliable solution strategies for a fully monolithic scheme.

To our knowledge, there are only a few published results concerning the monolithic framework. These advances were accomplished either by modifying the under-

lying weak formulations or by applying some globalization strategy during the minimization process. Vignollet et al. [VMDBV14] augment the weak form by using a dissipation-based arc-length procedure [VRG09]. Heister et al. apply a "convexification" trick, where the phase-field variable in the critical terms is replaced by its extrapolation in time [HWW15]. Gerasimov et al. employ a line-search based Newton's method, which takes into account negative curvature [GL16]. Globalization of Newton's method, based on an error oriented approach from Reference [Deu11], is successfully applied by Wick [Wic17a]. In Reference [Wic17b], Wick employs a so-called modified Newton's method, which is based on dynamically switching between full Newton and modified Newton steps. The modified Newton steps are based on an altered Hessian matrix, where coupling terms are scaled in an appropriate manner.

All aforementioned globalizations of Newton's method require the solution of a system of linear equations in order to determine the step direction. Solving those linear systems can become computationally demanding as they are usually very large and ill-conditioned, see Section 4.1. An ideal choice of the linear solution strategy is a multilevel method due to its optimal complexity and scalability. In the context of phase-field fracture models, an algebraic multilevel method is applied by Farrell and Maurini [FM17]. The authors of Reference [FM17] employ smoothed aggregation in order to precondition systems related to the displacement and the phase-field subproblems. Also within the staggered solution scheme, the semi-geometric multigrid method was applied by Bilgen et al. [BKKW18]. The matrix-free linear multigrid method was proposed in Reference [JLW20], which demonstrated good convergence properties and scalability up to 128 cores. An alternative approach, based on truncated non-smooth nonlinear monotone multigrid, was taken in References [KGSK18, GKS20], where the authors obtained a significant improvement in terms of computational time, but the parallel performance was not reported. To the best of our knowledge, a fully nonlinear parallel multilevel solution strategy has yet not been applied to the coupled systems arising in the monolithic framework.

4.1 Phase-field fracture model

In this section, we briefly review the variational formulation of the second-order phase-field model for brittle fracture. Our presentation focuses on quasi-static fracture problems and neglects inertia effects. A pseudo-time $t \in [0, T]$, where $T > 0$ is some final time, serves for indexing the deformation state of the loading process.

Let $\Omega \subset \mathbb{R}^d, d = 2, 3$ be the computational domain with internal fracture surface $\Gamma \subset \mathbb{R}^{d-1}$. The computational domain Ω is assumed to have Lipschitz boundary $\partial\Omega$, which is decomposed into the Dirichlet boundary $\partial\Omega_D$ and the Neumann boundary $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$. The variational approach to brittle fracture predicts the crack

evolution by minimizing the total potential energy of the body Ω, given by

$$\Psi(\mathbf{u}) := \underbrace{\int_{\Omega} \psi_e(\boldsymbol{\varepsilon}(\mathbf{u}))\, d\Omega}_{\text{elastic energy}} + \underbrace{\int_{\Gamma} \mathcal{G}_c\, d\Gamma}_{\text{fracture energy}}. \tag{4.1}$$

The energy functional (4.1) incorporates elastic energy and fracture energy. In this work, the elastic energy describes small deformations and utilizes the linearized strain tensor $\boldsymbol{\varepsilon}(\mathbf{u}) = 0.5(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$, where $\mathbf{u} : \Omega \to \mathbb{R}^d$ is the displacement at given pseudo-time t.

The term describing the fracture energy originates from the classical theory of brittle fracture [Gri21, Flü13]. It states that the energy required to create a unit area of fracture surface equals the critical energy release $\mathcal{G}_c \in \mathbb{R}$. As a consequence, the fracture energy is defined as a contribution of $\mathcal{G}_c$ integrated over the fracture surface Γ. The presence of a fracture surface integral in (4.1) requires precise tracking of the crack surface paths and necessitates tedious remeshing, which leads to computationally exhausting algorithms.

4.1.1 Energy formulation

The phase-field approach to fracture [BFM08] overcomes difficulties related to the remeshing of the crack surface Γ by introducing the phase-field variable $c : \Omega \to [0, 1]$, which characterizes the material state from unbroken $c = 0$ to fully broken $c = 1$ and "damaged" in between. The regularization of the fracture is then performed by replacing the fracture surface integral in (4.1) by its volumetric approximation

$$\int_{\Gamma} \mathcal{G}_c\, d\Gamma \approx \underbrace{\int_{\Omega} \frac{\mathcal{G}_c}{c_w} \frac{w(c)}{l_s}\, d\Omega}_{\text{reaction term}} + \underbrace{\int_{\Omega} \frac{\mathcal{G}_c}{c_w} l_s\, |\nabla c|^2\, d\Omega}_{\text{diffusion term}}, \tag{4.2}$$

where function w defines a decaying profile of the phase-field c, while c_w is an induced normalization constant, defined as $c_w := 4\int_0^1 \sqrt{w(c)}\, dc$. Approximation (4.2) takes into account the phase-field c, its spatial gradient ∇c and the regularization/length-scale parameter $l_s \in \mathbb{R}^+$. The reaction term in (4.2) corresponds to the area of the region, where the material is damaged, e.g., $c \gg 0$. The diffusion term describes the area of the transition between the broken and intact part of the domain, where the gradient is high. The length-scale parameter l_s controls the thickness of the damaged region by weighting the contributions of the reaction and the diffusion terms. Increasing l_s makes the crack zone more diffused, while $l_s \to 0$ gives rise to a sharp crack surface Γ. The effect of the length scale parameter l_s is demonstrated in Figure 4.2.

Taking into account (4.2), the total potential energy from (4.1) can be reformulated

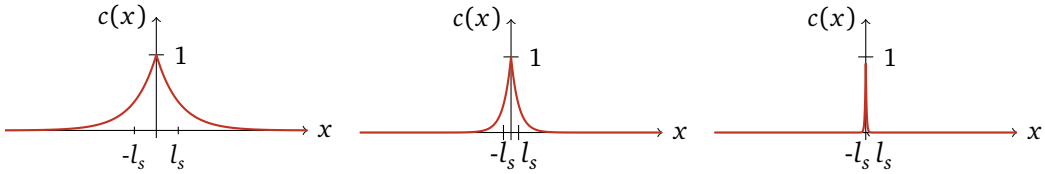

Figure 4.2: One dimensional phase-field approximation to the crack surface at $x = 0$. As $l_s \to 0$, the crack approximation approaches the sharp crack surface. Crack approximations with parameter l_s set to 0.3, 0.1, 0.05 (from left to right).

as follows

$$\Psi(\mathbf{u}, c) := \underbrace{\int_\Omega \big(g(c)(1-k)+k\big)\psi_e^+(\boldsymbol{\varepsilon}(\mathbf{u})) + \psi_e^-(\boldsymbol{\varepsilon}(\mathbf{u}))\, d\Omega}_{\text{elastic energy}} + \underbrace{\int_\Omega \frac{\mathcal{G}_c}{c_w}\left(\frac{w(c)}{l_s} + l_s|\nabla c|^2\right) d\Omega}_{\text{fracture energy}}, \tag{4.3}$$

where g is a degradation function, which accounts for the loss of stiffness in the fracture. It should be noted that the degradation function g acts only on the tension part of the elastic energy. This is due to the fact, that in reality, fracture develops only in the direction of tension [BFM00]. The linear isotropic material models do not capture this feature and allow for the fracture to occur in both tension and compression. In order to avoid unrealistic crack patterns, anisotropic material models have to be employed [HA09, AGDL15]. We employ an anisotropic model introduced by Miehe et al. [MWH10], where $\psi_e(\boldsymbol{\varepsilon}(\mathbf{u}))$ is additively decomposed into $\psi_e^+(\boldsymbol{\varepsilon}(\mathbf{u}))$ and $\psi_e^-(\boldsymbol{\varepsilon}(\mathbf{u}))$, which correspond to the tension and compression, respectively. For more details about the employed energy split, we refer to Appendix A.3.1.

Several choices of g, w and c_w are used in the literature, leading to various phase-field fracture formulations [AMM09, PAMM11, BOS13, KSM15, SKBN18]. Here, we follow References [BFM00, MWH10] and employ $g(c) := (1-c)^2$, $w(c) = c^2$ and $c_w = 2$, resulting in the widely used *AT-2* phase-field fracture model [AT90]. Given these particular choices, it is possible to asymptotically show via Γ-convergence, that the minimizer of (4.3) tends to a minimizer of (4.1), as $l_s \to 0$, see Reference [Gia05] for more details.

Remark 13. *For simplicity, Formulation (4.3) neglects body forces and surface tractions. The artificial parameter $k \approx \mathcal{O}(\epsilon)$ in (4.3) circumvents a full degradation of energy at the completely broken state and helps to improve the condition number of the numerical system [AT90].*

4.1.2 Minimization problem

At each loading step t, the displacement $\mathbf{u}$ and phase-field c are characterized as a minimizer of the following minimization problem:
Find $(\mathbf{u}, c) \in \boldsymbol{V}^t \times H^1(\Omega)$, such that

$$\begin{aligned} (\mathbf{u}, c) &:= \text{ argmin } \Psi(\mathbf{u}, c), \\ &\text{subject to } \partial_t c \geq 0 \,, \end{aligned} \tag{4.4}$$

where $\boldsymbol{V}^t := \{\mathbf{u} \in \boldsymbol{H}^1(\Omega) \mid \mathbf{u} = \boldsymbol{g}^t \text{ on } \partial\Omega_D\}$ is an admissible space for the displacement field. The symbol H^1 denotes the standard Sobolev space of weakly differentiable functions in L^2 with one weak derivative also in L^2. Since the minimization problem (4.4) is quasi-static, the time t enters the above formulation only via the prescribed Dirichlet boundary condition $\boldsymbol{g}^t$ and the time-dependent irreversibility constraint $\partial_t c \geq 0$. This irreversibility constraint ensures a positive evolution of the phase-field and prevents the crack from healing. Discretization of $\partial_t c \approx \frac{c^t - c^{t-1}}{\Delta t}$, with Δt denoting the time-step size, allows for the reformulation of the minimization problem (4.4) as follows:
Find $\boldsymbol{U} = (\mathbf{u}, c) \in \boldsymbol{V}^t \times H^1(\Omega)$ such that

$$\begin{aligned} \boldsymbol{U} &:= \text{ argmin } \Psi(\boldsymbol{U}), \\ &\text{subject to } \boldsymbol{U} \geq \boldsymbol{U}^{\boldsymbol{t-1}} \quad \text{on } \Phi, \end{aligned} \tag{4.5}$$

where $\boldsymbol{U}^{\boldsymbol{t-1}}$ represents the solution in the previous time-step and $\Phi = \mathbf{0} \times H^1(\Omega)$. In this way, the inequality constraint acts only on the phase-field variable c.

We can solve the minimization problem (4.5) by using a monolithic [MWH10] or a staggered solution scheme [MHW10]. The monolithic scheme requires the solution of the coupled problem in each time-step, see Algorithm 7. Solving the arising nonlinear problems is numerically challenging, as the energy functional (4.3) is non-convex, see also Section 4.1.2 for a detailed discussion. In contrast, the staggered solution scheme minimizes (4.5) separately with respect to the displacements and the phase-field, while the other variable is held fixed, see Algorithm 8. This gives rise to two convex minimization problems on each iteration step. From the algorithmic point of view, it is easier to design reliable solution strategies for the decoupled minimization problems. Therefore, this scheme has become very popular and is often preferable in practical applications. Its main drawback is a slow convergence rate. Hence, our motivation is to design a reliable solution strategy for the non-convex minimization problems arising in the monolithic scheme.

Properties of the coupled minimization problem

The first-order necessary conditions of coupled minimization problem (4.5) have the following form:

Algorithm 7 Monolithic scheme

Input: Initial guess $(\mathbf{u}^0, c^0)$
for t $= 0, \cdots, T$ **do**
  Compute displacement and phase-field:
  $(\mathbf{u}^t, c^t) := \underset{(\mathbf{u},c) \in \boldsymbol{V}^t \times H^1(\Omega)}{\arg\min} \Psi(\mathbf{u}, c),$
  subject to $c \geq c^{t-1}$,
end for

Algorithm 8 Staggered scheme

Input: Initial guess $(\mathbf{u}_0^0, c_0^0)$
for t $= 0, \cdots, T$ **do**
  Set $k = 1$
  while not converged **do**
    Compute phase-field:
    $c_{k+1}^t := \underset{c \in H^1(\Omega)}{\arg\min} \Psi(\mathbf{u}_k^t, c),$
    subject to $c \geq c_0^t$,

    Compute displacement field:
    $\mathbf{u}_{k+1}^t := \underset{\mathbf{u} \in \boldsymbol{V}^t}{\arg\min} \Psi(\mathbf{u}, c_{k+1}^t),$
    $k = k + 1$
  end while
  $(\mathbf{u}_0^{t+1}, c_0^{t+1}) = (\mathbf{u}_k^t, c_k^t)$
end for

Find pair $(\mathbf{u}, c) \in \boldsymbol{V}^t \times H^1(\Omega)$, such that

$$\begin{aligned} \nabla_{\mathbf{u}} \Psi(\mathbf{u}, c; \boldsymbol{v}) &= 0, && \forall \boldsymbol{v} \in \boldsymbol{H}_0^1(\Omega), \\ \nabla_c \Psi(\mathbf{u}, c; w) &\geq 0, && \forall w \in W^t \cap L^\infty(\Omega), \end{aligned}$$

where $W^t := \{c \in H^1(\Omega) \mid 0 \leq c^{t-1} \leq c \leq 1 \text{ a.e. on } \Omega\}$. The symbol c^{t-1} denotes the solution of the phase-field on the previous time-step. The directional derivatives of $\Psi(\mathbf{u}, c)$ with respect to $\mathbf{u}$ and c, are denoted by $\nabla_{\mathbf{u}}\Psi$ and $\nabla_c \Psi$, respectively. They are given as

$$\begin{aligned} \nabla_{\mathbf{u}} \Psi(\mathbf{u}, c; \mathbf{v}) &:= \int_\Omega \left(\left(g(c)(1-k) + k \right) \frac{\partial \psi_e^+}{\partial \boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}(\mathbf{u})) + \frac{\partial \psi_e^-}{\partial \boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}(\mathbf{u})) \right) : \boldsymbol{\varepsilon}(\mathbf{v}) \, d\Omega, \\ \nabla_c \Psi(\mathbf{u}, c; w) &:= \int_\Omega \left(\nabla_c g(c)(1-k) + k \right) \psi_e^+(\boldsymbol{\varepsilon}(\mathbf{u})) \, w d\Omega + \int_\Omega \mathcal{G}_c \left(\frac{1}{l_s} c w + l_s \nabla c \nabla w \right) d\Omega. \end{aligned}$$

We solve the minimization problem (4.5) numerically by discretizing the underlying partial differential equations using the finite-element method. Solving this problem is numerically challenging for the following reasons:

- The functional (4.5) is non-convex and, therefore, its minimization may admit many local minimizers. The most critical term is $\left(g(c)(1-k) + k\right)\psi_e^+(\boldsymbol{\varepsilon}(\mathbf{u}))$ which

is well known to be non-convex in both variables $\mathbf{u}$ and c [Wic17b]. The most popular solution strategy for solving nonlinear equations is Newton's method, as it is simple and locally quadratically convergent [NW06]. However, convergence properties of the standard Newton's method rely on a good initial guess. In the context of phase-field fracture, Newton's method typically fails in the case of a "brutal" crack evolution and in the post-peak regime [GL16].

- Resolving the regularized crack surface Γ_{l_s} demands a fine mesh with a high spatial resolution. In fact, the mesh size h has to fulfill $h \leq \frac{l_s}{2}$. Here, l_s is the length-scale parameter from (4.2) controlling the width of the diffusive crack zone. As a consequence, we end up with large-scale coupled nonlinear systems of equations, which are computationally demanding to solve.
- The arising systems of equations might also be ill-conditioned as the presence of the crack leads to strongly varying elastic stiffness.
- The irreversibility condition defines a lower bound on the phase-field variable c. Therefore, any employed solution strategy has to take into account point-wise constraint.

4.2 Recursive multilevel trust-region method

We address all aforementioned points by employing the RMTR method with the trust-region defined by the ℓ_∞-norm [GMTW08, GK09c]. On one side, the multilevel framework addresses the ill-conditioning and a large number of degrees of freedom (dofs). On the other side, the trust-region framework tackles the non-convexity of the energy functional. In addition, the trust-region method with the trust-region defined by the ℓ_∞-norm allows for natural treatment of the point-wise inequality constraints [CGT00].

Our implementation of the RMTR method for phase-field fracture simulations follows the discussion provided in Chapter 2. Here, we provide additional details about our construction of the multilevel hierarchy and the transfer operators, the level-dependent objective functions, and the level-dependent feasible sets.

4.2.1 Multilevel hierarchy and transfer operators

Recall Section 2.2, the RMTR method requires a set of auxiliary functions $\{f^l\}_{l=1}^L$. In this chapter, we obtain the hierarchy of functions $\{f^l\}_{l=1}^L$ by discretizing the optimization problem (4.5) with an increasing accuracy. To this aim, we create a multilevel hierarchy of nested approximation/finite element spaces $\{\mathcal{X}^l\}_{l=1}^L$, from a hierarchy of nested meshes $\{\mathcal{T}^l\}_{l=1}^L$. Here, we assume that the meshes $\{\mathcal{T}^l\}_{l=2}^L$ are obtained by uniformly refining an initial coarse-level mesh $\mathcal{T}^1$ and that $\dim(\mathcal{X}^l) > \dim(\mathcal{X}^{l-1})$.

In order to transfer data between subsequent levels of the multilevel hierarchy, we employ pseudo-L^2-projection operators [Woh00]. The pseudo-L^2-projection belongs to the more generic group of mortar/variational transfer operators, which are commonly

used in the context of non-conforming domain decomposition methods [Bel99] and contact problems [WK03]. The mortar projections are designed to transfer the information of functional quantities from one approximation space to another. Following the naming convention in the literature, the data is transferred from a master space to a slave space. Therefore, throughout the following, we use subscripts m and s to denote quantities related to the master and slave side, respectively.

Variational transfer

Let us consider finite element spaces $\mathcal{X}^m$ and $\mathcal{X}^s$ associated with meshes $\mathcal{T}^m$ and $\mathcal{T}^s$, respectively. The mortar projection operator $\Pi : \mathcal{X}^m \to \mathcal{X}^s$ transfers a function from the master space $\mathcal{X}^m$ to the slave space $\mathcal{X}^s$. Thus, for a function $m_h \in \mathcal{X}^m$, we are looking for $s_h = \Pi(m_h) \in \mathcal{X}^s$, such that the following equality conditions hold

$$\begin{aligned} (m_h, q_h)_{L_2(\mathcal{I})} &= (\Pi(m_h), q_h)_{L_2(\mathcal{I})}, \qquad && \forall q_h \in \mathcal{X}^q, \\ (m_h, q_h)_{L_2(\mathcal{I})} &= (s_h, q_h)_{L_2(\mathcal{I})}, && \forall q_h \in \mathcal{X}^q, \end{aligned} \tag{4.6}$$

where $\mathcal{I} = \mathcal{T}^m \cap \mathcal{T}^s$. The symbol $\mathcal{X}^q$ in (4.6) denotes the space of Lagrange multipliers defined on slave mesh $\mathcal{T}^s$, thus $\dim(\mathcal{X}^q) = \dim(\mathcal{X}^s)$.

We can express $m^h \in \mathcal{X}^m$, $s^h \in \mathcal{X}^s$ and $q^h \in \mathcal{X}^q$ in terms of their respective basis $\{N_i^m\}_{i=1}^{n^m}$, $\{N_j^s\}_{j=1}^{n^s}$, $\{N_k^q\}_{k=1}^{n^q}$, as $m^h = \sum_{i=1}^{n^m} N_i^m m_i$, $s^h = \sum_{j=1}^{n^s} N_j^s s_j$, $q^h = \sum_{k=1}^{n^q} N_k^q q_k$, where $\{m_i\}_{i=1}^{n^m}$, $\{s_j\}_{j=1}^{n^s}$, $\{m_k\}_{k=1}^{n^q}$ denote the coefficients. This allows us to reformulate Equation (4.6) as

$$\sum_{i=1}^{n^m} m_i (N_i^m,\ N_k^q)_{L_2(\mathcal{I})} = \sum_{j=1}^{n^s} s_j (N_j^s,\ N_k^q)_{L_2(\mathcal{I})}, \qquad \text{for } \ k \in \{1, \dots, n^q\}. \tag{4.7}$$

The Equation (4.7) can be expressed using vector-matrix notation as follows:

$$\mathbf{Bm} = \mathbf{Ds}, \tag{4.8}$$

where the vectors $\mathbf{m}$ and $\mathbf{s}$ contain the coefficients m_i and s_j, respectively. The components of the coupling matrix $\mathbf{B}$ are defined as $(\mathbf{B})_{k,i} = (N_i^m,\ N_k^q)_{L_2(\mathcal{I})}$, while the matrix $\mathbf{D}$ contains the entries $(\mathbf{D})_{k,j} = (N_j^s,\ N_k^q)_{L_2(\mathcal{I})}$. As a result, the discrete projection operator $\mathbf{\Pi}$ is obtained from (4.8) as follows:

$$\mathbf{\Pi} = \mathbf{D}^{-1}\mathbf{B}. \tag{4.9}$$

Transfer operators via pseudo-L^2-projection

Different choices of the multiplier space $\mathcal{X}^q$ produce different transfer operators. For example, if we set $\mathcal{X}^q = \mathcal{X}^s$, the transfer operator $\mathbf{\Pi}$ is nothing else than the well-known L^2-projection. In this particular case, the matrix $\mathbf{D}$ in (4.9) represents the mass

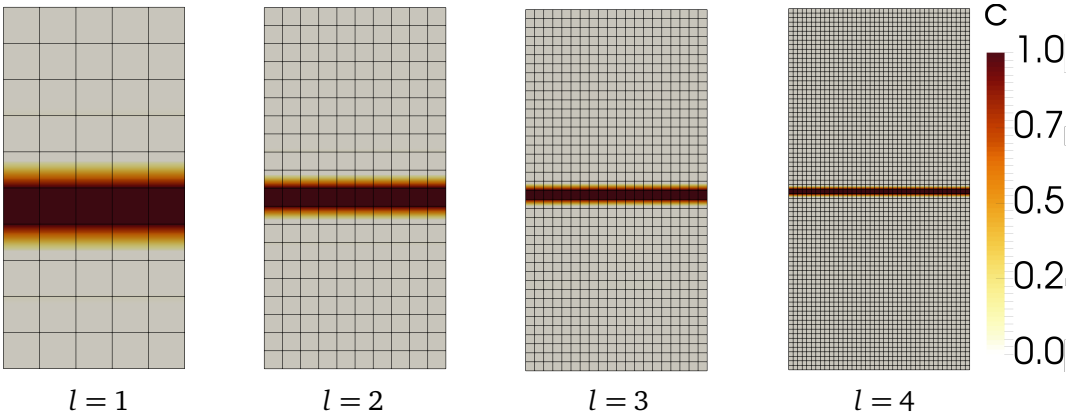

Figure 4.3: Two-dimensional tension test. The solution, crack pattern, depends on the mesh size and reassembles sharp fracture surface as $h \to 0$.

matrix defined on the slave space. Although, the mass matrix is usually sparse matrix, its inverse is dense, which produces the dense transfer operator $\mathbf{\Pi}$.

In this work, we consider a computationally cheaper alternative, namely the pseudo-L^2-projection [Dic10, OW01]. Thus, we employ the multiplier space $\mathcal{X}^q$, which is spanned by the dual basis functions, see Reference [Dic10] for details. This gives rise to a diagonal matrix $\mathbf{D}$. As its inverse is also a diagonal matrix, the evaluation of (4.9) is computationally cheap. In addition, resulting transfer operator $\mathbf{\Pi}$ is a sparse matrix.

As described in Section 2.2, the RMTR method makes use of three different transfer operators: prolongation, restriction, and projection. We assemble the prolongation operator $\mathbf{I}_l^{l+1}$ as in (4.9), by setting $\mathcal{X}^m = \mathcal{X}^l$ and $\mathcal{X}^s = \mathcal{X}^{l+1}$. In order to assemble the projection operator $\mathbf{P}_{l+1}^l$, we associate master and slave spaces as follows: $\mathcal{X}^m = \mathcal{X}^{l+1}$ and $\mathcal{X}^s = \mathcal{X}^l$. The restriction operator $\mathbf{R}_{l+1}^l$ is obtained as the adjoint of the prolongation operator, thus as $\mathbf{R}_{l+1}^l = (\mathbf{I}_l^{l+1})^T$. We note that the parallel implementation of the mortar projections is not a trivial task. Here, we employ library MOONoLith [KZ16], which provides an efficient and scalable implementation.

Remark 14. *We note that we refer to the operator $\mathbf{P}_{l+1}^l$ as to the projection. However, both transfer operators $\mathbf{I}_l^{l+1}$ and $\mathbf{P}_{l+1}^l$ are technically projections.*

Remark 15. *As described in Section 4.2.1, the mesh $\mathcal{T}^{l+1}$ is obtained by uniformly refining the mesh $\mathcal{T}^l$. In this case, i.e., $\mathcal{T}^l \subset \mathcal{T}^{l+1}$, the prolongation operator $\mathbf{I}_l^{l+1} : \mathcal{X}^l \to \mathcal{X}^{l+1}$ obtained as described above, coincides with standard finite element interpolation.*

4.2.2 Level-dependent objective functions

In this section, we discuss how to construct level-dependent objective functions $\{h^l\}_{l=1}^L$, given phase-field fracture simulations (4.5). We note that the standard approaches, discussed in Section 2.4, can not be readily used. This is due to the fact that the volumetric

approximation of the fracture surface integral (4.2) is determined by the length-scale parameter l_s. In the numerical simulations, the mesh size has to be at least half the size of the length-scale parameter l_s in order to resolve the fracture regions for the given l_s. Therefore, we are not able to represent the same fracture zones on the coarse level, as we can on the finest level. Figure 4.3 demonstrates this effect for a two dimensional tension test. Here, we consider a rectangular computational domain discretized with quadrilateral finite elements. The body is subjected to tensile loading, which results in a straight crack path inside the specimen. We perform the same experiment repeatedly while increasing the resolution of the mesh. At the same time, we also modify the length-scale parameter l_s as follows $l_s = 2h$, where h represents the mesh size. As we can see, the obtained crack paths differ for different resolution levels. Moreover, even larger inconsistencies between different refinement levels can be expected for more complex loading scenarios.

We propose to employ a novel type of level-dependent objective functions, which we call solution-dependent objective functions. The proposed functions can be seen as a variation of the second-order additive approach, described in Section 2.4.2, Equation (2.23). In particular, for all $l < L$, we define $h^l : \mathbb{R}^{n^l} \to \mathbb{R}$ as

$$h^l(\boldsymbol{x}^l) := \underbrace{\widetilde{f}^l(\boldsymbol{x}^l)}_{\text{discretization of (4.12)}} + \underbrace{\langle \delta\widetilde{\boldsymbol{g}}^l, \boldsymbol{s}^l \rangle}_{1^{\text{st}}-\text{order coupling}} + \chi_1(\mathbf{c}, \epsilon) \underbrace{\frac{1}{2}\langle \boldsymbol{s}^l, \delta\widetilde{\boldsymbol{H}}^l(\boldsymbol{s}^l) \rangle}_{2^{\text{nd}}-\text{order coupling}}. \tag{4.10}$$

The three main differences of the formulation (4.10) compared to (2.23) are:

- The indicator function $\chi_1 : \mathbb{R}^{n^L} \times \mathbb{R} \to \mathbb{R}$ is used to turn on and off the second-order coupling term.
- The objective function f^l, obtained by discretizing the energy functional (4.5), is replaced by its modified variant, function $\widetilde{f}^l$.
- The first- and the second-order coupling terms $\delta\widetilde{\boldsymbol{g}}^l$ and $\delta\widetilde{\boldsymbol{H}}^l$ are obtained using the truncated transfer operator $\widetilde{\mathbf{I}}_l^{l+1}$.

Our first modification is motivated by the fact that during the pre-stage phase of the simulation, the crack has not yet propagated. Thus, the coarse-level discretization of (4.5) is a valid representation of a fine-level discretization of (4.5), with respect to the discretization error. In this case, the usage of the first-order consistency approach is sufficient. Therefore, we define indicator function

$$\chi_1(\mathbf{c}, \epsilon) := \begin{cases} 1, & \text{if } \max\limits_{i=0,\dots,n^L}[c_i] > \epsilon, \\ 0, & \text{otherwise}, \end{cases} \tag{4.11}$$

which allows for turning on and off the second-order consistency term. The symbol c_i in (4.11) denotes the nodal coefficients of the phase-field on the finest level. A threshold $\epsilon \in [0, 1)$ defines a limit, after which we believe that the fine-level model cannot be

represented well by the coarse-level discretization. In our experience, $\epsilon \approx 0.85$ provides a reasonable choice.

Our second modification consists of replacing the objective function f^l by its modified variant $\widetilde{f}^l$. To this aim, we split the total potential energy (4.3) into the elastic and the fracture energy. For the elastic part of the energy, we employ a coarse-level discretization, since it can represent the fine-level elastic energy well. For the fracture part of the energy, we use information restricted from the finest level, since the value of a regularized fracture energy differs at the different discretization levels. To incorporate both of the above requirements, the modified objective function $\widetilde{f}^l$ is obtained by discretizing the modified energy functional

$$\begin{aligned}\widetilde{\psi}(\mathbf{u}, c) := &\int_\Omega \Big(\big(g(c)+k\big)\psi_e^+(\boldsymbol{\varepsilon}(\mathbf{u})) + \psi_e^-(\boldsymbol{\varepsilon}(\mathbf{u}))\Big)\, d\Omega \\ &+ \int_\Omega \chi_2(c,\epsilon)\left(\mathcal{G}_c\left(\frac{1}{2l_s}c^2 + \frac{l_s}{2}|\nabla c|^2\right)\right) d\Omega,\end{aligned} \tag{4.12}$$

where the indicator function χ_2 is defined as

$$\chi_2(c,\epsilon) := \begin{cases} 0, & \text{if} \quad c > \epsilon, \\ 1, & \text{otherwise.} \end{cases}$$

Our third modification consists of using the truncated transfer operator $\widetilde{\mathbf{I}}_l^{l+1}$ in order to evaluate first- and second-order consistency terms, e.g., $\delta\widetilde{\boldsymbol{g}}^l$ and $\delta\widetilde{\boldsymbol{H}}^l$, respectively. In addition, the truncated transfer operator is used for prolongation of the coarse-level correction. We define the truncated transfer operator $\widetilde{\mathbf{I}}_l^{l+1}$ as follows:

$$(\widetilde{\mathbf{I}}_l^{l+1})_{pk} = \begin{cases} 0, & \text{if } \; c_p^{l+1} > \epsilon, \\ (\mathbf{I}_l^{l+1})_{pk}, & \text{otherwise,} \end{cases}$$

where c_p^{l+1} denotes a coefficient of the phase-field variable on level $l+1$ associated with p-th node. The usage of truncated transfer operator $\widetilde{\mathbf{I}}_l^{l+1}$ is two-fold. On one side, it does not allow for the broken part of the domain to be modified on the coarse level. On the other side, it ensures that the prolongated coarse-level correction does not modify broken parts of the domain on the finest level. This is essential in order to avoid the widening of the crack.

Remark 16. *Our design of a coarse-level model for phase-field fracture is conceptually similar to an active-set RMTR method with truncated basis functions, described in Section 2.5.4. The main difference is that we do not allow a coarse level to modify phase-field as soon as it becomes larger than some threshold ϵ, regardless of the irreversibility condition being active or not.*

4.2.3 Multilevel treatment of constraints

We ensure that the prolongated coarse-level corrections do not violate the fine-level irreversibility constraint by utilizing constraint projection rules from Section 2.5.4. The projection of the trust-region bounds to coarser levels follows the discussion in Section 2.5.3.

4.3 Implementation

Our parallel phase-field fracture simulation framework combines parallel finite element discretization and solution strategy. While all presented solution strategies were implemented as part of UTOPIA [ZKN+16] library, we used two different finite element frameworks for the implementation of finite element assembly.

Finite element assembly and unstructured meshes

We implemented the discretization of several phase-field fracture models in the library ROOK [KK17]. The ROOK library is built on top of the finite element framework MOOSE [GNHLG09] and exploits distributed memory parallelism. The mesh distribution and the data layout are provided by the mesh partitioner ParMETIS [KSK97].

The MOOSE framework is built on top of library LibMesh [KPSC06], which provides support for unstructured meshes and also performs adaptive refinement in parallel. This is very convenient, especially in the multilevel settings, as the refinement procedure gives rise to the multilevel hierarchy of the meshes and, therefore, the finite element spaces.

Finite element assembly and structured meshes

The implementation of finite elements using structured meshes allows for many performance optimizations. Therefore, we have implemented the finite element discretization of the phase-field fracture model specifically for structured meshes using UTOPIA library. This implementation utilizes PETSc-DMDA [BAA+19] for creating a hierarchy of distributed structured meshes. This implementation was used to produce results related to dense stochastic fracture networks, presented in Section 4.4.3.

Numerical considerations

The most time-consuming part of the phase-field fracture simulations is an assembly of the derivatives. In particular, around 75% and 10% of the computational time is invested in the assembly of the Hessian and gradient, respectively. Here, we discuss the details that define the resulting performance of our implementation. We perform eigendecomposition of the strain tensor for each quadrature point using the library LAPACK [ABB+99]. The positive and negative parts of stress tensor are computed using techniques described in Reference [ML01, Algorithm A].

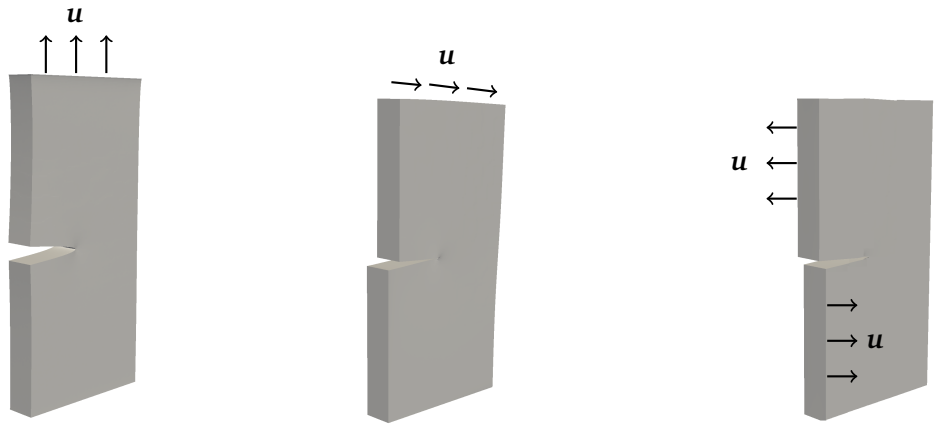

Figure 4.4: Boundary value problem set up for three fracture modes. *Left:* Mode I, tension test. We impose an incremental displacement $\Delta\mathbf{u} = 1 \times 10^{-4}$ mm. *Middle:* Mode II, shear test. We prescribe an incremental displacement $\Delta\mathbf{u} = 1 \times 10^{-4}$ mm. *Right*: Mode III, tear test. An incremental displacement $\Delta\mathbf{u} = 1 \times 10^{-4}$ mm is enforced.

To save computational time, we pre-compute all linear components associated with the model, which include all test-space related quantities such as shape-functions, gradients, and geometric quantities such as Jacobian matrices and determinants. However, due to the fact that the model is nonlinear, many quantities can not be precomputed and have to evaluated using the current iterate.

The time required by the assembly of the Hessian is additionally reduced by exploring the symmetry and assembling only the lower part of the Hessian matrix. Furthermore, we take into account the fact that the phase-field parameter equals zero on the fully intact parts of the domain. This allows us to skip the evaluations of all the terms involving multiplication with phase-field variable c, for $c = 0$.

4.4 Numerical experiments

In this section, we present several numerical examples, which we use in order to examine the performance of our RMTR method. All of the presented examples are three dimensional and employ the length scale parameter $l_s = 2h$, where h denotes the size of the largest element.

4.4.1 Fracture modes

In the beginning, we investigate the performance of the method on three fracture modes. We consider a three dimensional brick of size 1 mm×0.5 mm×0.1 mm with 0.2 mm initial notch. The specimen is then subjected to three different loading scenarios, which can force crack propagation, namely

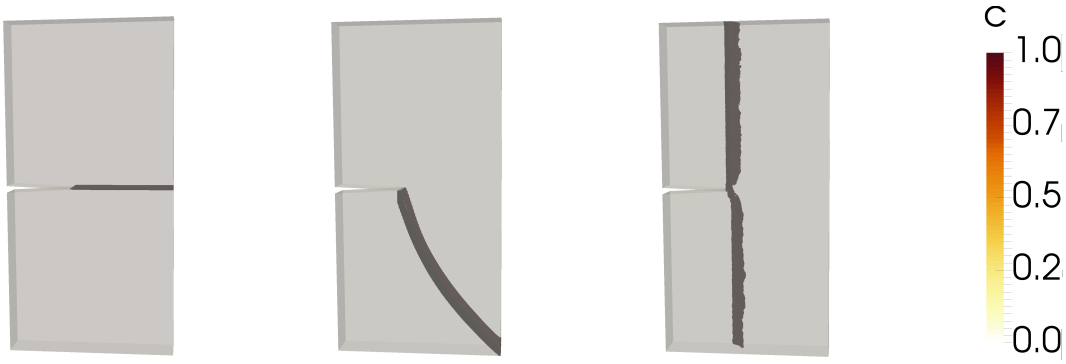

Figure 4.5: Iso-surfaces of the damage in the reference configuration for $c = 0.99$ for tension, shear, tear tests (from left to right). The presented results originate from an numerical experiment with approx. 105 thousand dofs.

- **Mode I.** - Tension (tensile stress)
- **Mode II.** - Shear (in-plane shear stress)
- **Mode III.** - Tear (out-of-plane shear stress).

Figure 4.4 depicts an initial set up and details on the prescribed Dirichlet boundary conditions. During the simulations, we use the following set of Lamè parameter $\lambda = 12.1\,\text{GPa}$, shear modulus $\mu = 7.9\,\text{GPa}$ and the critical energy release $\mathcal{G}_c = 10^{-3}\,\text{GPa}$.

The final crack patterns for all three loading scenarios are shown on Figure 4.5 by depicting the iso-surface for $c = 0.95$. As we observe, the volumetric approximation of the fracture resembles closely the sharp crack surface. The quality of this approximation depends on mesh resolution and hence the ability to resolve the fracture zones accurately. Figure 4.6 on the left demonstrates the three-dimensional representation of the phase-field c for the tension example. The cross-section of phase-field across a crack is depicted in the Figure 4.6 on the right.

Figure 4.7 on the left depicts the elastic (solid lines) and fracture energy (dotted lines) as a function of time for tension example for different mesh resolutions. The obtained results suggest that until the critical time, $t_c = 3.8 \times 10^{-3}\,\text{s}$, is reached the elastic energy is increasing, while the fracture energy remains zero. Once the critical time t_c is reached, the “brutal" crack propagation takes place. The specimen breaks and the fracture surface is formed. As a consequence, the fracture energy increases, and the elastic energy vanishes. We observe that the mesh resolution and the associated value of the length-scale parameter l_s influence the time of crack propagation as well as the values of the elastic and fracture energy. In particular, each mesh refinement step allows us to reduce the value of length-scale parameter l_s by a factor of two. As the l_s tends to zero, the fracture energy converges to $\mathcal{G}_c W$, where W denotes the area of the fracture surface. For our boundary value problem setup, the fracture energy can be computed as $\Psi_\text{f} = \mathcal{G}_c W = 10^{-3}(0.3 \times 0.1)$.

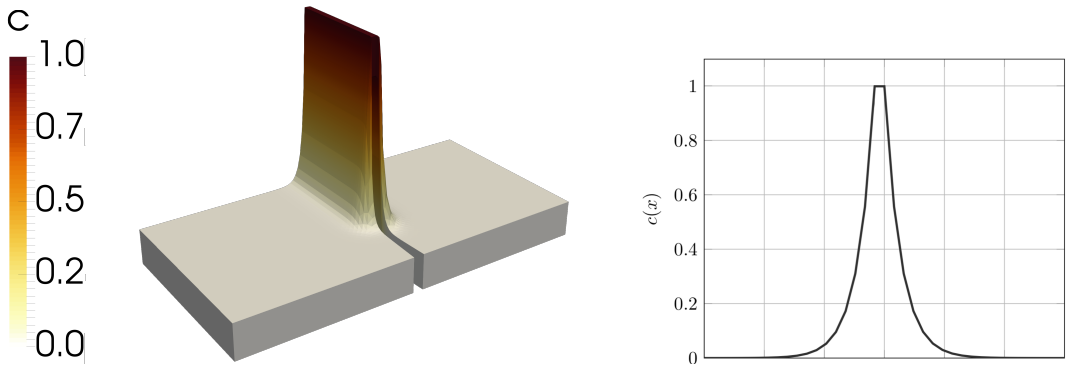

Figure 4.6: *Left:* Three dimensional representation of the phase-field for a crack for a tension example with approx. 782 thousand dofs. The displacements are scaled by a factor of two. *Right:* The phase-field profile corresponding to the fractured solution.

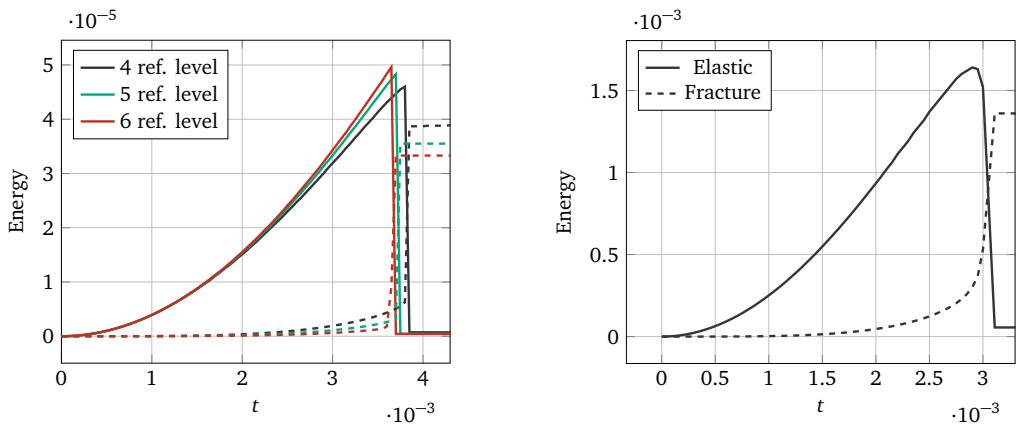

Figure 4.7: Evolution of the elastic (solid lines) and fracture (dashed lines) energy. *Left:* Tension example with respect to increasing mesh resolution and decreasing length-scale parameter l_s. *Right:* Conchoidal fracture example with approx. 1 million dofs.

4.4.2 Conchoidal fracture

The conchoidal fractures are commonly observed in fine-grained or amorphous materials such as rocks, minerals, and glasses. They are fascinating because they do not follow any natural way of separation and result in a curved breakage, which resembles the concentric undulations of a seashell [Hod61]. Figure 4.8 on the left demonstrates an example of conchoidal fracture in quartz. From the modeling point of view, the conchoidal fracture is interesting because the crack initiates usually inside the body and not on the surface of the specimen. Therefore, the phase-field model needs to be capable of predicting the crack initiation and propagation without any initial crack or notch.

Our numerical experiment considers a real rock geometry as depicted in Figure 4.8 on

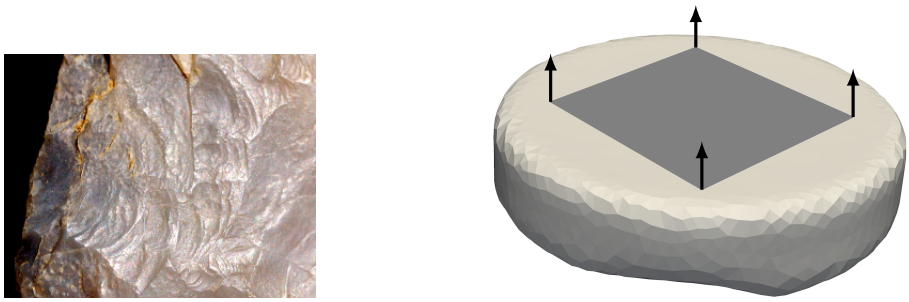

Figure 4.8: *Left:* Conchoidal fracture in quartz [Hen18]. *Right:* Rock geometry and the boundary value problem set up. The shaded area (1m×1 m) illustrates a part of the surface, where time-dependent Dirichlet boundary conditions are enforced. We prescribe an incremental displacement $\Delta\mathbf{u} = 5 \times 10^{-3}$ mm while we keep the phase-field variable fixed, e.g., $c = 0$. The bottom surface of the rock was held fixed during the entire simulation.

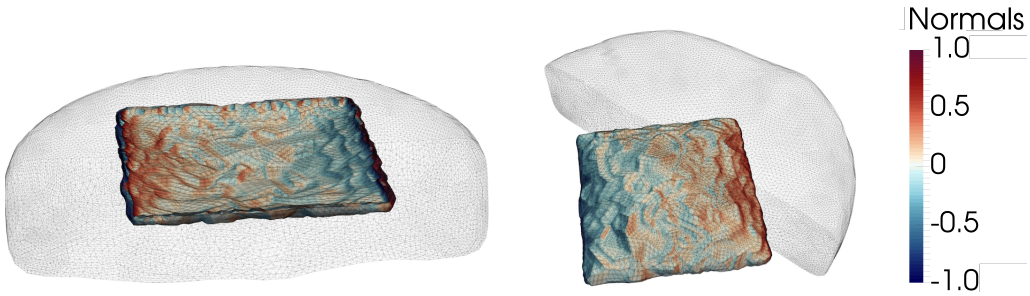

Figure 4.9: Result of a conchoidal fracture simulation, a numerical experiment with approx. 7.9 million dofs. A crack domain is shown inside of the rock geometry. One should note that the fracture resembles curved patterns as expected for conchoidal fracture. *Left:* Top view. *Right:* Bottom view.

the right. The rock mesh is unstructured and contains rough surfaces [PVN+18], which can only be represented well with a relatively fine mesh. Following Reference [MK16], we simulate a conchoidal fracture by prescribing the displacement on the part of the upper boundary, see Figure 4.8 on the right. On the same part of the boundary, the phase-field variable is enforced by prescribing Dirichlet boundary condition $c = 0$. On the lower part of the boundary, we prescribe zero Dirichlet boundary conditions for both displacement and the crack phase-field.

As a material parameters, we use Lamé constant $\lambda = 100\,\text{GPa}$, shear modulus $\mu = 100\,\text{GPa}$ and the critical energy release $\mathcal{G}_c = 10^{-3}\,\text{GPa}$. The result of the simulation, the crack surface, is demonstrated in Figure 4.9. The Figure 4.7 on the right demonstrates the evolution of the elastic and fracture energy for this numerical example. The detailed study of the proposed simulation on a parallelepipedal domain is presented in

References [BKKW18, BKKW19].

4.4.3 Pressure induced fracture

Our last numerical experiment considers pressure-induced (pneumatic) fracture. The pressure-induced fracture involves the injection of air or fluid at sufficient pressure which induces crack propagation. In order to simulate pneumatic fracture, we follow the approach derived in [MWW14, MWW15a, MWW15b]. The energy functional $\Psi(\mathbf{u}, c)$ from (5.3) is extended as follows

$$\Psi(\mathbf{u}, c, p) := \int_\Omega \big(g(c)(1-k)+k\big)\psi_e^+(\boldsymbol{\varepsilon}(\mathbf{u})) + \psi_e^-(\boldsymbol{\varepsilon}(\mathbf{u}))\, d\Omega + \int_\Omega \mathcal{G}_c\left(\frac{1}{2l_s}c^2 + \frac{l_s}{2}|\nabla c|^2\right) d\Omega + \int_\Omega g(c)p\nabla\cdot\mathbf{u}\, d\Omega,$$

where $p : \Omega \to \mathbb{R}$ is a pressure, which can induce fracture propagation. We focus only on the fracture propagation, i.e., we assume that pressure p is given a priori. To improve the reliability of the model, the phase-field fracture model can be coupled with the poroelasticity equations such as Biot's equations [MWW15a]. This would allow for simulating induced hydraulic fracturing in a poroelastic medium rather than in an elastic medium.

Our numerical experiments consider pressure induced fracture of realistic stochastic fracture networks in two-dimensional and three-dimensional scenarios. The initial fractures are prescribed by initializing the value of phase-field variable c. In particular, we check if the nodal position lies inside of a parametric fracture description, then we mark the related parts of the domain as broken. This is done by setting the nodal coefficient for the phase-field variable to be equal to 1. Otherwise, we mark the material as intact by prescribing the nodal value of the phase-field to be equal to 0.

During the simulations, we use the material parameters describing the mechanical response of granite material [YWN+18]. The critical energy release rate is set to $\mathcal{G}_c = 10^{-3}$ GPa whereas the Lamè parameters are set to $\lambda = 100$ GPa and $\mu = 100$ GPa.

Small scale experiment

This numerical experiment is performed on a cube of size 1mm ×1mm ×1mm. The initial set-up of the simulation takes into account ten randomly distributed fractured zones, see Figure 4.10 on the left. Through the simulation, we apply zero Dirichlet boundary conditions for the displacement field on all six sides of the domain. At each pseudo time-step t, the pressure load is linearly increased as $p(t) = p_0 + tp_0$, with an initial pressure $p_0 = 1$ GPa. The simulation time-step was set-up to $\Delta t = 1$ s. Figures 4.10 in the middle demonstrate evolution of the fracture by illustrating crack iso-surfaces for $c = 0.9$. Figure 4.10 on the right depicts the obtained results by introducing a cut across the computational domain. This allows us to observe interesting crack patterns and interacting fractures inside the computational domain.

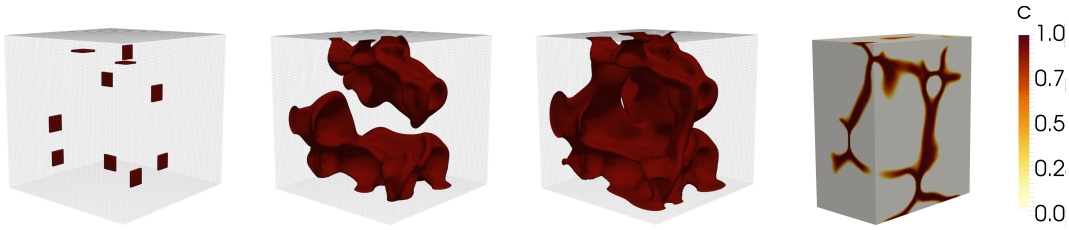

Figure 4.10: Pressure-induced fracture, an example with approx. 4.1 million dofs. *Left:* Initial set-up with ten randomly distributed fracture zones. *Middle:* Crack patterns at time $t = 11\,\mathrm{s}$ and $t = 12\,\mathrm{s}$. One can observe, robust crack propagation between two subsequent time-steps. *Right:* Cut across the computational domain at the end of the simulation, thus at $t = 12\,\mathrm{s}$. The interesting crack patterns as well as crack merging and crack branching can be observed on several places.

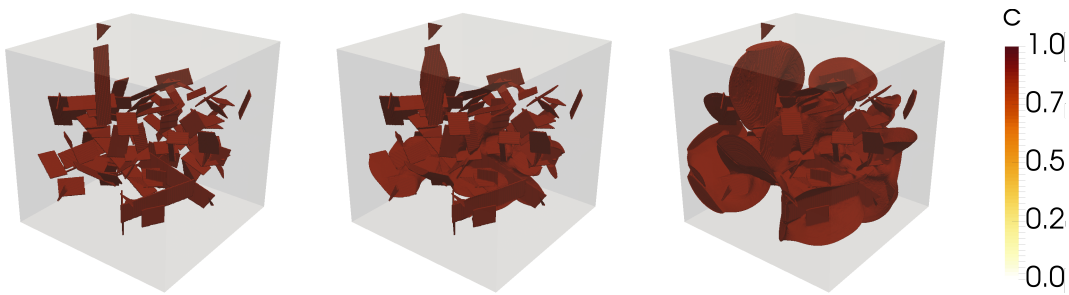

Figure 4.11: A three-dimensional simulation with 100 randomly distributed fractures and approx. 242 million dofs. The fracture iso-surface is displayed for $c = 0.95$. Snapshots taken at different times, i.e., $t = \{0; 0.75; 0.9\}$ s (from left to right).

Large scale experiments (stochastic fracture networks in 2D and 3D)

These numerical examples consider pressure induced fracture propagation of realistic stochastic fracture networks. We generate the pre-existing fracture networks using a two-stage process. First, we describe each fracture as a one-dimensional object with a randomly assigned hypo-center, orientation, and length. In particular, we employ a uniform distribution to place the hypo-centers over the entire domain and assign their orientation to a value between $-80°$ and $80°$. The fracture length is drawn from a scale-invariant power-law distribution [dDDB01]. In the second stage, each fracture is regularized through a volumetric representation with an artificial width w proportional to the mesh size h, where $w = 2h$. Hence, the resulting fracture networks consist of smooth parallelepipeds randomly embedded in the surrounding matrix. The fracture network represents the initial datum for the phase-field parameter which evolves during the simulation depending on the prescribed pressure and the boundary conditions.

Our two-dimensional experiment considers a rectangular domain of size

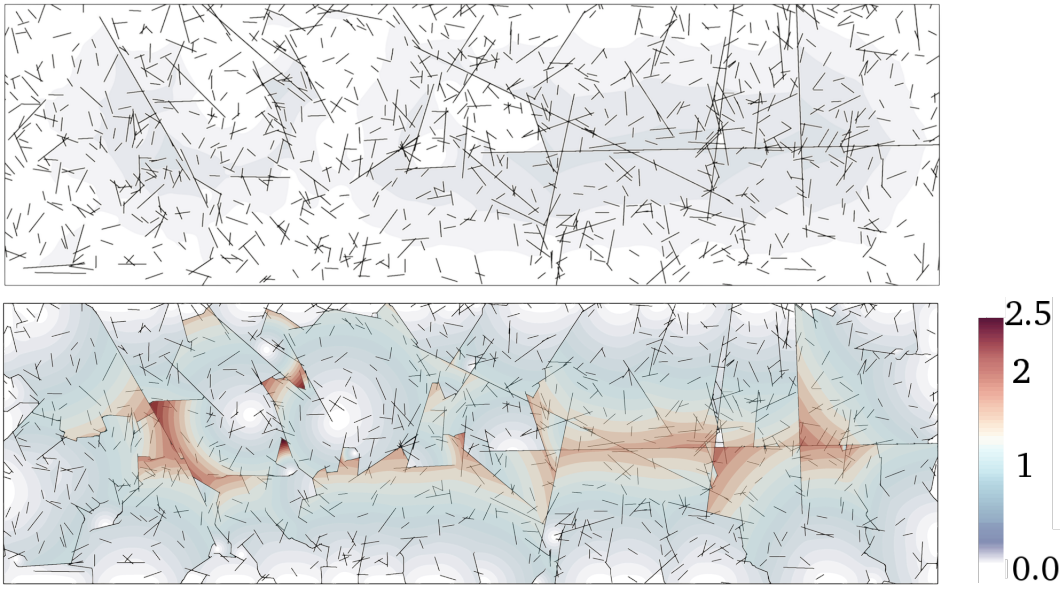

Figure 4.12: Two-dimensional simulation with 1,000 fractures and approx. 13 million dofs. The colored overlay represents the displacement magnitude. *Top:* Initial fracture network. *Bottom:* Final configuration.

3 mm × 10 mm. We construct a fracture network by generating 1,000 initial fractures. During the whole simulation, we apply zero Dirichlet boundary conditions for the displacement field on all four sides of the domain. A pressure load is linearly increased at each loading step and defined as $p(t) = p_0 + \Delta t p_c$, with $p_0 = 10^{-3}$ GPa, $\Delta t = 1$ s and $p_c = p_0 = 10^{-3}$ GPa. The initial setup and the simulation result are depicted in Figure 4.12.

Our three-dimensional experiment considers a fracture network embedded in cube of size 1mm × 1mm × 1mm. The initial set-up of the simulation takes into account 100 randomly distributed fractures as shown in Figure 4.11. We apply zero Dirichlet boundary conditions for the displacement field on all sides of the domain. A pressure load is linearly increased at each loading step and defined as $p(t) = p_0 + \Delta t p_c$, with $p_0 = 10^{-5}$ GPa, $\Delta t = 0.05$ s and $p_c = 10^{-3}$ GPa. Figure 4.11 depicts the evolution of the fracture network.

4.5 Numerical results

In this section, we investigate the performance of the RMTR method. Our investigation starts by analyzing the effect of different coarse-level models on the performance of the RMTR. Further, we compare the performance of the RMTR method with the single-level TR. Both trust-region based methods are employed inside a monolithic solution scheme.

Table 4.1: Choice of parameters used inside TR/RMTR algorithms.

Parameter	η_1	η_2	γ_1	γ_2	$\Delta^L_{0,0}$	μ_1	μ_2
Value	0.1	0.75	0.5	2.0	1	1	1

Additionally, we include a comparison with a staggered solution scheme, as it is almost universally used in the literature for solving the phase-field fracture problems. While comparing these solution strategies, we focus on several aspects, such as the evolution of fracture and elastic energy, computational time, and scaling properties.

Building blocks of solution strategies

While benchmarking, trust-region based algorithms were set up with the parameters as described in Table 4.1. The single-level TR method employs a Projected Conjugate Gradient method with a block Jacobi preconditioner to solve the trust-region subproblem. The number of blocks used by the preconditioner corresponds to the number of processors (PETSc default). On each block/processor, we employ the sparse direct linear solver MUMPS (version 5.1.1) [ADLK00].

The RMTR method was configured as a V-cycle with one pre/post-smoothing step and two coarse solves. We solve the trust-region subproblems on levels $l = 1, \dots, L$ approximately with 10 steps of the Projected Conjugate Gradient method preconditioned with a Jacobi preconditioner [NW06]. On the coarsest level, $l = 0$, we employ the active-set strategy [BMM+10] with the sparse linear solver MUMPS. The computations on levels $l = 1, \dots, L$ are performed in parallel, using distributed computing without hyperthreading. On the coarsest level, we perform redundant computations and solve the problem on each processor. This reduces the communication cost and allows for better scalability of the multilevel method.

In each time-step, the staggered solution scheme alternatively solves two subproblems, related to the displacement and the phase-field, respectively, see Algorithm 8 for details. Our choice of solution strategies for both subproblems follows the most common practice. For the displacement subproblem, we use Newton's method with the sparse direct linear solver MUMPS, as the problem is convex. The phase-field subproblem requires the solution of a convex constrained quadratic minimization problem. Therefore, we use the semi-smooth Newton's method [Ulb11], also with the sparse direct solver MUMPS.

Although the linear solver MUMPS is designed for distributed computing, the applicability of direct solvers to large scale problems is limited. Here, the memory requirements and computational cost grow quadratically with the number of dofs in 3D. The efficient alternatives include for example Krylov-space methods preconditioned with a linear multigrid [Saa03]. The presented work does not investigate those alternatives but rather focuses on comparison with a direct solver since this approach is almost universally used in the literature.

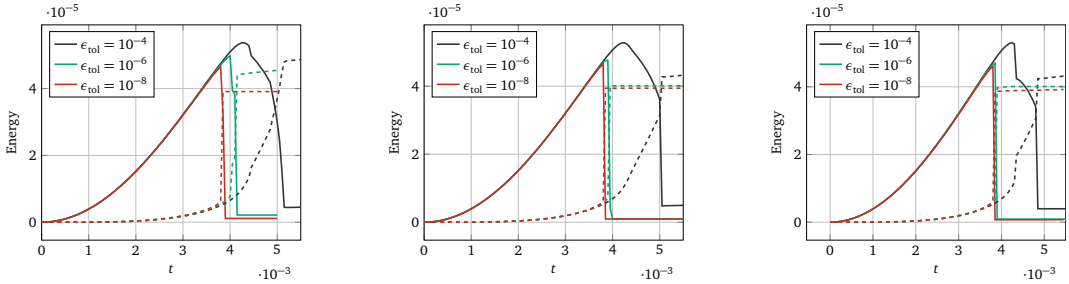

Figure 4.13: Evolution of elastic (solid lines) and fracture energy (dashed lines) for the tension example with approx. 105 thousand dofs with respect to different stopping tolerance ϵ_{tol}. *Left:* Staggered solution scheme. *Middle:* The TR method. *Right:* The RMTR method.

Importance of reliable stopping criterion

The employed solution strategies terminate, when the following stopping criterion is satisfied

$$\mathcal{E}(\boldsymbol{x}_i) < \epsilon_{\text{tol}} \text{ or } \|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\| < 10^{-14}, \tag{4.13}$$

where the current iterate is denoted by $\boldsymbol{x}_i$. In the context of our recursive multilevel trust-region algorithm, $\boldsymbol{x}_i$ is considered to be the fine-level iterate. The criticality measure $\mathcal{E}(\boldsymbol{x}_i)$ is as defined in Section 1.1 and $\|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\|$ denotes the correction size.

Our numerical experience indicates that the choice of ϵ_{tol} influences the quality of the results drastically. Figure 4.13 depicts the evolution of elastic and fracture energy for the tension example obtained by using the staggered scheme, the TR, and the RMTR method, respectively. As we can see, higher values of ϵ_{tol} slow down the crack propagation and overestimate the total value of the energy. For values of $\epsilon_{\text{tol}} = 10^{-8}$, the crack propagation is equivalent for all presented solution strategies.

Remark 17. *We note that the choice of the stopping criterion* (4.13) *is not widely used for the staggered scheme. Following Reference [BFM00], most authors terminate the staggered scheme, when the change in the phase-field drops below a certain tolerance. However, this does not allow for a fair comparison with solution methods employed inside of the monolithic scheme.*

4.5.1 Effect of different coarse-level models on the performance of the RMTR

In this section, we investigate the performance of the RMTR method configured with different types of additive coarse-level models, as described in Section 4.2.2. We monitor the number of nonlinear V-cycles required by the RMTR method to converge to the desired tolerance during the time-step, see Figure 4.14. As we can see from Figure 4.14, all

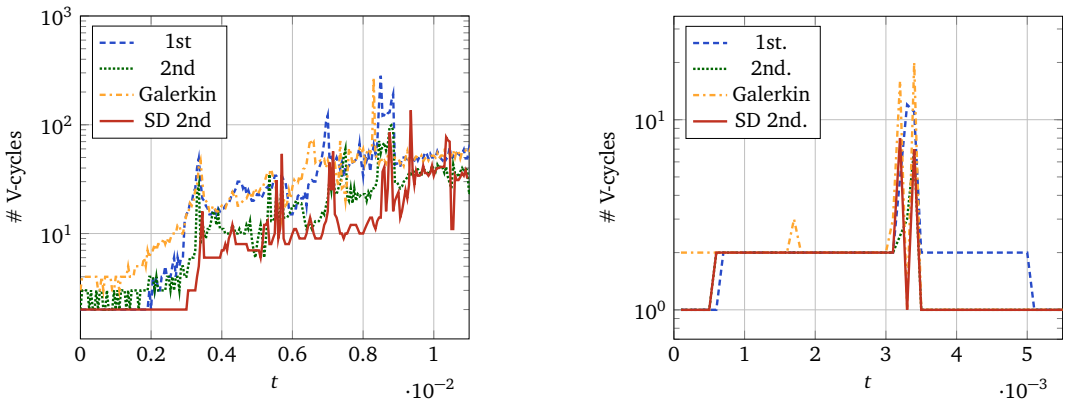

Figure 4.14: Number of nonlinear V-cycles over time-steps. *Left:* Shear test, the specimen with approx. 782 thousand dofs. *Right:* Conchoidal fracture, example with approx. 8 million dofs. RMTR is set up with 3 levels. For each example, we depict the performance of the RMTR method with respect to different coarse-level models. Please note that x and y axes differ for both graphs.

variants of the RMTR method converge. However, the convergence rate differs. Based on our numerical experiments, we conclude that the first-order consistency approach (2.19) yields the worst convergence rates. This is especially true in the post-stage, where the fracture occurs and propagates. The slow convergence is connected to the fact that the coarse-level models are not suitable representations of the fine-level fracture zones. All presented experiments use value of threshold $\epsilon = 0.85$ in (4.11).

Figure 4.14 demonstrates that all three approaches: Galerkin, the first-order consistency and the second-order consistency achieve worse convergence rates as our novel solution dependent second-order model (4.10). We can compute achieved speed up with respect to any other coarse-level model as follows

$$\frac{\#\text{ accumulated V-cycles}}{\#\text{ accumulated V-cycles with the solution dependent second-order model}}.$$

From the obtained results, Table 4.2, we can clearly see that usage of our solution dependent second-order coarse-level model is beneficial and leads to significant speed up. For example, for the tear test, the speed up is close to 3.15, when compared to the first-order consistency approach.

All results reported in the remainder of this section employ our RMTR method with the coarse-level models based on the solution dependent second-order consistency approach, Section 4.2.2.

4.5.2 Comparison of RMTR with standard solution strategies

In this section, we compare the performance of the trust-region methods used from the monolithic scheme with the performance of the staggered scheme. The comparison starts

Table 4.2: Speed up of the RMTR method, configured with the solution dependent second-order model with respect to alternative variants, namely first-order consistency, second-order consistency, and Galerkin model.

	Tension	Shear	Tear	Conchoidal
1st-order	1.17	2.79	3.15	1.36
2nd-order	1.03	1.71	1.21	1.12
Galerkin	1.08	2.59	1.42	1.30

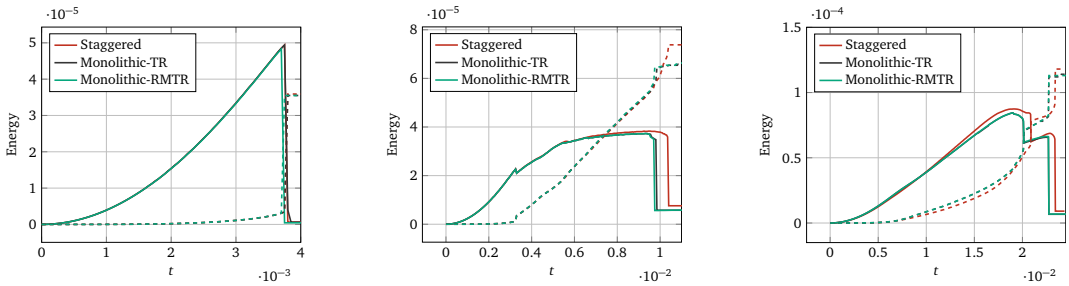

Figure 4.15: Evolution of the elastic (solid lines) and fracture (dashed lines) energy for tension, shear, tear (from left to right). The simulations were performed on a brick with approx. 782 thousand dofs. Results were obtained with different solution strategies, namely the staggered solution scheme and a monolithic solution scheme with the TR and the RMTR method.

by analyzing the evolution of elastic and fracture energy for three fracture modes. Figure 4.15 depict the elastic (solid lines) and fracture energy (dotted lines) as a function of time for tension, shear, and tear examples, respectively. We observe, that the fracture propagation starts always at the same time independently from solution strategy. However, the crack propagates slower when the staggered scheme is used. Whereas, both TR and RMTR methods produce the same result.

The inconsistencies in the results can be caused by several factors. For example, the error accumulated over several time-steps can be larger when the staggered scheme is used. As discussed in the previous section, the insufficiently resolved spatial problems in each time-steps can slow down the crack propagation. This would also explain an occurrence of the discrepancies only in the later stages of the simulations. Similar observations are found in References [GL16, LLMZ16]. The difference in the results can also be a consequence of the non-convexity of energy functional (4.5). In general, there are no theoretical guarantees that different solution strategies should converge to the same local minimizer. If this is indeed the case, comparison with experimental data is necessary to assess the quality of the obtained solution and of the model.

Robustness with respect to the refinement level and the length-scale parameter

Classical multilevel solvers, for instance, multigrid, are level-independent, in the sense, that the number of iterations stays constant with respect to an increasing number of dofs. Here, we investigate the level-independence of our RMTR algorithm by measuring the number of required V-cycles for three fracture modes while the increasing number of levels/dofs. We remark that the length-scale parameter l_s decreases with increased resolution. As a consequence, the volumetric approximation of the fracture reassembles the sharp fracture surface more closely as the refinement level increases.

Figure 4.16 shows, that the number of required V-cycles increases slightly with the number of levels. This might be caused by two factors: Firstly, phase-field fracture problems discretized on the different grid are not able to represent the same fractures. Therefore, simulating fracture propagation on different discretization levels does not imply that we are solving the same problem. The second reason, why we are not able to observe level-independence, might come directly from the use of the trust-region globalization. During certain stages of the simulation, the trust-region radius might become too small. In this case, the sum of corrections coming from upper levels of the multilevel hierarchy already satisfies the constraints provided by the trust-region radius on the finest level. As a consequence, the recursion terminates before reaching the coarsest level. The low frequencies of the error are not eliminated, which results in a larger number of nonlinear V-cycles. However, this is a small price to pay for obtaining a globally convergent multilevel solution strategy.

In comparison, Figures 4.17 demonstrates how the number of required nonlinear iterations changes with the refinement level for the single-level TR method. We notice that the growth in the number of iterations is prevalent from one refinement level to another. Figure 4.18 illustrates the number of iterations required by the staggered scheme as a function of time. Here, we perform experiments only with the refinement levels 4 and 5, as we are not able to satisfy the memory requirements of a direct solver for the problem on a coarser grid. The obtained results suggest, that the number of iterations also grows with the resolution and a smaller value of the length-scale parameter l_s.

Remark 18. *Figure 4.18 illustrates only the number of outer/staggered iterations. It does not take into account the increase in the computational cost of the inner nonlinear solvers.*

4.5.3 Computational complexity and scalability

One of the most appealing properties of the multilevel algorithm is its optimal complexity $\mathcal{O}(n)$, where n denotes the number of unknowns. This means that the required solution time increases linearly with an increasing number of dofs. We can demonstrate the computational complexity by measuring the execution time of one V-cycle while increasing the number of dofs. Our experiment considers tension test from Section 4.4.1.

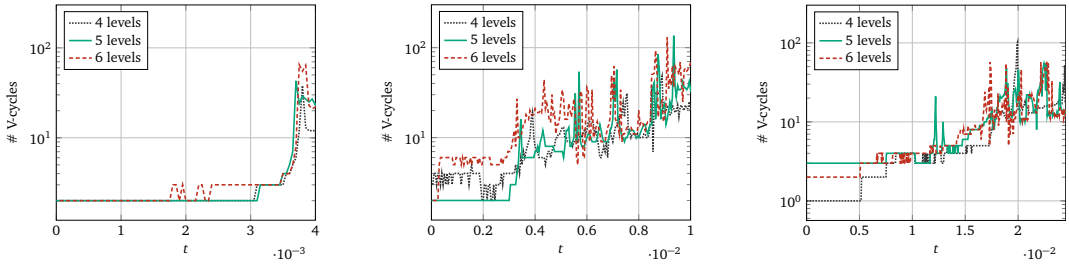

Figure 4.16: Number of nonlinear V-cycles required by the RMTR method over time-steps. *Left:* Tension test. *Middle:* Shear test. *Right:* Tear test. The performance of our RMTR algorithm is monitored for different discretization levels.

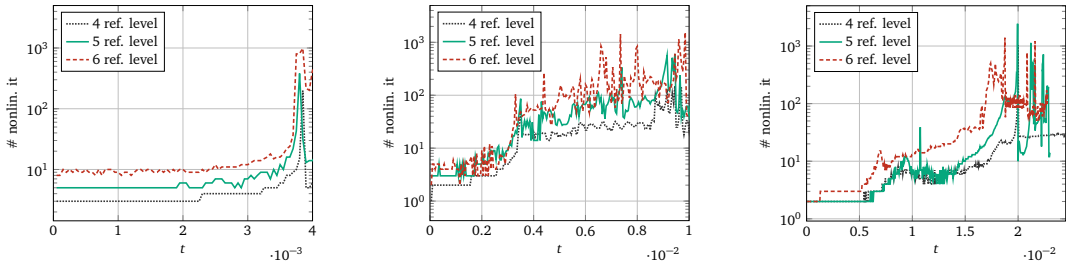

Figure 4.17: Number of nonlinear iterations required by a single-level TR method overall time-steps for tension, shear, tear test (from left to right). The performance of the TR algorithm is monitored for different discretization levels.

The obtained results, depicted in Figure 4.20 on the left, demonstrate that our implementation of the RMTR method is indeed of optimal complexity.

The computational complexity of a single-level TR and the staggered solution scheme highly depends on the underlying linear solver. For example, if we employ a direct linear solver, the computational cost grows as $\mathcal{O}(n^2)$ in 3D. Therefore, when n becomes too large, direct solvers suffer from the quadratic growth of computational cost and the memory requirements. Indeed, we are not able to solve problems with more than the million dofs with the staggered scheme due to the huge memory requirements.

To make our comparison of the solution strategies complete, we evaluate the total computational time required by the simulations. Table 4.3 summarizes the total execution time for all numerical examples. The reported times are obtained by using 5 processors, since all methods scale well up to 5 processors, see Section 4.5.4. The obtained results demonstrate that the monolithic scheme outperforms the staggered scheme for all test cases. We also notice that the RMTR method provides significant benefits compared to the single-level TR. For example, we achieve a reduction in time by 68% for tear example.

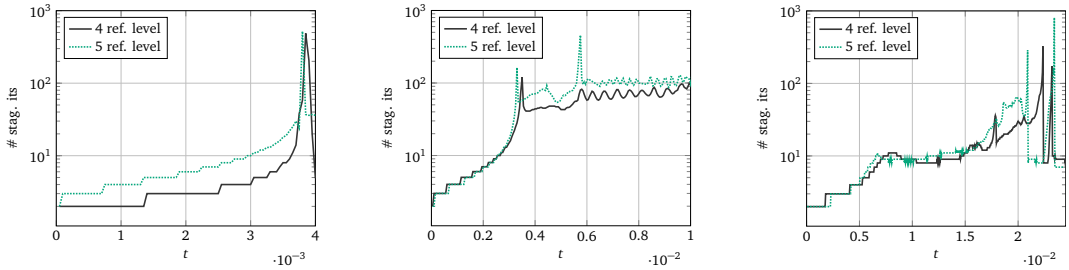

Figure 4.18: The number of iterations required by the staggered solution scheme over time-steps for tension, shear, tear test (from left to right). The performance of the algorithm is monitored for different discretization levels.

Table 4.3: An execution time of simulations for three fracture modes (approx. 782 thousand dofs) and conchoidal fracture example (approx. 1 million dofs). The experiments were performed using 5 processors. The time is measured in hours.

Example	Staggered scheme Time	Monolithic scheme TR Time	TR Red.(Stag.)	RMTR Time	RMTR Red.(Stag.)	RMTR Red.(TR)
Tension	9.06	6.41	29.24%	4.18	53.86%	34.79%
Shear	137.72	52.40	61.95%	21.32	84.52%	59.31%
Tear	123.15	48.16	60.89%	15.09	87.75%	68.67%
Conchoidal	N/A	14.73	N/A	10.86	N/A	26.27%

4.5.4 Scalability

In this section, we investigate both the strong and weak scaling properties of our implementation of the multilevel phase-field fracture simulation framework. Strong scaling examines the speedup for a problem with fixed size with respect to the number of processors/nodes. Weak scaling studies the speedup for a problem with increasing size with respect to the number of processors/nodes. The speedup is defined as

$$\text{speedup} = \frac{T_b}{T_p},$$

where T_b and T_p denote the computational time required by a base experiment and the experiment with p cores/nodes, respectively. Furthermore, we also study the parallel efficiency defined as

$$\text{efficiency} = \frac{T_b M_b}{T_p M_p},$$

where M_b, and M_p describe the number of nodes used for base experiment and the experiment with p cores/nodes, respectively.

All results reported in this section are obtained by bootstrapping the main routine, V-cycle of the RMTR algorithm.

Small scale experiments

First, we investigate the scaling properties of our implementation of a multilevel phase-field fracture framework for unstructured meshes, see Section 4.3. We perform all experiments on the local cluster at the Institute of Computational Science (ICS), Università della Svizzera Italiana. The cluster consists of 42 compute nodes, each equipped with 2 Intel R E5-2650 v3 processor with a clock frequency of 2.60 GHz. Having 10 cores per processor leads to 20 cores per node. Memory per node is 64 GB.

As a first test, we consider a small scale tension test with approx. 782 thousand dofs. The base experiment uses one core. The obtained results are reported in Figure 4.20 in the middle, which demonstrates that our implementation gives rise to ideal scaling. We also illustrate the scaling of the staggered solution scheme and single-level TR. We remark that the scalability of both methods depends on the properties of the underlying linear solvers. Using the parallel sparse direct linear solver MUMPS, we demonstrate the scalability of the displacement and phase-field subproblems on Figure 4.19 on the left and Figure 4.19 in the middle, respectively. We notice that the speedup stops being ideal for more than 5 processors. In general, the results are worse for the phase-field subproblem, which is a consequence of fewer dofs compared to the displacement subproblem. This inconsistency in a number of dofs related to different subproblems prevents good scalability of the staggered scheme, even if iterative and highly scalable linear solvers are used.

Figure 4.19 on the right depicts the scaling results for a single-level TR. Although we use a highly scalable linear solver, a Projected Conjugate Gradient method with block Jacobi preconditioner, the scaling is not ideal. This is due to the fact, that the convergence of block Jacobi slows down as the number of blocks increases. In our set up (PETSc default), the numbers of blocks corresponds to the number of processors. Hence, the number of iterations to reach the desired tolerance increases with the number of processors.

Further, we consider a slightly larger example, i.e., a tension test with approx. 6 million dofs. During this experiment, we configure our RMTR as a V-cycle with 4 levels. We use a base experiment with 20 cores (1 node), which allows us to capture internode communication patterns. In this way, the number of processors and the internode communication cost increase proportionally. The obtained results are reported in Figure 4.20 on the right, which demonstrates that our implementation gives rise to almost ideal scaling up to 250 cores. For more than 250 processors, scaling is not ideal anymore, which is caused by an insufficient amount of dofs on the coarsest level. Further examination of the scaling properties of our RMTR method as well as performing tests with larger examples is left to future work.

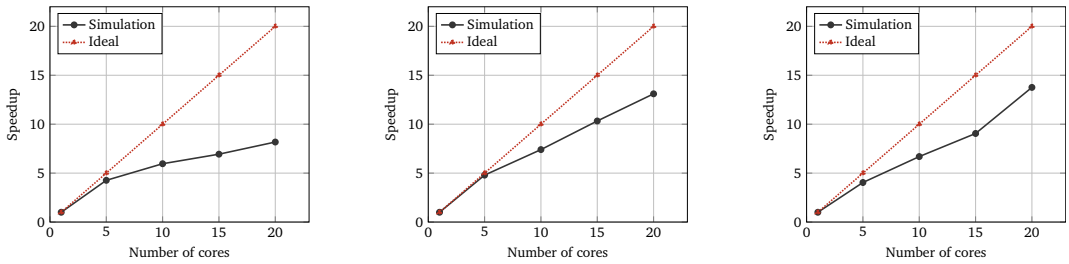

Figure 4.19: Strong scaling of *Left:* The semismooth Newton's method, phase-field subproblem, staggered scheme. *Middle:* The Newton's method, displacement subproblem, staggered scheme. *Right:* The trust-region method, monolithic scheme.

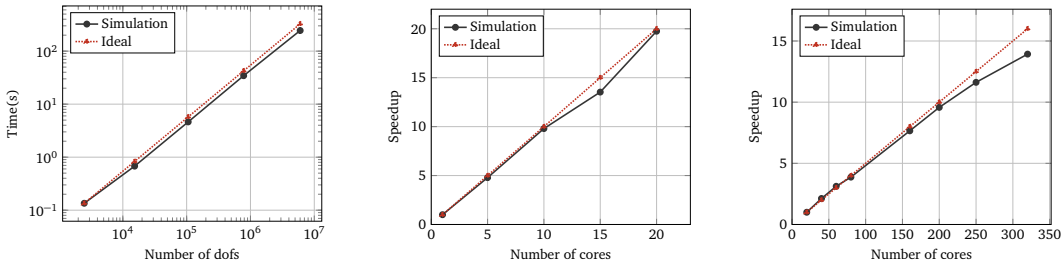

Figure 4.20: *Left:* Computational complexity of the RMTR method. *Middle:* Strong scaling of the RMTR with 5 levels performed on the tension test with approx. 782 thousand dofs. *Right:* Strong scaling of the RMTR with 4 levels performed on the tension test with approx. 6 million dofs.

Remark 19. *The measured times for both complexity and scalability tests contains also time spent for finite-element assembly.*

Large scale experiments

Furthermore, we investigate the scaling properties of our implementation of a multilevel phase-field fracture framework for structured meshes. All experiments reported in this section consider the pressurized fracture network in 3D with 100 initial randomly distributed fractures, see Section 4.4.3 for detailed description. We note, the experiments reported in this section are significantly larger than the experiments presented in the previous section for unstructured meshes. Indeed, here we study the parallel performance to the limits of the standard PETSc configuration with 32 bit indices.

We perform all experiments at the Swiss National Supercomputing Centre (CSCS) with the Piz Daint super-computer on XC50 compute nodes. Each XC50 compute node consists of one Intel Xeon E5-2690 v3 processor with a clock frequency of 2.60GHz. Each node uses 12 cores with hyper-threading disabled. The memory per node is 64 GB.

We conducted two strong scaling experiments, which consider RMTR setup with 4

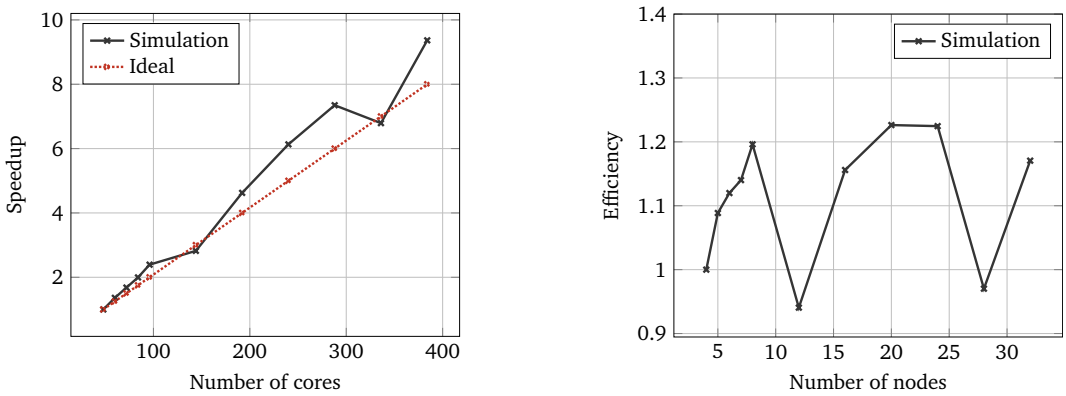

Figure 4.21: Strong scaling test performed using pressure induced fracture test with approx. 28.7 million dofs, RMTR setup with 4 levels. *Left:* Parallel speedup. *Right:* Parallel efficiency.

levels. The solution of trust-region subproblems is tackled by the projected successive coordinate method on all levels [GMTW08, GMS+10], except on the coarsest, where we use the MPRGP QP-solver [Dos09]. The first/small experiment considers approx. 28.7 million dofs on the finest level, while the second/large experiment considers approx. 242.7 million dofs on the finest level. We choose the number of nodes/cores of base experiments such that they fit into the node's RAM. The number of nodes/cores for a small base experiment is 4/48 nodes/cores, while we use 40/480 nodes/cores for a large experiment. Figures 4.21 and 4.22 demonstrate the obtained results. As we can see, the parallel efficiency oscillates depending on the number of nodes. We assume that this is due to slight imbalances which appear to have sometimes a bigger effect on the total runtime than with the same coarse level size but a different node count.

For weak scaling, we have set up the experiment with a coarse mesh of $10 \times 10 \times 10$ vertices on a single node and incremented then by doubling the vertices and adapting the dimensions to have a similar number of dofs on the coarsest level. In Figure 4.23 on the left, we can see the obtained speedup. Additionally, we also report an upper estimate of the parallel efficiency, see Figure 4.23 on the right. This estimate can be considered as "corrected" value where we multiply the speedup by the constant $c = \frac{n}{n_b M_p}$ with n being the number of dofs on the finest level and n_b being the number of dofs on the finest level for the experiment on one node. This correction factor is larger than 1 because doubling each dimension on the coarse level will increase the number of dofs by a factor larger than 8 on the finest level. For a setup with 4 levels, the number of dofs on the finest level in x-direction is $8n_x - 7$, similarly in y and z-direction, which results in a larger multiplication factor on the finest level than the multiplication factor on the coarse level.

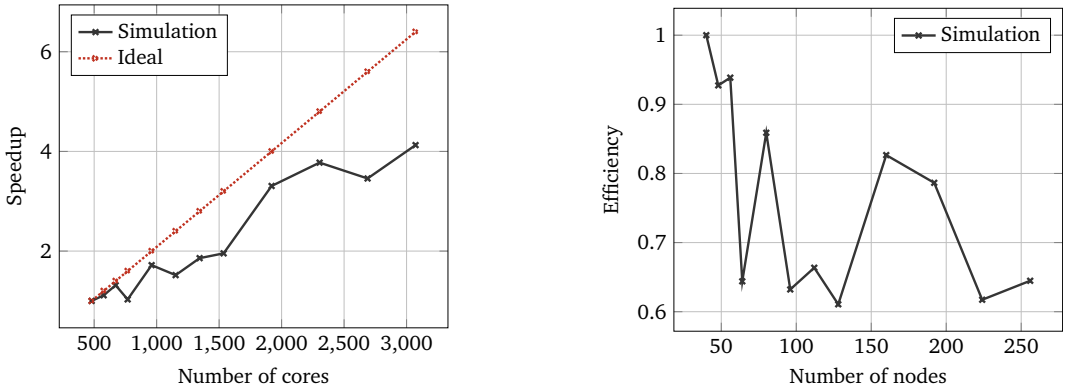

Figure 4.22: Strong scaling test performed using pressure induced fracture test with approx. 242.7 million dofs, RMTR setup with 4 levels. *Left:* Parallel speedup. *Right:* Parallel efficiency.

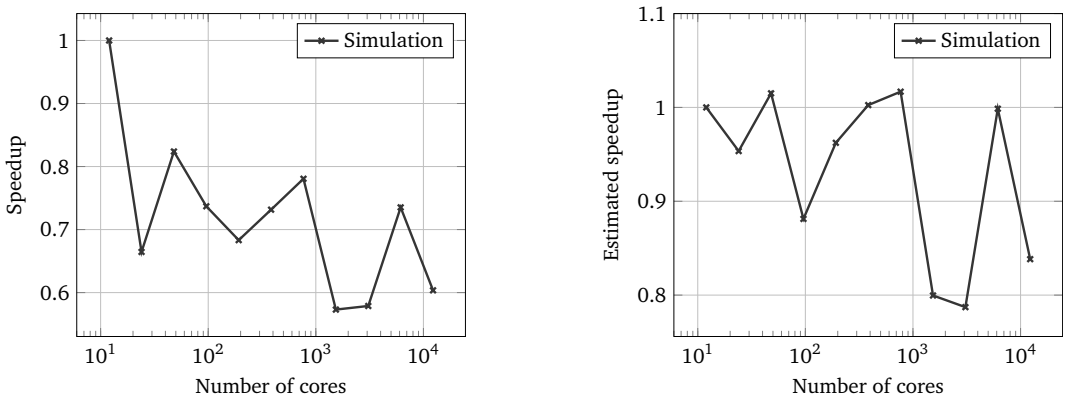

Figure 4.23: Weak scaling test performed using pressure induced fracture test, RMTR setup with 4 levels. *Left:* Parallel speedup. *Right:* Corrected/estimated parallel speedup.

4.6 Conclusion and outlook

In this chapter, we proposed a variant of the RMTR method for solving the monolithic systems arising from the phase-field fracture simulations. The proposed RMTR method employs novel level-dependent objective functions, which combine coarse-level discretizations with a fine-level representation of the fracture. The use of this novel formulation improved the performance of the RMTR method substantially. Furthermore, the novel RMTR method provided significant improvements in terms of computational time compared to the staggered solution scheme and the single-level trust-region method. The performed experiments also indicated that our implementation of the RMTR method is of optimal complexity and shows good scaling behavior.

Outlook

The work presented in this chapter can be extended in several ways. From a simulation point of view, the phase-field model can be expanded to take into account nonlinear elastic material and anisotropic crack growth. Furthermore, we could couple the phase-field fracture model with the poro-elasticity, which would allow for accurate simulations of hydraulic fracturing and reservoir processes. The overall efficiency of our simulation framework can be also enhanced by incorporating adaptivity in space and time.

From the solution strategy point of view, the obtained results could be improved by replacing currently employed trust-region subproblem solvers. The scaling properties of our implementation of the RMTR method could additionally be enhanced by reducing the communication cost required by the assembly on the coarser levels. This could be achieved by decreasing the number of processes used for performing assembly of gradient/Hessians on the lower level of the multilevel hierarchy.

In the end, we note that the current implementation of both discretization and model can be ported to GPU-based computing architectures. To achieve this goal, the memory requirements of our multilevel framework have to be lowered. This can be accomplished by using matrix-free variants of the RMTR method.

Chapter 5

Thin shell cloth simulations

Figure 5.1: Cloth animation of 1m × 1m piece of fabric. The opposite top corners are moved closer to each other, until their distance reaches 0.6m.

Cloth simulation is a well-established topic in computer graphics. It is essential for creating compelling animated characters, and it is also increasingly important for virtual try-on applications in e-commerce. Convincing visual results often require high levels of detail to capture wrinkles and folding patterns, but these results can be computationally expensive to obtain, and faster simulations are often achieved at the expense of being less realistic by using simplified physical models. The goal of this chapter is to accelerate cloth simulations without degrading the accuracy of the simulation. In particular, we focus on cloth modeled as orthotropic Kirchhoff–Love thin shells. However, the results obtained in this chapter also apply to other applications that depend on thin shell simulation with large strains and displacements.

The discretized partial differential equations for the Kirchhoff–Love model are strongly nonlinear and ill-conditioned. Given the fourth-order nature of the Kirchhoff–Love equations, the condition number of this linear system scales as $\mathcal{O}(h^{-4})$, where h represents the mesh size. Consequently, solving large systems arising from fine discretizations, which allow for high levels of detail, can be prohibitive. In this work, we employ the RMTR method to address both: nonlinearity and ill-conditioning, of the underlying optimization problems. To generate multiple levels of detail (multilevel hierarchy), we

The content of this chapter is extracted from published Manuscript [KKT19].

leverage the Catmull–Clark subdivision scheme and the corresponding subdivision finite elements. In feature films, cloth objects are typically modeled and rendered as subdivision surfaces, but often simulated using triangular meshes and linear finite elements. However, using different discretizations at different stages requires multiple stages of remeshing, which leads to a time-consuming set-up process and reduces the overall accuracy of the solutions. By using an isogeometric analysis approach based on subdivision surfaces, we avoid this problem as advocated by Reference [HCB05]. The subdivision scheme also provides a natural way to generate the multilevel hierarchy required for the RMTR method. In the end, we incorporate the subdivision scheme in four ways: to describe the geometry, to describe the displacements/solution, to connect different levels of the multilevel hierarchy, and to produce the final rendered surface.

In this chapter, we demonstrate that the RMTR method can be used effectively in the context of cloth simulations. The existing literature offers several linear geometric and algebraic multilevel methods, but to our knowledge, the RMTR method has not previously been used with isogeometric analysis for thin shells or cloth. We do not consider collision handling in this work, although it is important for cloth simulation, but it can be incorporated into the proposed method and we hope to address it in future work.

5.1 Cloth model

Cloth animation amounts to numerically simulating the motion of a piece of cloth in a 3D environment. A cloth object is often subjected to external forces such as gravity or wind, and additionally a set of prescribed boundary conditions that have to be satisfied at each time-step. Figure 5.1 illustrates a simple example, where a piece of cloth is subjected to the gravitational force and time-dependent boundary conditions.

Many cloth simulations in graphics are based on the seminal work by Baraff and Witkin [BW98] and later work by Bridson et al. [BFA02, BMF03]. Subdivision finite elements were first introduced by Cirak et al. [COS00] around the same time for linear elasticity and extended to nonlinear elasticity in Reference [CO01], and later this was used for cloth simulation in Reference [TWS06]. More recently, Clyde et al. [CTT17a] used a subdivision finite element discretization of cloth to estimate material parameters. In this work, we employ same cloth model as proposed in Reference [CTT17a].

5.1.1 Energy formulation

We model cloth objects by employing shell kinematics, as the cloth thickness is significantly smaller than its other geometric dimensions. The piece of cloth is modelled in terms of a 2D midsurface with constant thickness $\tau \in \mathbb{R}$. The deformed midsurface is represented by a map $\boldsymbol{x} : \omega \to \Omega$ from a 2D parameter space ω to the worldspace $\Omega \subset \mathbb{R}^3$. A volumetric cloth object is described by a map $\boldsymbol{r} : \omega^\tau \to \Omega^\tau$, where ω^τ is the 3D parameter space and $\Omega^\tau \subset \mathbb{R}^3$ is the region occupied by the shell. Both maps $\boldsymbol{x}$ and $\boldsymbol{r}$ are

time-dependent, i.e., $\boldsymbol{x} = \boldsymbol{x}(t)$ and $\boldsymbol{r} = \boldsymbol{r}(t)$, where t denotes time.

The kinematics of thin shells can be expressed by using the Kirchhoff-Love theory [Kir50], which assumes that straight lines initially perpendicular to the midsurface remain straight and perpendicular to the midsurface and retain their length during deformation. Equivalently, this means that the transverse shear strain must be zero. This assumption directly implies that

$$\boldsymbol{r}(\xi^1, \xi^2, \xi^3) = \boldsymbol{x}(\xi^1, \xi^2) + \xi^3 \boldsymbol{a}_3(\xi^1, \xi^2), \quad -\frac{\tau}{2} \leq \xi^3 \leq \frac{\tau}{2},$$

where $\boldsymbol{\xi} = (\xi^1, \xi^2, \xi^3)$ represent curvilinear coordinates and $\boldsymbol{a}_3(\xi^1, \xi^2)$ is the unit normal to the deformed midsurface.

The equilibrium configuration of the cloth can be found by minimizing the total mechanical energy

$$\psi^{\text{Mech}}(\boldsymbol{x}) := \underbrace{\frac{1}{2}\int_{\omega^\tau} \hat{\rho}\, \|\dot{\boldsymbol{x}} + \xi^3 \dot{\boldsymbol{a}}_3\|^2 \bar{J}\, d\boldsymbol{\xi}}_{:=\psi^{\text{Kin}}} + \underbrace{\int_{\omega^\tau} (\hat{\rho}\, \boldsymbol{g}^T \boldsymbol{r} + \psi^E\, \bar{J})\, d\boldsymbol{\xi}}_{:=\psi^{\text{Pot}}}, \tag{5.1}$$

which is defined as the sum of kinetic energy ψ^{Kin} and potential energy ψ^{Pot}. Here, we use overbar notation for quantities related to the undeformed configuration and overdot notation to indicate time derivatives. The symbol $\bar{J}$ denotes the Jacobian determinant $|\frac{\partial \bar{\boldsymbol{r}}}{\partial \boldsymbol{\xi}}|$, and $\hat{\rho} = \bar{\rho} \circ \bar{\boldsymbol{r}}$ is the density pullback. The total potential energy, ψ^{Pot}, consists of two terms. The first term represents a gravitational force defined by the gravity field $\boldsymbol{g} \in \mathbb{R}^3$. The second term, ψ^E, denotes the hyperelastic energy density which describes an orthotropic elastic material model and is defined as follows:

$$\psi^E(\boldsymbol{E}, \boldsymbol{O}, \boldsymbol{k}) := \frac{k_1}{2}\varphi_1(\tilde{E}_{11}^2) + k_2\varphi_2(\tilde{E}_{11}\tilde{E}_{22}) + \frac{k_3}{2}\varphi_3(\tilde{E}_{22}^2) + k_4\varphi_4(\tilde{E}_{12}^2), \tag{5.2}$$

where the vector $\boldsymbol{k} = [k_1, k_2, k_3, k_4]$ contains the material parameters. The $\tilde{E}_{jj}$ represents an element of the reduced Green–Lagrange strain tensor $\tilde{\boldsymbol{E}} = \boldsymbol{O}^T \boldsymbol{E} \boldsymbol{O}$, where $\boldsymbol{O} = [\boldsymbol{o}_1, \boldsymbol{o}_2, \boldsymbol{o}_3]$ provides an orthotropic basis corresponding to the normalized material warp, weft, and normal directions. The functions φ_i describe nonlinear material response and are defined as

$$\varphi_i(u) = \sum_{i=1}^{d_j} \frac{\mu_{ji}}{\alpha_{ji}}((u+1)^{\alpha_{ji}} - 1),$$

where d_j, α_{ji}, and μ_{ji} are also material parameters. A detailed description of the described model (5.2) can be found in Reference [CTT17a, CTT17b]. In particular, it can be shown that we must have $\boldsymbol{x} \in \boldsymbol{H}^2(\omega \to \mathbb{R}^3)$ in order to satisfy the zero transverse shear strain constraint [KBLW09].

5.1.2 Minimization problem

The unknown deformation is defined as a minimizer of the following minimization problem:

$$\begin{aligned} \boldsymbol{x} = \underset{\boldsymbol{x} \in \boldsymbol{H}^2(\omega \to \mathbb{R}^3)}{\operatorname{argmin}} \quad & \psi^{\text{Mech}}(\boldsymbol{x}), \\ \text{subject to} \quad & \boldsymbol{x}(\xi^1, \xi^2) = \boldsymbol{b}(\xi^1, \xi^2) \quad \text{on } \partial\omega^D. \end{aligned} \tag{5.3}$$

The equality constraint in (5.3) enforces the boundary conditions $\boldsymbol{b}(\xi^1, \xi^2)$ for all parameter space points (ξ^1, ξ^2) along the boundary segment $\partial\omega^D \subseteq \partial\omega$.

Here, we impose the boundary conditions (5.3) by using a penalty method [Bab73]. In particular, we expand the objective function from (5.3) by adding a penalty term as

$$\tilde{\psi}^{\text{Mech}}(\boldsymbol{x}) := \psi^{\text{Mech}}(\boldsymbol{x}) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{b}\|^2_{L^2(\partial\omega^\tau)}, \tag{5.4}$$

where $\beta > 0$ is the penalty parameter, and $\|\cdot\|_{L^2(\partial\omega^\tau)}$ denotes the L^2-norm on the boundary. We employ $\beta = h^{-\sigma}$, where h represents the mesh size and $\sigma > 0$ is a constant [Bab73, BE86].

The solution $\boldsymbol{x}$ of the constrained minimization problem (5.3) is then approximated by the minimizer $\tilde{\boldsymbol{x}}$ of the following unconstrained minimization problem:

$$\boldsymbol{x} \approx \tilde{\boldsymbol{x}} = \underset{\tilde{\boldsymbol{x}} \in \boldsymbol{H}^2(\omega \to \mathbb{R}^3)}{\operatorname{argmin}} \quad \tilde{\psi}^{\text{Mech}}(\tilde{\boldsymbol{x}}, \boldsymbol{b}). \tag{5.5}$$

The solution $\tilde{\boldsymbol{x}}$ of (5.5) approximates $\boldsymbol{x}$ well for sufficiently large values of β, and $\tilde{\boldsymbol{x}} \to \boldsymbol{x}$ as $\beta \to \infty$. While the boundary conditions are not strictly satisfied, their violation does not affect the space of admissible solutions since the additional term in (5.5) is clearly finite when $\boldsymbol{x} \in \boldsymbol{H}^2 \subset \boldsymbol{L}^2$. However, the violation of the boundary conditions does influence convergence rates of the finite element method under consideration. Inspired by work in References [COS00, GT04], we satisfy the $\boldsymbol{H}^2$ requirement by using subdivision basis functions for the finite element discretization; cf. Section 5.2.1.

As an alternative to the penalty method, the boundary conditions could be imposed using the method of Lagrange multipliers [GT04, CTT17a] or by using Nitsche's method [JS09]. The use of the Lagrange multipliers leads to constrained minimization problems with a saddle point structure, which complicates the solution process, and the proper derivation of Nitsche's method for shells is beyond the scope of this work. Hence, we utilize the penalty method.

Properties of the minimization problem

The minimization of (5.5) is an optimization problem with the following optimality conditions:

Find $\boldsymbol{x} \in \boldsymbol{H}^2(\omega \to \mathbb{R}^3)$ satisfying the first-order optimality condition

$$\nabla_{\boldsymbol{x}} \tilde{\psi}^{\text{Mech}}(\boldsymbol{x}; \boldsymbol{v}) = 0 \quad \forall \boldsymbol{v} \in \boldsymbol{H}^2(\omega \to \mathbb{R}^3).$$

The directional derivative of the energy with respect to $\boldsymbol{x}$ is defined as

$$\begin{aligned}\nabla_{\boldsymbol{x}}\tilde{\psi}^{\text{Mech}}(\boldsymbol{x},\boldsymbol{b};\boldsymbol{v}) = \int_{\omega^\tau} \hat{\rho}\ \ddot{\boldsymbol{x}}^T \boldsymbol{v}\bar{J}d\xi - \int_{\omega^\tau}\Bigg(\frac{\partial\psi^E}{\partial\boldsymbol{x}_{,1}}\boldsymbol{v}_{,1} + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,2}}\boldsymbol{v}_{,2} + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,11}}\boldsymbol{v}_{,11} \\ + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,12}}\boldsymbol{v}_{,12} + \frac{\partial\psi^E}{\partial\boldsymbol{x}_{,22}}\boldsymbol{v}_{,22} + \hat{\rho}\boldsymbol{g}^T\boldsymbol{v}\Bigg)\bar{J}d\xi + \int_{\partial\omega^\tau}\beta\Big(\boldsymbol{x}-\boldsymbol{b}\Big)^T\boldsymbol{v}\bar{J}d\xi,\end{aligned} \tag{5.6}$$

where we use comma notation to denote partial derivatives, e.g., $\boldsymbol{x}_{,\alpha} = \frac{\partial\boldsymbol{x}}{\partial\xi^\alpha}$. The symbol $\ddot{\boldsymbol{x}}$ in (5.6) denotes the second derivative of $\boldsymbol{x}$ with respect to time t. For simplicity, we discard the term $\xi^3\dot{\boldsymbol{a}}_3$ from (5.1) in (5.6), as suggested in Reference [COS00, KBLW09, CTT17a]. A detailed derivation of the weak form (5.6) can be found in References [CTT17b, Cly17].

In order to solve the minimization problem (5.5) numerically, we discretize the Euler–Lagrange equations (5.6) in space by the finite element method (FEM) and in time by the implicit Euler method. Solving the minimization problem (5.5) is challenging, because the functional (5.1) is non-convex and may, therefore, admit many local minimizers. In addition, as mentioned in the introduction, the condition number of the resulting linear systems is of order $\mathcal{O}(h^{-4})$, where h represents the mesh size; see Reference [SF08, Theorem 5.1]. This condition number can in theory increase additionally due to the penalty term introduced in (5.4). This work addresses both non-convexity and ill-conditioning by employing the RMTR algorithm [GST08, GK09b].

We note that multilevel methods have already been applied in different contexts related to shell/subdivision/cloth simulations. For example, Green et al. [GTS02] show speed up in solution time for subdivision shell simulations by applying a geometric multigrid method. Tamstorf et al. [TJM15] apply smoothed aggregation algebraic multigrid to achieve speed up for cloth simulations without leveraging subdivision methods. In the context of shell problems, a full approximation scheme is applied in Reference [GT06]. Bandara et al. [BC18] employ a subdivision-based multilevel framework for a gradient-based shape optimization. However, none of the aforementioned methods offer global convergence guarantees.

5.2 Recursive multilevel trust-region method

We propose to solve optimization problem (5.3) using the RMTR method with the trust-region defined by ℓ_2-norm, see Chapter 2 for details. This offers several benefits. On one hand, trust-region–based globalization ensures convergence to first-order critical points. On the other hand, the multilevel framework addresses problems with ill-conditioning and many degrees of freedom.

In this section, we provide details regarding our variant of the RMTR method, developed for solving the thin-shell cloth simulations. More precisely, we discuss how to construct a multilevel hierarchy and the transfer operators using subdivision surfaces.

We also specify the choice of level-dependent objective functions and the multilevel-treatment of the trust-region constraints.

5.2.1 Multilevel hierarchy and transfer operators

As noted before, the Kirchhoff–Love formulation requires test and trial functions to be H^2 continuous [KBLW09]. Such smoothness can be guaranteed, for example, by employing Non-uniform rational basis splines (NURBS) [HCB05] or subdivision basis functions [COS00]. Subdivision schemes generally provide the advantage that they are defined on control meshes with arbitrary topology. For cloth simulations in the animation industry, a subdivision-based discretization provides an additional advantage since cloth objects are often also modeled and rendered as subdivision surfaces.

Many subdivision schemes are available, but the most widely used are the Catmull–Clark [CC78] and the Loop [Loo87] schemes. This work employs the Catmull–Clark subdivision scheme as implemented by Pixar's open-source package OpenSubdiv [Stu17]. The Catmull–Clark scheme was developed for quadrilateral meshes as a generalization of bi-cubic B-spline surfaces [CC78]. The resulting surfaces are C^2-continuous everywhere except at extraordinary vertices where they are only C^1-continuous.

In the following, the subdivision surfaces are used to represent both the geometry and the finite element space $\mathcal{X}^l$ on every level $l = 1, \dots, L$ in the multilevel hierarchy. Additionally, we employ subdivision-based transfer operators for transferring the data between subsequent levels.

Subdivision-based multilevel hierarchy

The subdivision schemes are developed from the refinability property of the B-spline basis functions [Zor00]. Consider the initial control mesh $\mathcal{T}^1$. We denote coarse-level control points by $\boldsymbol{q}_I^1 = (q_{I1}^1, q_{I2}^1, q_{I3}^1)^T \in \mathbb{R}^3$, where $I = 1, \dots, n^1$ and n^1 is the number of control points on the coarsest level.

Here, we slightly abuse a notation and use the symbol n^l to denote the number of control points on a given level l. For the time being, we assume that there exist B-spline basis functions $\{N_I^1\}$ such that the mid-surface $\boldsymbol{x}$ is represented as

$$\boldsymbol{x}(\boldsymbol{q}, \xi^1, \xi^2) = \sum_{I=1}^{n^1} \boldsymbol{q}_I^1 N_I^1(\xi^1, \xi^2). \tag{5.7}$$

The refinability property states that every B-spline basis function N_I^1 can be expressed as a translated and dilated copy of itself, i.e.,

$$N_I^1(\xi^1, \xi^2) = (\boldsymbol{z}_I^2)^T \boldsymbol{N}_I^1(2\xi^1, 2\xi^2), \tag{5.8}$$

where the vector $\boldsymbol{z}_I^2 \in \mathbb{R}^{n^s}$ defines weights for the linear combination of refined basis functions $\boldsymbol{N}_I^1(2\xi^1, 2\xi^2)$. The size n^s of $\boldsymbol{N}_I^1(2\xi^1, 2\xi^2)$ depends on the order of N_I^1.

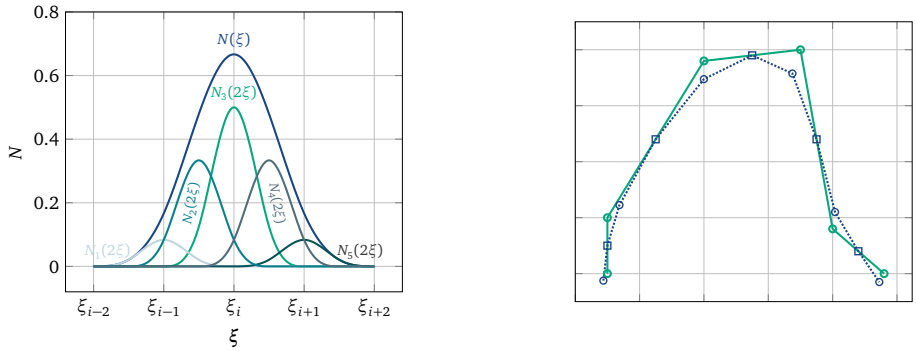

Figure 5.2: *Left:* Refinement of one dimensional (1D) cubic B-spline ($N(\xi)$). *Right:* A subdivision applied to the 1D coarse-level control polygon (green) results in refined control polygon (blue).

Figure 5.2 demonstrates the refinability property of a 1D cubic B-spline, which can be written as sum of five refined cubic B-splines.

The relation (5.8) allows us to change the basis in (5.7), such that

$$\boldsymbol{x}(\boldsymbol{q},\xi^1,\xi^2)=\sum_{I=1}^{n^1}\boldsymbol{q}_I^1 N_I^1=\sum_{I=1}^{n^1}\boldsymbol{q}_I^1((\boldsymbol{z}_I^2)^T\boldsymbol{N}_I^2)=\cdots=\sum_{I=1}^{n^1}\boldsymbol{q}_I^1((\boldsymbol{z}_I^l)^T\boldsymbol{N}_I^l). \tag{5.9}$$

The mid-surface $\boldsymbol{x}$ in (5.9) is defined in terms of the refined basis $\boldsymbol{N}_I^l=\boldsymbol{N}_I^1(2^l\xi^1,2^l\xi^2)$, which can be expressed by using matrix notation as

$$\boldsymbol{x}(\boldsymbol{q},\xi^1,\xi^2)=\mathbf{N}^1\boldsymbol{q}^1=(\mathbf{Z}^1)^T\mathbf{N}^2\boldsymbol{q}^1=\cdots=(\mathbf{Z}^l)^T\mathbf{N}^l\boldsymbol{q}^1,$$

where $(\mathbf{Z}^l)^T\in\mathbb{R}^{3n^l\times 3n^{l+1}}$ is the subdivision matrix containing entries from $\boldsymbol{z}_I^l$. Instead of applying subdivision weights $\mathbf{Z}^l$ to the basis functions $\mathbf{N}^l$, we can perform an alternative operation in terms of control points

$$\boldsymbol{q}_{Ii}^l=\boldsymbol{z}_I\boldsymbol{q}_{Ii}^{l-1},\quad \text{for}\quad i=1,2,3,\quad \boldsymbol{q}^l=\mathbf{Z}^l\boldsymbol{q}^{l-1}. \tag{5.10}$$

Reformulating (5.7) with the help of (5.10) results in

$$\boldsymbol{x}(\boldsymbol{q},\xi^1,\xi^2)=\mathbf{N}^1\boldsymbol{q}^1=\mathbf{N}^2\boldsymbol{q}^2=\cdots=\mathbf{N}^L\boldsymbol{q}^L. \tag{5.11}$$

The formulation in (5.11) suggests that the same mid-surface $\boldsymbol{x}$ can be represented by basis functions and control cages of the different refinement levels. This allows us to create the multilevel hierarchy required by the RMTR algorithm in a very natural way as

$$\boldsymbol{\mathcal{X}}^L(\mathcal{T}^L)\supset\boldsymbol{\mathcal{X}}^{L-1}(\mathcal{T}^{L-1})\supset\cdots\supset\boldsymbol{\mathcal{X}}^1(\mathcal{T}^1), \tag{5.12}$$

where the spaces associated with different subdivision levels are defined by

$$\boldsymbol{\mathcal{X}}^l(\mathcal{T}^l):=\text{span}\{N_I^l\boldsymbol{e}_j\},\quad\forall\ j=1,2,3,\quad\forall\ I=1,\ldots,n^l,$$

where $\boldsymbol{e}_j\in\mathbb{R}^3$ denotes Euclidean basis vector.

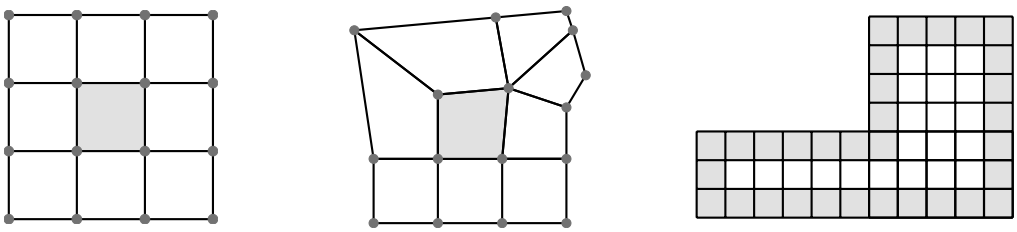

Figure 5.3: *Left:* Ordinary subdivision finite element (gray). A bi-cubic B-spline is defined by 16 control vertices (indicated by dots) in a one-ring neighborhood (white). *Middle:* Extraordinary subdivision finite element (gray) and its one-ring neighborhood (white) defined by 16 control vertices (indicated by dots). *Right:* Open control mesh with one layer of ghost faces (gray).

Subdivision FEM

On each level l of the multilevel hierarchy (5.12), the RMTR method requires the computation of the multilevel objective function, its gradient, and its Hessian. Following an isogeometric analysis approach, we approximate the domain geometry and the solution of (5.5) by the same basis functions, $\{N_I^l \boldsymbol{e}_j\}_{j=1,2,3;I=1,\dots,n^l}$. The finite element formulation is then obtained by inserting

$$\boldsymbol{x}(\boldsymbol{q}^l, \xi^1, \xi^2) = \sum_{I=1}^{n^l} \boldsymbol{q}_I^l N_I^l(\xi^1, \xi^2) \quad \text{and} \quad \boldsymbol{v} = \sum_{I=1}^{n^l} N_I^l \boldsymbol{e}_j$$

into (5.6). Integration in (5.6) is performed by numerical quadrature, which requires evaluation of the basis functions $\{N_I^l\}_{I=1}^{n^l}$ and their first- and second-order derivatives at any parameter location $(\xi^1, \xi^2) \in \omega$. Since the parametrization of a subdivision surface depends on the neighborhood of a given face F, we split the faces of the control cage into two groups: ordinary and extraordinary. An ordinary face only contains ordinary vertices defined as vertices with valence $\vartheta = 4$ (i.e., four incident edges), while an extraordinary face contains at least one extraordinary vertex, i.e., a vertex with valence $\vartheta \neq 4$.

For ordinary patches as shown in Figure 5.3 on the left, the Catmull–Clark scheme reduces by design to a uniform bi-cubic B-spline surface. These regions are easy to evaluate since the B-spline basis functions are polynomials. The surface patch corresponding to a given face F depends only on the one-ring neighborhood of the face. By one-ring neighborhood, we mean the set of all faces incident to the given face F. In the case of an ordinary patch, the one-ring is topologically a 4×4 rectangular grid and there are exactly 16 nonzero values of N_I^l.

The extraordinary surface patches, such as the one shown in Figure 5.3 in the middle, can be difficult and expensive to evaluate because the surface is no longer defined by a B-spline basis. However, for any point away from an extraordinary point, the evaluation can be performed by applying a finite number of subdivision steps, such that the

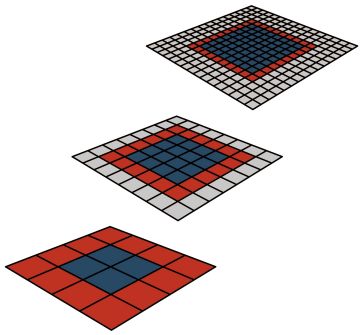

Figure 5.4: Multilevel hierarchy obtained by the Catmull–Clark subdivision scheme. Blue represents the parametrized surface. The ghost vertices associated with a given level are marked by red. The gray part represents the redundant part of the domain, which is removed.

given point is no longer in an extraordinary region. Stam [Sta98] demonstrated that the eigenstructure of the local subdivision matrix may be exploited in order to evaluate the subdivision basis functions and their derivatives even at extraordinary vertices.

Multilevel treatment of boundary conditions The use of subdivision basis functions requires careful treatment of the prescribed boundary conditions $\boldsymbol{b}(\xi^1, \xi^2)$ for the problem (5.5). This is due to the fact that the discretization of boundary conditions

$$\boldsymbol{b}(\boldsymbol{q}^l, \xi^1, \xi^2) = \sum_{I=1}^{n^l} \boldsymbol{q}_I^l N_I^l(\xi^1, \xi^2)$$

employs basis functions $\{N_I^l\}$, which have one-ring support. As a consequence, we have to attach a one-ring of artificial/ghost faces around the boundary; see Figure 5.3 on the right. It is important to notice that the mid-surface is not parametrized over ghost faces, but the ghost control points affect the solution on the neighboring interior faces.

Our implementation incorporates ghost faces into the design of the multilevel hierarchy as follows. During the modeling process, we add one layer of ghost faces to the coarse-level control cage $\mathcal{T}^1$. In the next step, we subdivide $\mathcal{T}^1$ and, as a result, obtain the control cage $\mathcal{T}^2$, which now contains two layers of ghost faces. Since the outer layer of ghost faces does not influence the geometry along the boundary, it can be discarded. In fact, removing unused vertices/faces saves a significant amount of memory. Figure 5.4 provides an example, where the initial control cage with ghost faces is subdivided two times. The parametrized surface, shown in blue, retains its domain on each level. The red color represents the one-ring of ghost vertices. On levels $l = 2, 3$, additional layers of ghost vertices (gray color) are removed.

Transfer operators via subdivision scheme

The exchange of data between adjacent levels of the multilevel hierarchy is ensured by transfer operators. As already introduced in Section 2.2, the RMTR algorithm employs three different types of transfer operators: prolongation, restriction, and projection. In this work, all three transfer operators are based on subdivision surfaces.

Prolongation and restriction

A solution of the discretized Euler–Lagrange equation (5.6) is a set of optimal fine-level control points $(\boldsymbol{q}^L)^* \in \mathbb{R}^{3n^L}$. Recalling Section 5.2.1, the relation between the fine-level and the coarse-level control points can be expressed by means of the subdivision matrix $\mathbf{Z}_l^{l+1} \in \mathbb{R}^{3n^{l+1}\times 3n^l}$ as

$$\boldsymbol{q}^{l+1} = \mathbf{Z}_l^{l+1}\, \boldsymbol{q}^l.$$

The non-zero elements of $\mathbf{Z}_l^{l+1}$ form the local subdivision rules, thus columns $\{3I, 3I+1, 3I+2\}$, where $I = 1,\dots,n_l$ of $\mathbf{Z}_l^{l+1}$ contain coefficients $\boldsymbol{z}_I$ from Equation (5.10). As a consequence, the global subdivision matrix $\mathbf{Z}_l^{l+1}$ is a natural choice for the prolongation of the primal variables (control points).

The assembly of $\mathbf{Z}_l^{l+1}$ can be performed using the Catmull–Clark subdivision scheme, which consists of two steps: refinement and smoothing. Refinement introduces new control points, called face points $\{\boldsymbol{f}_I^{l+1}\}$ or edge points $\{\boldsymbol{e}_I^{l+1}\}$. Smoothing repositions the coarse-level control points. The fine-level control points created in this way are denoted as $\{\boldsymbol{v}_I^{l+1}\}$. The fine-level control cage $\mathcal{T}^{l+1}$ contains all three types of control points, collectively denoted by $\boldsymbol{q}^{l+1} \in \mathbb{R}^{3n^{l+1}}$.

In the following, we describe subdivision rules for the Catmull–Clark scheme in the generic form, such that they can be applied to both ordinary and extraordinary control points. Figure 5.5 illustrates the subdivision steps, (5.13)–(5.16) explained below, for extraordinary control points. The local subdivision rules for the Catmull–Clark scheme are the following:

1. A face point $\boldsymbol{f}_I^{l+1}$ is created by taking an average of all control points surrounding a given element/face F_I, i.e.,

$$\boldsymbol{f}_I^{l+1} = \frac{1}{|F_I|} \sum_{\boldsymbol{v}_I^l \in F_I} \boldsymbol{v}_I^l. \tag{5.13}$$

2. A new edge point $\boldsymbol{e}_I^{l+1}$ is computed by averaging the endpoints $(\boldsymbol{e}_I^l, \boldsymbol{v}_I^l)$ of a given edge together with the face points $(\boldsymbol{f}_I^{l+1}, \boldsymbol{f}_{I+1}^{l+1})$ of the two faces incident to the edge as follows:

$$\boldsymbol{e}_I^{l+1} = \frac{\boldsymbol{e}_I^l + \boldsymbol{v}_I^l + \boldsymbol{f}_I^{l+1} + \boldsymbol{f}_{I+1}^{l+1}}{4}. \tag{5.14}$$

   The face points $\boldsymbol{f}_I^{l+1}$ and $\boldsymbol{f}_{I+1}^{l+1}$ are obtained by (5.13).

3. The fine-level position $\boldsymbol{v}_I^{l+1}$ of the coarse-level control point $\boldsymbol{v}_I^l$ is obtained as

$$\boldsymbol{v}_I^{l+1} = \frac{\vartheta-2}{\vartheta}\boldsymbol{v}_I^l + \frac{1}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{e}_I^l + \frac{1}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{f}_I^{l+1}, \tag{5.15}$$

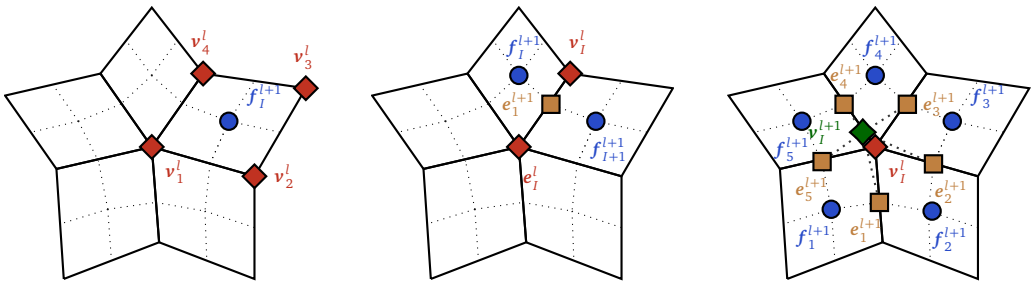

Figure 5.5: Catmull–Clark subdivision scheme for extraordinary control points: *Left:* Construction of a face point (blue circle). *Middle:* Computation of an edge point (brown rectangle). *Right:* Reposition of the coarse-level vertex (green diamond). Red diamond markers represent coarse-level control points.

where ϑ is the valence of $\boldsymbol{v}_I^l$. Employing (5.14), we can replace $\boldsymbol{e}_I^l$ in (5.15) with the newly created edge points $\boldsymbol{e}_I^{l+1}$ as

$$\boldsymbol{v}_I^{l+1} = \frac{\vartheta - 3}{\vartheta}\boldsymbol{v}_I^l + \frac{4}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{e}_I^{l+1} - \frac{1}{\vartheta^2}\sum_{I=0}^{\vartheta-1}\boldsymbol{f}_I^{l+1}. \tag{5.16}$$

The fine-level position $\boldsymbol{v}_I^{l+1}$ of $\boldsymbol{v}_I^l$ is then expressed as the weighted average of $\boldsymbol{v}_I^l$ and the newly created points in its one-ring neighborhood.

In addition to (5.13)–(5.16), subdivision on the boundaries requires a different set of rules; see Appendix A.4.1.

Projection by reverse subdivision In this section, we introduce a novel transfer operator $\mathbf{Q}_{l+1}^l \in \mathbb{R}^{3n^l \times 3n^{l+1}}$, which we then employ as a projection operator within the RMTR method. This novel operator $\mathbf{Q}_{l+1}^l$ is based on reverse subdivision. Following the work presented in Reference [LN06a], the reverse subdivision operator $\mathbf{Q}_{l+1}^l$ is affine and satisfies identity relation (2.6). Reverse subdivision reconstructs coarse-level control points $\boldsymbol{v}_I^l$ by using knowledge about the fine-level control points in the one-ring neighborhood of $\boldsymbol{v}_I^{l+1}$. The coarse-level point $\boldsymbol{v}_I^l$ can be expressed in a generic way as

$$\boldsymbol{v}_I^l = c_1\boldsymbol{v}_I^{l+1} + c_2\sum_{I=0}^{\vartheta-1}\boldsymbol{e}_I^{l+1} + c_3\sum_{I=0}^{\vartheta-1}\boldsymbol{f}_I^{l+1}, \tag{5.17}$$

where c_1, c_2, c_3 for the moment are coefficients to be determined, and ϑ denotes the valence of $\boldsymbol{v}_I^l$. We can substitute $\boldsymbol{v}_I^{l+1}$ and $\boldsymbol{e}_I^{l+1}$ in (5.17) by (5.15) and (5.14), respectively. Equation (5.17) can then be reformulated as follows:

$$\boldsymbol{v}_I^l = \left(\frac{\vartheta-2}{\vartheta}c_1 + \frac{\vartheta}{4}c_2\right)\boldsymbol{v}_I^l + \left(\frac{c_1}{\vartheta^2} + \frac{c_2}{4}\right)\sum_{I=0}^{\vartheta-1}\boldsymbol{e}_I^l + \left(\frac{c_1}{\vartheta^2} + \frac{c_2}{2} + c_3\right)\sum_{I=0}^{\vartheta-1}\boldsymbol{f}_I^{l+1}.$$

The coefficients c_1, c_2, c_3 can now be obtained by solving the following simple 3×3 system of equations

$$\begin{pmatrix} (\vartheta-2)/\vartheta & \vartheta/4 & 0 \\ 1/\vartheta^2 & 1/4 & 0 \\ 1/\vartheta^2 & 1/2 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{5.18}$$

The solution of (5.18) is

$$c_1 = \frac{\vartheta}{(\vartheta-3)}, \qquad c_2 = \frac{-4}{\vartheta(\vartheta-3)}, \qquad c_3 = \frac{1}{\vartheta(\vartheta-3)}. \tag{5.19}$$

The reverse subdivision formula is finally obtained by inserting (5.19) into (5.17). The formula (5.17) is straightforward to implement. The assembly of the reverse subdivision operator is based only on the information from the local neighborhood of the fine-level control point $\boldsymbol{v}_I^{l+1}$. This leads to a transfer operator with sparse structure, in contrast to the one obtained in (2.8).

The result in (5.17) cannot be applied for extraordinary vertices with valence $\vartheta = 3$ because the determinant of the system in (5.18) equals zero in that case. Luckily, an alternative reverse subdivision rule can be applied; see Appendix A.4.3. Additionally, the formula (5.17) cannot be used to reconstruct coarse-level ghost points. This is due to the fact that during the construction of the multilevel hierarchy (see Section 5.2.1), we delete one layer of the ghost faces on each level. Therefore, we do not have complete information about the one-ring neighborhood of $\boldsymbol{v}_I^{l+1}$ required by (5.17). We can overcome this difficulty by introducing a new set of reverse subdivision rules; see Appendix A.4.2.

Even though the results in this work are obtained using the Catmull–Clark scheme, the reverse subdivision operator also exists for the other subdivision schemes, e.g., the Loop scheme [SPSP03].

5.2.2 Level-dependent objective functions

Our RMTR method designed for solving cloth simulations employs level-dependent objective functions built using the first-order consistent additive model, see Section 2.4.2 for details.

5.2.3 Multilevel treatment of constraints

Since we employ a variant of RMTR method with a trust-region defined by the ℓ_2-norm, we can use techniques described in Section 2.5.3 to preserve fine-level trust-region constraint on coarser levels. The size of coarse-level corrections is then controlled by using trust-region radius update rules, see also Algorithm 5.

5.3 Implementation

In order to generate results presented in this section, we implemented the cloth model as well as the solution strategies inside of an animation package called Beast. Our multilevel simulation workflow is as follows: The simulation starts with the multilevel initialization. During this process, the algorithm generates the hierarchy of control cages $\{\mathcal{T}^l\}_{l=1,\dots,L}$ by subdividing the initial control cage $\mathcal{T}^1$ and assembling the transfer operators. For the assembly of the global subdivision matrix, we use the OpenSubdiv library [Stu17], while the assembly of the reverse subdivision operator is implemented separately. Note that the transfer operators are only assembled once since the topology of the control cages does not change during the simulation. The initialization phase also contains a pre-computation step, where we compute basis functions and their derivatives at quadrature points. These values only depend on the local topology, so we only do this for one ordinary face and each type of extraordinary face. The resulting stencils of basis function values are stored in a hash-table, which is later used during the solution process for assembly of the gradient and the Hessian on every level. Our current implementation supports extraordinary vertices with valence up to 99.

Once the initialization has been completed, the simulation continues into a time-stepping loop, which consists of pre-processing, nonlinear solve, and post-processing. Pre-processing and post-processing have to be executed on each level of the multilevel hierarchy. This ensures synchronization of different levels in time. For example, at each time-step, the animator prescribes a set of boundary conditions for the finest level. Those conditions need to be distributed to the coarser levels. Otherwise, the coarse-level models would not be suitable representations of the fine-level problem. The RMTR method has access to the assembly routines of the coarse-level models, which allows for construction of level-dependent objective functions.

Our implementation of this framework leverages Intel's Math Kernel Library library [Int07] for linear algebra.

5.4 Numerical results

In this section, we study the performance of the RMTR method using four scenarios, each using a $1\,\mathrm{m} \times 1\,\mathrm{m}$ square piece of cloth. The four scenarios are subject to different sets of boundary conditions:

- **Trampoline**. All four sides of the boundary are held fixed.
- **Drooping**. Two neighboring sides of the boundary are fixed, while the remaining two sides are free to fall.
- **Re-entrant corner**. A $0.3\,\mathrm{m} \times 0.3\,\mathrm{m}$ square is removed from the corner of the original piece of the cloth, and the edges adjacent to the removed area are held fixed. This example introduces a structural singularity due to the re-entrant corner.

Table 5.1: Material parameters for the denim material used in all experiments. The thickness τ is given in mm, the density ρ in kg/m^2, and k_i in MPa. All other parameters are dimensionless.

k_1	4.79	d_1	5	μ_{12}	-1482.1	μ_{33}	-185.7	α_{11}	-7.80	α_{32}	-12802
k_2	4.52	d_2	1	μ_{13}	276.43	μ_{34}	-45.82	α_{12}	10.80	α_{33}	17.91
k_3	9.03	d_3	5	μ_{14}	2407.67	μ_{35}	146.98	α_{13}	18.95	α_{34}	30.07
k_4	2.44	d_4	3	μ_{15}	3416.79	μ_{42}	0.79	α_{14}	-12.89	α_{35}	24.68
τ	0.66	ρ	0.4	μ_{32}	0.658	μ_{43}	11.08	α_{15}	1.61	α_{41}	2.95
								α_{21}	1	α_{42}	-2823.9
								α_{31}	12.71	α_{43}	4.29

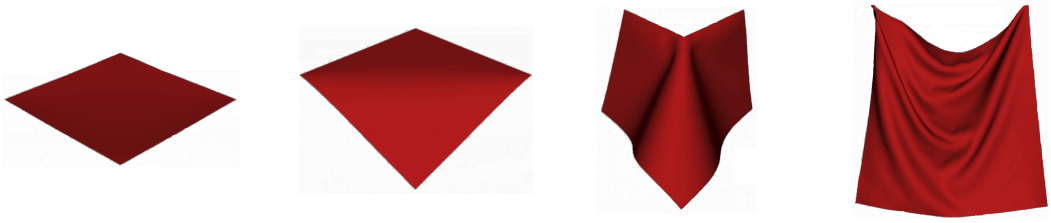

Figure 5.6: Test cases, from left to right: Trampoline, Drooping, Re-entrant corner, Drape.

- **Drape**. Time-dependent boundary conditions are applied by moving two opposite corners of the cloth together. Each corner is translated 20 cm in the direction of the opposite corner. This example is of interest because it contains a lot of wrinkles and folding patterns, which cannot be represented on the coarsest level.

All prescribed boundary conditions are applied just on the displacement field, while rotational degrees of freedom are unconstrained. The presented numerical examples employ vertices with valences 2, 3, and 4. Throughout the following we use a denim material given by the parameters in Table 5.1. Figure 5.6 depicts the simulation results.

Condition number and machine precision

As noted in Section 5.1.2, the condition number of the linear systems is of order $\mathcal{O}(h^{-4})$, where h represents the mesh size. Figure 5.7 on the left demonstrates how the condition number increases with respect to the number of degrees of freedom (dofs). As we can see, the condition number is always larger than 10^8.

Figure 5.7 on the right illustrates the effect of the penalty parameter β on the overall condition number. For this specific example, the overall condition number of the Hessian $\mathbf{H}$ is not affected by the penalty term β for $\beta < 10^5$. For $\beta > 10^5$, the overall condition number increases. This numerical behavior is in agreement with the theory discussed by Pospíšil [Pos15, Theorem 1.7.2]. In particular, let $\mathbf{M}$ denote the sec-

Level	# dofs	$\approx$ cond(**H**)
1	147	10^8
2	363	10^9
3	1,083	10^{10}
4	3,675	10^{11}
5	13,467	10^{12}
6	51,483	10^{13}

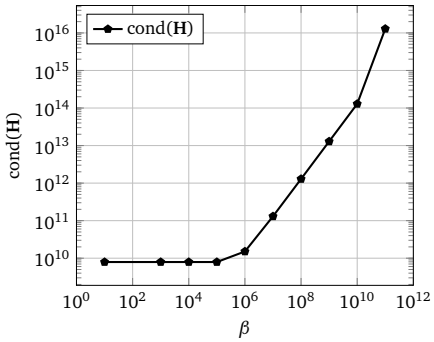

Figure 5.7: *Left:* Condition number as function of the number of dofs. *Right:* Condition number as function of penalty parameter β. The data are based on the trampoline example with 1,083 dofs and a time-step of dt = 1.

ond derivative of the penalty term from (5.5) and let $\mathbf{H}_{\text{mech}}$ denote the second derivative of the mechanical energy (5.1). Pospíšil shows that if cond($\mathbf{M}$) $\leq$ cond($\mathbf{H}_{\text{mech}}$), then cond($\mathbf{H}$) = cond($\mathbf{H}_{\text{mech}}$).

The presented numerical examples use $\beta = h^{-\sigma}$, where h is the mesh size and $\sigma = 5$. This particular choice provides a reasonable tradeoff between the accuracy with which the boundary conditions are imposed and the ill-conditioning of the arising linear systems.

Convergence to critical points and stopping criterion

The RMTR method applied to the minimization problem (5.5) converges to the first-order critical points since the energy functional $\tilde{\psi}^{\text{Mech}}$ is continuously differentiable, is bounded from below, and has uniformly bounded Hessians [GST08, Theorem 4.13], [GK09b, Theorem 4.7]. In order to identify the convergence numerically, we use the following stopping criterion:

$$\begin{aligned} &\|\boldsymbol{g}(\boldsymbol{x}_i)\| < 10^{-7} \text{ or} \\ &\big((|f(\boldsymbol{x}_{i-1}) - f(\boldsymbol{x}_i)| \text{ or } \|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\|) < 10^{-12} \text{ if } \Delta_i > 10^{-15}\big), \end{aligned} \tag{5.20}$$

where $\|\boldsymbol{g}(\boldsymbol{x}_i)\|$ is the norm of the gradient at the iterate $\boldsymbol{x}_i$, $|f(\boldsymbol{x}_{i-1}) - f(\boldsymbol{x}_i)|$ measures the difference in the objective function between two successive iterations, and $\|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i\|$ denotes correction size. The second part of (5.20) is activated only if the current trust-region radius Δ_i is bigger than 10^{-15}. In this case, the current iterate $\boldsymbol{x}_i$ is close to the minimizer $\boldsymbol{x}_*$ and the trust-region acts as Newton's method [GK09b, Theorem 4.10]. Unfortunately, the convergence behavior of Newton's method in floating point arithmetic is restricted by the limiting accuracy and the limiting gradient [Hig02,

Chapter 25]. The limiting accuracy is proportional to the condition number of the Hessian at the solution $\boldsymbol{x}_*$ and the accuracy with which gradient is evaluated [Tis01, Section 2.2]. The limiting gradient provides a lower bound on the norm of gradient [Tis01, Section 2.3]. In particular, $\|\boldsymbol{g}(\boldsymbol{x}_i)\|$ is bounded below by the error made in computing the gradient plus the term $\epsilon_m \|\mathbf{H}(\boldsymbol{x}_i)\| \|\boldsymbol{x}_i\|$, where ϵ_m denotes machine precision.

The cloth simulations considered in this work are severely ill-conditioned; see Section 5.4. In addition, the orthotropic elastic material model (5.2) describing the cloth behavior is defined as a sum of several polynomials. Hence, the evaluation of the gradient (5.6) is subjected to large rounding errors. For this reasons, it is of major importance to prevent the algorithm from meandering aimlessly, which we achieve by including the second condition in (5.20). In practice, $\|\boldsymbol{g}(\boldsymbol{x})\| \leq 5 \times 10^{-5}$ is satisfied by all presented numerical examples.

Choice of constrained QP solver

Both the TR and the RMTR methods require the solution of a constrained quadratic (QP) subproblem (1.5) within each iteration. A suitable QP solver must produce a correction that satisfies the sufficient descent condition (SDC). In the context of the cloth simulations considered here, we have tested the Dogleg method and ST-CG method. The ST-CG method suffers more from the severe ill-conditioning of the Hessian than the Dogleg method. As a consequence, the TR method configured with ST-CG reports failure even for examples with $1,083$ dofs. The failure is caused by exceeding 5×10^5 iterations without satisfying the stopping criterion (5.20). For this reason, the numerical results presented here for the single-level TR are obtained by using the Dogleg method.

Our choice of the constrained QP solver on each level of the multilevel hierarchy follows concepts well known from multigrid. First, we use a fact that the low-frequency components of the error appear more oscillatory on a coarser grid. Some iterative solvers can eliminate those high-frequency components of the error quickly, while leaving low-frequencies essentially unchanged [Saa03]. We employ ten iterations of ST-CG method on all levels except on the coarsest, since it is known to reduce the components of the gradient associated with large eigenvalues first [For06]. In addition, the ill-conditioning of the Hessian is not reflected in the Conjugate Gradient iterations until the gradients associated with the large eigenvalues have been made small [For06]. We also utilize the fact that the computational cost on the coarsest level is significantly lower than on the finer levels. Our RMTR setup, therefore, employs the Dogleg method with the sparse direct solver PARDISO [SG04] on the coarsest grid. In this way, we obtain the exact solution of (1.5), once the trust-region radius Δ_i is sufficiently large. This assumption is usually fulfilled close to the minimizer; see Reference [GK09b, Theorem 4.10] for details. Hence, the nonlinear multilevel method mimics the behavior of linear multigrid close to the minimizer.

Table 5.2: Choice of parameters used inside TR/RMTR algorithms.

Parameter	η_1	η_2	γ_1	γ_2	Δ_0^L	μ_1	μ_2
Value	0.1	0.75	0.5	2.0	1	1	1

5.4.1 Convergence study

During all simulations, the RMTR solver is configured to use a V-cycle with one pre- and one post-smoothing step and the parameters shown in Table 5.2.

Robustness with respect to time-stepping

Our first benchmark investigates the robustness of the RMTR method with respect to large time-steps. We consider examples with $1,083$ dofs and monitor the largest possible time-step, dt, that can be used during simulation, such that the solver successfully converges to the desired tolerance. We do not test time-steps larger than $\mathrm{dt} = 1$ sec, since at least one result is usually required per frame in an animation (with 1 frame $= 1/24$ sec).

We compare the robustness of the RMTR algorithm with the following solvers: single-level trust-region method (Algorithm 1), backtracking line-search (LS) Newton's method [NW06, Chapter 3], damped Newton's method, and the "linearized equation". The last two solvers are included because they are commonly used in the graphics community for cloth simulations. The damped Newton's method is Newton's method without any convergence control and with constant step-size $\alpha = 0.3$. The "linearized equation" [BW98] denotes a solution strategy, which exploits the fact that if the time-step dt is very small, then the nonlinearity in (5.6) becomes weaker (almost linear). Therefore, the solution of (5.6) can be found by simply solving a linear system of equations.

Figure 5.8 illustrates the results. We observe that the trust-region–based algorithms are very robust with respect to the large time-steps. For all test scenarios, the solvers report convergence independently of the size of dt. In contrast, backtracking line-search, scaled Newton's and "linearized equation" require smaller time-steps to maintain convergence. We also notice that with increased complexity of the example, the performance of those solvers decreases. For instance, the difference in the number of required time-steps between trust-region–based methods and the "linearized equation" approach is about 10 million for the drape example.

Number of nonlinear iterations per time-step

Next, we compare the convergence behavior of the RMTR method to its single-level counterpart, the TR method. Our analysis focuses on the ability of the methods to solve the cloth simulations as well as on their convergence behavior with respect to number of

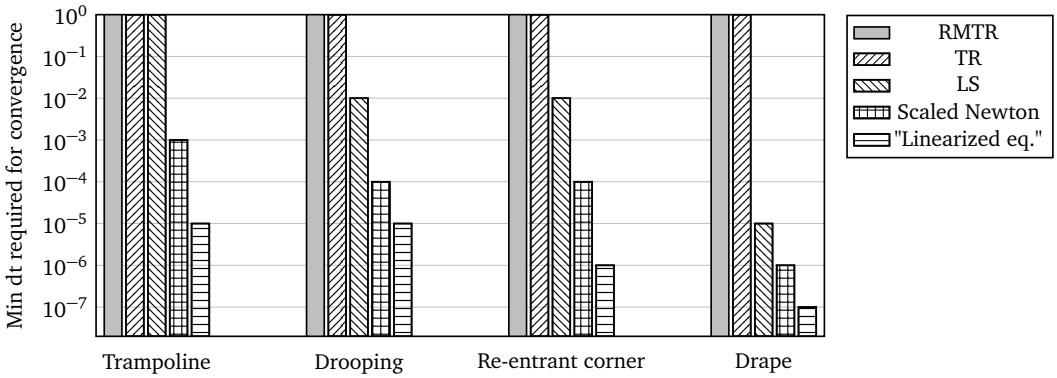

Figure 5.8: Robustness of solution strategies with respect to different time-steps. The benchmark is run with 1,083 dofs for each example.

Table 5.3: Number of nonlinear iterations for the single-level TR method as a function of the number of dofs. The time-step is dt = 1.

# dofs	147	363	1,083	3,675	13,467	51,483
Trampoline	1,385	29,426	34,456	—	—	—
Drooping	1,976	43,475	15,444	12,792	—	—
Re-entrant corner	8,293	13,965	18,873	47,536	334,463	—
Drape	7,331	53,654	175,132	—	—	—

dofs. In addition, we compare the convergence speed and the computational complexity of both methods. During these tests, we set the time-step to dt = 1.

Solvability We analyze the performance of the trust-region–based solvers by measuring the number of required nonlinear iterations/V-cycles. Table 5.3 summarizes the results obtained by using the single-level TR. As we can see, the number of iterations increases rapidly with the increased number of dofs. This is not surprising, as the condition number of the linear systems arising on each iteration grows rapidly. For example, the condition number for the test cases with approximately 50,000 dofs is on the order of 10^{13}. This is very close to one over the machine precision, and the trust-region solver reports failure because the desired stopping criterion (5.20) is never met. We also note that solvability becomes an issue already for problems with fewer than 4,000 dofs. In contrast, the RMTR method behaves as a preconditioner, and it is able to solve all our numerical examples, as shown in Table 5.4.

Convergence speed of RMTR Table 5.4 demonstrates that the number of V-cycles decreases with increased number of levels. Intuitively, this behavior can be explained by the

Table 5.4: Number of nonlinear V-cycles for the RMTR method as a function of the number of dofs. The time-step is dt = 1.

Example/Levels	2	3	4	5	6
Trampoline	633	150	81	88	52
Drooping	4,671	645	707	1,636	333
Re-entrant corner	3,188	1,508	282	124	30
Drape	9,732	7,653	5,718	792	423

fact that the setup of our RMTR method uses approximate constrained quadratic solvers on each level of the multilevel hierarchy. If more levels are used, the coarse-level models are better able also to capture the low-frequency parts of the error. This is caused by the fact that a trust-region method is also employed on the coarsest level. Depending on the trust-region radius, this might act more as a gradient method—or smoother—than as a Newton's step—or full coarse grid correction. Consequently, adding more levels improves convergence speed, because it ensures that a bigger part of the spectrum is covered.

The fact that the number of V-cycles required decreases as the number of levels is increased is consistent with the theoretical results obtained in Reference [GST08, Theorem 4.10]. The upper bound on the total number of successful iterations π produced by the RMTR method is defined as

$$\pi = \sum_{l=1}^{L} \pi^l \leq \frac{h^L(\boldsymbol{x}_0^L) - h_*}{\theta(\epsilon_g)}, \tag{5.21}$$

where the $h(\boldsymbol{x}_1^L)$ is the value of the objective function at the initial iterate. The symbol h_* denotes a constant, such that $h_* \leq h^L(\boldsymbol{x}_i^L)$, for every fine-level iterate $\boldsymbol{x}_i^L$. The constant $\theta(\epsilon_g)$ depends on the properties of the minimization problem, on the constants chosen inside of Algorithm 4, and on the desirable stopping tolerance ϵ_g. Interestingly, the definition of the constant $\theta(\epsilon_g)$ is dimension-independent, which makes the complexity bound (5.21) mesh-independent [GST08, Theorem 4.10]. While the upper bound in (5.21) is constant, the total number of successful iterations π is defined as a sum of all successful iterations over all levels. Consequently, the successful coarse-level iterates help to decrease the number of expensive, fine-level iterates. For instance, for a single-level method, i.e., $L = 1$, π represents the successful iterates produced only on the finest level. However, for a three-grid method, i.e., $L = 3$, π contains the sum of the successful coarse-level and fine-level iterates. Note that an iteration is successful in the multilevel context if it is also accepted on the fine level.

Although we expect fewer V-cycles as the number of levels increases, we note that for the Drooping example, the number of V-cycles increases for four and five levels. This does not mean that the bound (5.21) is violated, since Table 5.4 reports the total num-

Table 5.5: Number of nonlinear iterations required by TR versus the number of nonlinear V-cycles required by the RMTR method. The simulation time-step is dt = 1.

Method	TR	RMTR	TR	RMTR	TR	RMTR
Example/ # dofs	1,083		3,675		51,483	
Trampoline	34,456	150	—	81	—	52
Drooping	15,444	645	12,792	707	—	333
Re-entrant corner	18,873	1,508	47,536	282	—	30
Drape	175,132	7,653	—	5,718	—	423

ber of V-cycles rather than π. The total number of V-cycles might increase, for instance, if the prolongated coarse-level corrections are not accepted by the finest level. Alternatively, the trust-region radius might be too small, which causes termination of the recursion before reaching the coarsest level. Despite the fact that the use of the globalization strategy might seem to slow down convergence, its use is crucial as it provides convergence guarantees.

Computational cost: TR versus RMTR The performance of the TR method deteriorates with increasing number of dofs, while the performance of the RMTR method improves. Table 5.5 provides a side-by-side comparison of the results. In order to compare the computational cost of TR and RMTR methods, we focus on the computationally most expensive parts of both algorithms: assembly of the derivatives (gradient and Hessian) and computation of the (approximate) solution of the QP subproblem (1.5).

First, we estimate the computational cost required by the assembly routines. Due to the sparsity of the Hessian matrices, we assume that the cost on each level, W^l, is proportional to the number of control points, i.e., $W^l = Cn^l$, where C is a constant. Let one *work unit*, W^L, be defined as the computational cost of matrix assembly on the finest level. Assuming a coarsening factor of 2 between each level, the corresponding cost on coarser levels is then given as $W^l = 2^{-2(L-l)}W^L$. From this, the total assembly cost W measured in work units can be computed as

$$W \approx \sum_{l=1}^{L} Q^l W^l = \sum_{l=1}^{L} 2^{-2(L-l)} Q^l \, W^L, \tag{5.22}$$

where Q^l denotes the number of assembly calls on the level l. Let one iteration in the RMTR solver correspond to one V-cycle. The average number of work units per iteration is then simply computed as

$$W_{\text{avg}} = \frac{W}{\#\text{iterations}}. \tag{5.23}$$

Table 5.6 shows the comparison of W and W_{avg} for the TR and RMTR methods based on our examples with 1,083 dofs. This resolution has been chosen since it is the finest

Table 5.6: The total and average computational cost measured in work units required by the assembly routines for the TR and RMTR methods. These numbers represent W and W_{avg}, respectively. The experiment is performed for the problem with $1,083$ dofs. The RMTR method is set up with three levels.

	Total cost			Average cost	
Example/Method	TR	RMTR	TR/RMTR	TR	RMTR
Trampoline	$30,981$	370	83.73	0.89	2.47
Drooping	$13,858$	$1,612$	8.59	0.90	2.50
Re-entrant corner	$16,632$	$3,788$	4.39	0.88	2.51
Drape	$120,089$	$19,163$	6.27	0.69	2.50

resolution for which the single-level TR method converges. We notice that the TR method requires less than one work unit per iteration, which is due to the fact that the derivatives are recomputed only after a successful iteration, e.g., when the trial point is accepted. Given our RMTR setup with one pre-/post-smoothing step, the assembly is performed every time we enter the given level. However, the number of assembly calls on each level of the RMTR method can vary, as the recursion may be terminated before reaching a given level. In practice, we see that the RMTR method requires roughly 2.5 work units per V-cycle, and based on (5.22) we compute that the cost of one V-cycle of the RMTR method is approximately 2.8 times higher than the cost of one TR iteration. However, in terms of total computational cost, the RMTR method is significantly more efficient than the single-level TR (from 4.4 to 83.7 times). This is not surprising since the RMTR method requires a considerably lower number of V-cycles than the single-level TR iterations to achieve convergence.

Next, we estimate the computational cost of the QP solvers. We follow the same reasoning as for the assembly cost and employ formulas (5.22) and (5.23). The quantities W^l and Q^l now represent the cost of performing one QP solve and the number of calls into the QP solver on a given level l, respectively. We note that the assumption that the cost is proportional to the size of the problem is not valid for a generic QP solver. However, our setup of the RMTR method employs a constant number of 10 iterations of the ST-CG method per smoothing step, each with cost $O(n^l)$. On the coarsest level, $l = 1$, we employ the Dogleg method with a sparse direct linear solver. Although the complexity of sparse direct solver is in general higher than $O(n^1)$, the difference can be hidden in the constant inside of the complexity bound, as n^1 is very small. In particular, all presented examples employ a problem with 147 dofs on the coarsest grid. Additionally, the contribution from the coarsest grid is scaled by the coarsening factor 2^{-2L}, which makes the coarse grid cost almost negligible.

Table 5.7 compares the computational cost of the QP solvers for the RMTR and the TR method. We see that the TR method requires one work unit per iteration because the

Table 5.7: The computational cost measured in work units for the QP solvers for the TR and RMTR methods. These numbers represent W and W_{avg}, respectively. The experiment is performed for the problems with 1,083 dofs. The RMTR method is set up with three levels.

	Total cost			Average cost	
Example/Method	TR	RMTR	TR/RMTR	TR	RMTR
Trampoline	34,456	370	93.12	1	2.47
Drooping	15,444	1,612	9.58	1	2.50
Re-entrant corner	18,873	3,788	4.98	1	2.51
Drape	175,132	19,163	9.14	1	2.50

Table 5.8: Number of V-cycles required by the RMTR method to converge for examples with 3,675 dofs. The RMTR method is configured with four levels.

Example/operator	Least square fit	Reverse subdivision
Trampoline	84	81
Drooping	659	707
Reentrant corner	277	282
Drape	5,691	5,718

QP solver is used once per iteration. In contrast, the RMTR method requires around 2.5 work units per V-cycle. In spite of that the RMTR method is significantly more efficient in terms of the total computational cost (from 5 to 93 times). Additionally, we note that this comparison strongly favors the single-level TR method since TR employs the Dogleg method, which is computationally more expensive than 10 steps of ST-CG. We could make the comparison fairer by replacing W^L in (5.22) with the complexity bound of the direct solver. However, this would only enhance the efficiency of the RMTR method compared to the single-level TR method.

Reverse subdivision operator versus least square fit

Finally, we evaluate the quality of the reverse subdivision operator (5.17). Our study compares the performance of the RMTR method with the reverse subdivision operator versus the RMTR method setup with the projection operator obtained by (2.9). Table 5.8 shows the results in terms of number of V-cycles. We see that the difference between the number of required nonlinear V-cycles is negligible. However, the reverse subdivision operator offers significant benefits in terms of computational complexity and memory requirements.

5.5 Conclusion and outlook

In this chapter, we proposed a variant of the RMTR method for the thin shell cloth simulations. The presented multilevel method is based on subdivision surfaces, which are also used for the construction of prolongation and restriction operators. Additionally, we incorporated a reverse subdivision operator for transferring the iterates from the fine levels to coarser levels. The novel use of this operator provides a computationally efficient alternative to the least-square-projection operator. Our numerical examples demonstrate the robustness of the proposed RMTR method with respect to large time-steps. A comparison with a single-level TR method demonstrates a reduction in the number of iterations by several orders of magnitude. In addition, we have shown that for problems with more than approx. $15,000$ dofs, the single-level trust-region method does not converge, while the RMTR method does.

Outlook

The results obtained in this chapter could be substantially improved if we carry out all numerical computations using higher precision. In this way, we could evaluate the function values/its derivatives and solve the trust-region subproblems more precisely.

The convergence properties of the RMTR method could be enhanced by incorporating different trust-region subproblem solvers. Furthermore, our simulation framework could be advanced by including the contact/collision handling, as this is also an essential component for cloth simulations. This would require several adjustments for both the simulation setup and the solution strategy. In particular, the RMTR method would have to be extended to handle non-linear inequality constraints.

Chapter 6

Training of deep residual networks

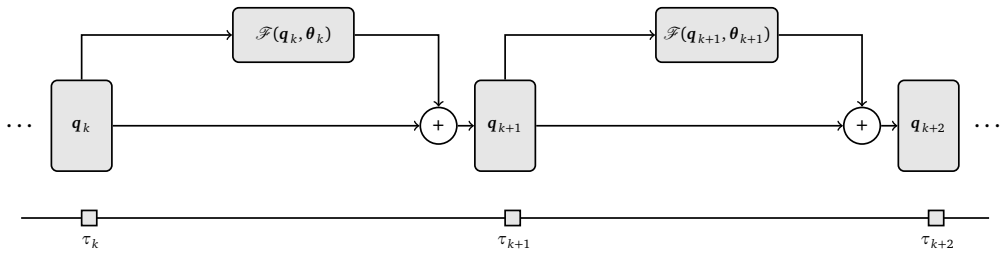

Figure 6.1: Deep residual network.

Deep residual networks or ResNets are widely used neural network architectures. They demonstrate state-of-the-art performance in complex statistical learning tasks with applications in various fields, such as computer vision [JCJ+17, CPK+17], or speech recognition [WSC+16, XWA+18]. The popularity of ResNets originates from their remarkable performance in the ImageNet [RDS+15] and the MS COCO [LMB+14] image recognition competitions.

A major drawback of very deep ResNets is their long training time. To mitigate this issue, different strategies have been proposed, for example networks with stochastic depth [HSL+16], mollifying networks [GMVB16], spatially adaptive architectures [FCZ+17], or multilevel parameter initialization strategies [HRHJ18, CMH+17]. In this chapter, we propose to accelerate the training of ResNets using a variant of the RMTR method [GST08, GK09b]. Our design of the proposed RMTR method is motivated by the following observations.

The training of ResNets is typically performed using variants of the stochastic gradient (SGD) method [RM51], which construct search directions using unbiased gradient estimators. Although these methods have a low computational cost per iteration, they often exhibit poor convergence properties. In addition, the convergence properties of

The content of this chapter is extracted from Manuscript [KK20a] (in preparation).

the SGD method depend on the choice of hyper-parameters. More precisely, it is important to carefully select the sequence of diminishing step-sizes to ensure convergence to a solution. To reduce the dependency of the solution method on the hyper-parameters, we propose a trust-region based solution strategy. The sequence of step-sizes is therefore determined automatically by the trust-region method [CGT00].

The network depth is of paramount importance for achieving the necessary approximation properties of the model [HG91, SZ14]. However, very deep networks are computationally expensive to train, as the cost of forward-backward propagation scales linearly with respect to the number of parameters [CR95]. To lower the computational cost, we propose to use a multilevel variant of the trust-region method. The multilevel method takes advantage of a multilevel hierarchy of auxiliary networks with different depths. The training of the deepest network is then accelerated by internally training the shallower networks. In this work, we construct a multilevel hierarchy of auxiliary networks by leveraging the emerging dynamical system's viewpoint [HRHJ18, Wei17], which casts ResNet as the discretization of an initial value problem. The training process is then formulated as the minimization of a time-dependent optimal control problem. As a consequence, we can obtain a hierarchy of ResNets with different depths by discretizing the same optimal control problem with different discretization parameters (time-steps).

We highlight the fact that the underlying optimization problem is non-convex. Hence, its minimization admits multiple local minimizers. We aim to find a solution, i.e., a set of parameters, which generalizes well, i.e., correctly classifies previously unseen samples. It has been observed in practice, that flat minimizers generalize better than the sharp minimizers [KMN+16, GVY+18, HHS17]. A sharp minimizer is characterized by a rapid increase in the value of the objective function in its small neighborhood. In contrast, a function value varies slowly in a relatively large neighborhood of a flat minimizer. The study provided in Reference [KMN+16] demonstrates that the large-batch/deterministic methods tend to be attracted to sharp minimizers. Instead, small-batch methods tend to be more exploratory, which helps them to escape basins of attraction of sharp minimizers and converge to flat minimizers. However, there are practical reasons, why large-batch methods should be employed. For example, they enable faster convergence in the local neighborhood of a minimizer [KMN+16]. Moreover, large-batch methods use computational resources more efficiently, e.g., by decreasing data movement between a CPU and a GPU device. In this work, we take advantage of both small-batch and large-batch techniques by using the RMTR method in conjunction with the dynamic sample size (DSS) strategy. This gives rise to the hybrid stochastic-deterministic multilevel method, which takes advantage of small-batches at the beginning of the training process. As training progresses, the mini-batch size is adaptively increased, which enables faster convergence to a solution.

The use of multilevel methods to enhance the training of ResNets has been investigated recently by several authors. For example, Haber et al. proposed two multilevel training approaches in Reference [HRHJ18]. In the first approach, the multilevel hier-

archy was created by changing an image resolution. In the second approach, the multilevel hierarchy was constructed using the dynamical system's viewpoint. The second approach was later studied in detail in References [CMH+17, CGS19]. Here, we point out that these methods took advantage of the multilevel hierarchy only in order to gradually initialize the network parameters. Furthermore, Wu et al. [WGH+20] proposed a multilevel method for training videos. The multilevel methods were also explored in the context of layer-parallel training in References [GRS+18, KSJ+20]. In the end, we note that a variant of the MG/OPT method was proposed in Reference [GMKK20]. Similar to the presented RMTR method, the proposed MG/OPT method utilized the dynamical system's viewpoint in order to construct a multilevel hierarchy. In contrast to our RMTR method, the devised MG/OPT method relied on a large set of hyper-parameters. More precisely, a learning rate had to be selected carefully on each level of the multilevel hierarchy, in order to ensure convergence. In addition, none of the aforementioned methods incorporates curvature information nor guarantees global convergence.

6.1 Supervised learning

In this section, we provide a brief overview of deep residual networks (ResNets) in the context of supervised classification. Our presentation is based on the dynamical system's viewpoint, which interprets the ResNet as a discretization of the continuous initial value problem [CMH+17, Wei17]. We consider a dataset $\mathcal{D} = \{(\boldsymbol{x}_j, \mathbf{c}_j)\}_{j=1}^{p}$, which contains p samples. Each sample consists of an input features $\boldsymbol{x}_j \in \mathbb{R}^q$ and corresponding label $\mathbf{c}_j \in \mathbb{R}^m$. Here, the symbol q denotes the size of the feature vector and m denotes the number of output classes. The i-th component of vector $\mathbf{c}_j$ corresponds to the probability that sample $\boldsymbol{x}_j$ belongs to the i-th class.

6.1.1 Classification as an optimal control problem

The main idea behind supervised learning is to construct a model $f_m : \mathbb{R}^q \to \mathbb{R}^m$, which captures a relationship between input and output for a given dataset $\mathcal{D}$. The model function f_m is usually obtained by composing the forward propagation function $f_p : \mathbb{R}^q \to \mathbb{R}^v$ with the hypothesis function $\mathscr{H} : \mathbb{R}^m \to \mathbb{R}^m$, thus as

$$f_m(\boldsymbol{x}) = \mathscr{H}(\mathbf{W}_K f_p(\boldsymbol{x}) + \boldsymbol{b}_K).$$

The hypothesis function $\mathscr{H}$ predicts the class label probabilities using the output of the forward propagation $f_p(\boldsymbol{x})$ by employing a set of parameters $\boldsymbol{\theta}_K = \{\mathbf{W}_K, \boldsymbol{b}_K\}$, where $\mathbf{W}_K \in \mathbb{R}^{m \times v}$ and $\boldsymbol{b}_K \in \mathbb{R}^m$. The forward propagation function f_p filters input features in a nonlinear manner. Its form is usually defined by the network architecture. Here, we leverage the dynamical system's viewpoint, where the forward propagation f_p is obtained by discretizing a continuous initial value problem.

In particular, let us consider the dynamical system

$$\begin{aligned} \partial_t \boldsymbol{q}(t) &= \mathscr{F}(\boldsymbol{q}(t), \boldsymbol{\theta}(t)), \qquad \forall t \in [0, T], \\ \boldsymbol{q}(0) &= \mathbf{Q}\boldsymbol{x}, \end{aligned} \tag{6.1}$$

where $\boldsymbol{\theta}(t) : [0, T] \to \mathbb{R}^d$ are time-dependent control variables and $\boldsymbol{q}(t) \in \mathbb{R}^\nu$ denotes system's state for all $t \in [0, T]$. The dynamical system (6.1) continuously transforms input $\boldsymbol{x}$ into the output $\boldsymbol{q}(T)$, while the time-dependent controls $\boldsymbol{\theta}(t)$ define the behavior of the system. The initial condition in (6.1) maps input $\boldsymbol{x} \in \mathbb{R}^q$ into the space of the dynamics with the help of the linear operator $\mathbf{Q} \in \mathbb{R}^{\nu \times q}$, e.g., $\boldsymbol{q}(0) := \mathbf{Q}\boldsymbol{x}$. The form of function $\mathscr{F} : \mathbb{R}^\nu \times \mathbb{R}^d \to \mathbb{R}^\nu$ is chosen as part of deep network architecture [HZRS16a].

Remark 20. *The residual block, function $\mathscr{F}$, and its derivatives have to satisfy certain assumptions, so that solution of (6.1) exists for any $\boldsymbol{\theta}$, see Reference [Cla05] for details.*

Now, we can formulate the supervised learning (classification) problem as a continuous optimal control problem [HR17], defined as follows:

$$\begin{aligned} \underset{\boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\theta}_K}{\text{minimize}} \quad & \frac{1}{p} \sum_{j=1}^{p} \ell(\mathscr{H}(\mathbf{W}_K \boldsymbol{q}_j(T) + \boldsymbol{b}_K), \mathbf{c}_j) + \int_0^T \mathscr{R}(\boldsymbol{\theta}(t), \boldsymbol{\theta}_K), \\ \text{subject to} \quad & \partial_t \boldsymbol{q}_j(t) = \mathscr{F}(\boldsymbol{q}_j(t), \boldsymbol{\theta}(t)), \\ & \boldsymbol{q}_j(0) = \mathbf{Q}\boldsymbol{x}_j, \end{aligned} \tag{6.2}$$

where $\mathscr{R}$ denotes a regularization function and $\boldsymbol{q}_j(T) \in \mathbb{R}^m$ symbolizes the output of the dynamical system (6.1), given sample $\boldsymbol{x}_j$. For each sample $\boldsymbol{x}_j$, a loss function $\ell : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ measures the deviation of the predicted output, i.e., result of hypothesis function, from the known label $\mathbf{c}_j$. An exact form of loss function depends on the problem at hand. In this work, we focus on multinomial classification problems. Therefore, we employ cross-entropy loss function together with softmax hypothesis function [GBC16].

Remark 21. *The regularization function $\mathscr{R}$ in (6.2) ensures existence and regularity of the parameters [HR17].*

Remark 22. *In this section, we consider only ResNets with constant width. More complex and practical scenarios will be considered in Section 6.2.1.*

6.1.2 Discrete minimization problem

We can solve (6.2) numerically by discretizing in time. Let us consider the time-grid $0 = \tau_0 < \cdots < \tau_K = T$ of $K+1$ uniformly distributed time points. Given a uniform time-step $\Delta_t := T/K$, the k-th time point is defined as $\tau_k := \Delta_t k$. Now, we can approximate controls and states at particular time τ_k as $\boldsymbol{\theta}_k \approx \boldsymbol{\theta}(\tau_k)$ and $\boldsymbol{q}_k \approx \boldsymbol{q}(\tau_k)$, respectively.

In discrete settings, we obtain the following constrained minimization problem:

$$\begin{aligned}
&\underset{\boldsymbol{\theta},\boldsymbol{q},\boldsymbol{\theta}_K}{\text{minimize}}\ \mathcal{L}(\boldsymbol{\theta},\boldsymbol{q},\boldsymbol{\theta}_K) := \frac{1}{p}\sum_{j=1}^{p}\ell(\mathscr{H}(\mathbf{W}_K\boldsymbol{q}_{j,K}+\boldsymbol{b}_K),\boldsymbol{c}_j)+\sum_{k=0}^{K-1}\mathscr{R}(\boldsymbol{\theta}_k,\boldsymbol{\theta}_K),\\
&\text{subject to}\qquad \boldsymbol{q}_{j,k+1} = \boldsymbol{q}_{j,k}+\Delta_t\mathscr{F}(\boldsymbol{q}_{j,k},\boldsymbol{\theta}_k),\\
&\qquad\qquad\quad \mathbf{y}_{j,0} = \mathbf{Q}\boldsymbol{x}_j,
\end{aligned} \tag{6.3}$$

where we have employed explicit Euler scheme to discretize the time derivative $\partial_t\boldsymbol{q}(t)$. This particular choice of the discretization scheme imposes the ResNet architecture. More precisely, we obtain the deep network with $K+1$ layers and identity shortcut connections [HZRS16b]. Figure 6.1 demonstrates the ResNet architecture constructed by discretizing dynamical system (6.1).

Necessary optimality conditions

The necessary optimality conditions of the equality constrained minimization problem (6.3) can be derived from the associated Lagrangian

$$\begin{aligned}
\mathscr{L}(\boldsymbol{\theta},\boldsymbol{q},\boldsymbol{\theta}_K,\hat{\boldsymbol{q}}) := \mathcal{L}(\boldsymbol{\theta},\boldsymbol{q},\boldsymbol{\theta}_K)+\sum_{j=1}^{p}\Bigg(&(\hat{\boldsymbol{q}}_{j,0})^T(\mathbf{Q}\boldsymbol{x}_j-\boldsymbol{q}_{j,0})\\
&+\sum_{k=0}^{K-1}\big((\hat{\boldsymbol{q}}_{j,k+1})^T(\boldsymbol{q}_{j,k}+\Delta_t\mathscr{F}(\boldsymbol{q}_{j,k},\boldsymbol{\theta}_k)-\boldsymbol{q}_{j,k+1})\big)\Bigg),
\end{aligned} \tag{6.4}$$

where $\hat{\boldsymbol{q}}_{j,k}\in\mathbb{R}^\nu$ is Lagrange multiplier, often called adjoint variable, associated with k-th layer and j-th sample. If we set the partial derivatives of (6.4) with respect to all state, adjoint, and control variables to zero, we obtain state, adjoint, and design equations. These equations collectively define the necessary optimality conditions for the optimization problem (6.3).

State equations (forward propagation via ResNet) State equations can be defined recursively as

$$\begin{aligned}
\boldsymbol{q}_{j,k+1} &= \boldsymbol{q}_{j,k}+\Delta_t\mathscr{F}(\boldsymbol{q}_{j,k},\boldsymbol{\theta}_k), \forall k=0,\dots,K-1, && \forall j=1,\dots,p,\\
\mathbf{y}_{j,0} &= \mathbf{Q}\boldsymbol{x}_j, && \forall j=1,\dots,p.
\end{aligned}$$

This corresponds to a forward propagation of input sample $\boldsymbol{x}_j$ through the network consisting of $K+1$ layers. Each k-th layer of the network is associated with $\boldsymbol{\theta}_k\in\mathbb{R}^d$ parameters obtained by discretizing $\boldsymbol{\theta}(t)$ at τ_k. Given sample $\boldsymbol{x}_j$, the state of the k-th layer is denoted by $\boldsymbol{q}_{j,k}\in\mathbb{R}^\nu$.

The function $\mathscr{F}$ can take on various forms, such as simple perceptron, defined by

$$\mathscr{F}(\boldsymbol{q}_k,\boldsymbol{\theta}_k) := \sigma(\mathbf{W}_k\boldsymbol{q}_k+\boldsymbol{b}_k), \tag{6.5}$$

where an affine transformation is performed using a set of parameters $\boldsymbol{\theta}_k := \{\mathbf{W}_k, \boldsymbol{b}_k\}$ consisting of weights $\mathbf{W}_k \in \mathbb{R}^{\nu\times\nu}$ and biases $\boldsymbol{b}_k \in \mathbb{R}^{\nu}$. The operator $\mathbf{W}_k$ can be a dense or a sparse matrix. The latter is common for convolutional neural networks [GBC16]. The function $\sigma : \mathbb{R}^{\nu} \to \mathbb{R}^{\nu}$ denotes the nonlinear activation function. Common examples contain a rectified linear unit (ReLu), defined as $\sigma(z) = \max[0, z]$, or hyperbolic tangent, defined as $\sigma(z) = \tanh(z)$.

Adjoint equations The propagation of partial derivatives with respect to network states through network layers is described by the following adjoint equations:

$$\begin{aligned}
\hat{\boldsymbol{q}}_{j,k} &= \left(\partial_{\boldsymbol{q}_{j,k}}\big(\boldsymbol{q}_{j,k} + \Delta_t \mathscr{F}(\boldsymbol{q}_{j,k}, \boldsymbol{\theta}_k)\big)\right)^T \hat{\boldsymbol{q}}_{j,k+1}, \qquad \forall j = 1,\dots,p, \;\; \forall k = 0,\dots,K-1,\\
\hat{\boldsymbol{q}}_{j,K} &= \frac{1}{p}(\partial_{\boldsymbol{q}_{j,K}} \ell(\mathscr{H}(\mathbf{W}_K \boldsymbol{q}_{j,K} + \boldsymbol{b}_K), \boldsymbol{c}_j))^T, \qquad \forall j = 1,\dots,p,
\end{aligned} \tag{6.6}$$

where $\partial_x = \frac{\partial}{\partial x}$. The propagation of derivatives in (6.6) is performed backward, starting at the final time T with a derivative of the loss function ℓ. This corresponds to the well-known back-propagation method [CR95]. Note, the back-propagation method can be seen as a special case of the adjoint method [CLPS03].

Design equations Given feasible state and adjoint variables, we can finally formulate the design equations as

$$\begin{aligned}
&\sum_{j=1}^{p}\left(\partial_{\boldsymbol{\theta}_k}\big(\boldsymbol{q}_{j,k} + \Delta_t \mathscr{F}(\boldsymbol{q}_{j,k}, \boldsymbol{\theta}_k)\big)\right)^T \hat{\boldsymbol{q}}_{j,k+1} + (\partial_{\boldsymbol{\theta}_k}\mathscr{R}(\boldsymbol{\theta}_k, \boldsymbol{\theta}_K))^T = 0, \;\; \forall k = 0,\dots,K-1,\\
&\frac{1}{p}\sum_{j=1}^{p}(\partial_{\mathbf{W}_K,\boldsymbol{b}_K}\ell(\mathscr{H}(\mathbf{W}_K \boldsymbol{q}_{j,K} + \boldsymbol{b}_K), \boldsymbol{c}_j))^T + (\partial_{\boldsymbol{\theta}_K}\mathscr{R}(\,\cdot\,, \boldsymbol{\theta}_K))^T = 0.
\end{aligned} \tag{6.7}$$

The equations (6.7) describe the sensitivity of the objective function $\mathcal{L}$ with respect to the network parameters (controls) and form a reduced gradient. This reduced gradient is often utilized by the solution strategies in order to find optimal network parameters $\boldsymbol{\theta} := \{\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{K-1}, \boldsymbol{\theta}_K\}$. Indeed, the majority of the optimizers rely on knowledge about the reduced gradient, while the higher-order derivatives are rarely used. Since the dynamic in (6.3) is decoupled across samples, the reduced gradient (6.7) can be evaluated using only a portion of the whole dataset $\mathcal{D}$. This is often required by stochastic/mini-batch solution strategies, such as stochastic gradient descent (SGD) [RM51], or Adam [KB14].

6.2 Recursive multilevel trust-region method

In this work, we propose to minimize the discrete optimization problem (6.3) using the RMTR method with a trust-region radius defined by the ℓ_2-norm. Our implemen-

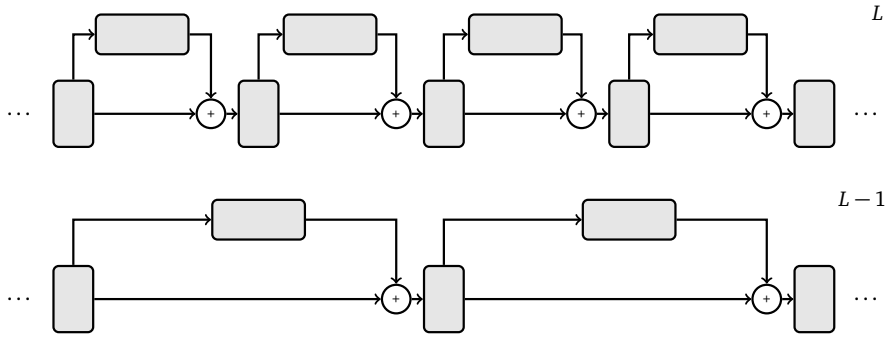

Figure 6.2: An example of a multilevel hierarchy of ResNets. The state and control variables are discretized using different time grids.

tation of the RMTR method for training deep residual networks follows the discussion in Chapter 2. Here, we provide additional details about the construction of the multilevel hierarchy, choice of level-dependent objective functions, and multilevel treatment of trust-region constraints. Furthermore, we propose to integrate the RMTR method into a stochastic framework. The V-cycle of the RMTR method is then performed using only a subset of samples from dataset $\mathcal{D}$. In contrast to standard stochastic optimization methods, our method also incorporates curvature information. In particular, we approximate the Hessian on each level of the multilevel hierarchy using a limited-memory secant method. A detailed description of limited-memory secant methods can be found in Section 1.3.1.

6.2.1 Multilevel hierarchy and transfer operators

The optimization problem (6.2) can be discretized using various discretization parameters, which allows for the construction of a multilevel-hierarchy. We consider a hierarchy of L levels, denoted by $l = 1, \ldots, L$. The finest level, L, represents the discretization of the optimal control problem (6.2) with satisfactory resolution/representation capacity. Thus, the network has sufficiently many layers to ensure desirable approximation properties of the model f_m.

To obtain networks associated with coarser levels, we discretize the time interval $[0, T]$ with larger time-steps. We assume a uniform coarsening in time by a factor of two, i.e., $\Delta_t^{l-1} = 2\Delta_t^l$. As a consequence, the number of layers and therefore parameter is halved between two subsequent levels. Since the cost of forward-backward propagation grows linearly with respect to the number of parameters [HN92], it is roughly two–times cheaper to perform one forward-backward propagation on level l than on level $l+1$. Figure 6.2 illustrates a multilevel hierarchy of ResNets.

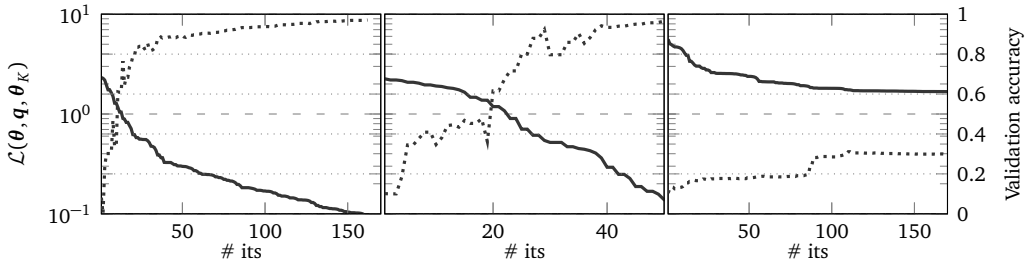

Figure 6.3: The convergence of the (deterministic) trust-region method with respect to the choice of final time T and time-step Δ_t. The solid lines correspond to the value of the objective function, while the dotted lines describe the validation accuracy. The experiment performed using ResNet with 20 residual blocks and Blobs dataset. The final time and time-step are chosen as follows: *Left:* $T = 2,\ \Delta_t = 0.1$. *Middle:* $T = 10,\ \Delta_t = 0.5$. *Right:* $T = 40,\ \Delta_t = 2$.

Numerically stable multilevel training

Deep networks are known to suffer from problems such as vanishing or exploding gradients [GBC16]. These problems are related to the stability of discrete forward propagation [HR17]. Indeed, it is well known that the explicit Euler method is not unconditionally stable. To ensure stability, a sufficiently small time-step Δ_t has to be employed on all levels of the multilevel hierarchy.

We note that the choice of the time-step Δ_t also influences the stability of the training process since the value of Δ_t scales the output of the residual block $\mathscr{F}$, for each layer. Figure 6.3 demonstrates the convergence of the trust-region method with respect to the time-step Δ_t. During this experiment, we kept the number of residual blocks/layers fixed and changed the size of time-step Δ_t. As we can see, small values of Δ_t slow down the convergence process. In contrast, very large values of Δ_t cause severe numerical instabilities and prevent convergence to a solution. As a consequence, it is necessary to carefully select the final time T, such that training on all levels of the multilevel hierarchy is numerically stable.

Transfer operators

We prolongate the network parameters from level l to $l+1$ using a two-step process. In the first step, we simply copy the parameters contained in $\mathbf{Q}^l$ and $\boldsymbol{\theta}_K^l$ to the level $l+1$. Thus, the prolongation operator is the identity, as parameters of the initial condition and the classification are represented by all networks. In the second step, we prolongate the network parameters contained in $\{\boldsymbol{\theta}_k^l\}_{k=1}^{K-1}$. Here, we employ linear interpolation in time (in 1D). The restriction operators are then obtained as adjoint of the prolongation operators, while projection operators are approximated using techniques described in Section 2.3.

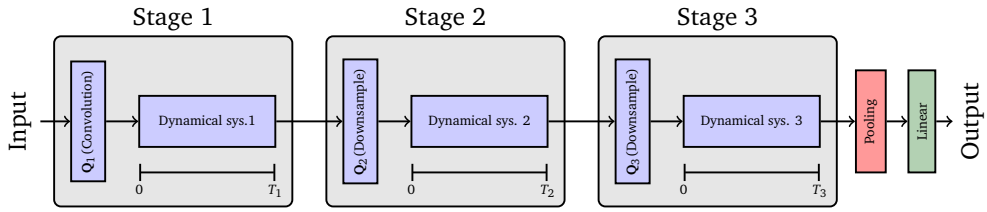

Figure 6.4: An example of three-stage ResNet, designed for image recognition tasks.

Networks with varying width

Until now, we considered only ResNets with constant width. This is due to the fact that dynamical systems, such as (6.1), do not allow for a change of dimensionality. The projection of the dynamics to space of higher/lower dimensions can be performed only at $t = 0$, or $t = T$. However, it is quite common in deep-learning to utilize networks with varying width. Special interest is given to convolutional networks, which split a network into R-stages. Each stage is associated with different image sizes and network width (number of filters). The change in dimensionality between different stages is usually performed by downsampling.

We incorporate R-stage network architectures into our multilevel framework by interpreting the forward propagation as a composition of several dynamical systems. The R-stage network is then obtained by stitching together R dynamical systems as follows:

$$\begin{aligned} \partial \boldsymbol{q}_r(t) &= \mathscr{F}(\boldsymbol{q}_r(t), \boldsymbol{\theta}_r(t)), \qquad \forall t \in [0, T_r], \quad \forall r \in 1, \dots, R, \\ \boldsymbol{q}_r(0) &= \begin{cases} \mathbf{Q}_r \boldsymbol{x}, & \text{if } r = 1, \\ \mathbf{Q}_r \boldsymbol{q}_{r-1}(T_{r-1}), & \text{otherwise.} \end{cases} \end{aligned} \tag{6.8}$$

Thus, each stage is associated with one dynamical system, which transforms input $\boldsymbol{x}$ or output of previous stage $\boldsymbol{q}_{r-1}(T_{r-1})$ into $\boldsymbol{q}_r(T_r)$. The matrices $\{\mathbf{Q}_r\}_{r=1}^R$ in (6.8) incorporate the dimensionality change. Since our goal is to obtain standard ResNet architecture [HZRS16a], we again discretize all time derivatives in (6.8) using the explicit Euler method. Figure 6.4 illustrates the above interpretation on a three-stage ResNet.

Similarly to the previous section, we can obtain a multilevel hierarchy of R-stage ResNets by discretizing the dynamical systems (6.8) with varying discretization parameters. The construction of transfer operators also follows the discussion from above.

Remark 23. *The choice of the time interval* $[0, T_r]$*, and the discretization strategy associated with different dynamical systems can be chosen differently.*

6.2.2 Level-dependent objective functions

Our RMTR method, designed for training ResNets, employs hybrid first-order level-dependent objective functions, as described in Section 2.4.4. Thus, level-dependent objective functions are constructed as a convex combination of additive and multiplicative

coarse-level models. We use (2.33), to determine the parameter, which prescribes the exact form of the convex combination.

6.2.3 Multilevel treatment of constraints

The proposed variant of the RMTR method employs a trust-region defined by the ℓ_2-norm. As a consequence, we can employ techniques described in Section 2.5.3 in order to control size of prolongated coarse-level corrections.

6.2.4 The RMTR in hybrid (deterministic-stochastic) settings

In this work, we use the RMTR method in conjunction with a dynamic sampling strategy. This gives rise to the hybrid deterministic-stochastic optimization method, named dynamic sample sizes RMTR (DSS-RMTR) method. The DSS-RMTR starts the training process in a stochastic regime, which uses only a small subset of all samples, called a mini-batch, in order to evaluate an objective function and its gradient. As training progresses, objective function and gradient are evaluated with increasing accuracy, i.e., by considering a larger subset of all samples. Eventually, the full dataset is used and the DSS-RMTR method operates in the deterministic regime. At this point, the global convergence properties of the DSS-RMTR method follow directly from the theory developed in References [GST08, GK09b].

The proposed DSS-RMTR method increases the mini-batch sizes in an adaptive manner. Similarly to the adaptive sample size trust-region (ASTR) method [MS19], the DSS-RMTR adjusts mini-batch sizes using information about objective function evaluated using full dataset $\mathcal{D}$. The DSS-RMTR method differs from ASTR in two main aspects. Firstly, the search-direction associated with a given mini-batch is obtained using a V-cycle of the RMTR method, not an iteration of the single-level trust-region method. Secondly, the DSS-RMTR method incorporates curvature information by means of limited-memory secant methods. The numerical results presented in Reference [MS19] rely only on first-order information. We remark that using limited-memory secant methods within the stochastic regime is not trivial and requires several adjustments compared to the deterministic regime.

The DSS-RMTR algorithm

The DSS-RMTR method consists of two phases: global and local/mini-batch. The global phase is performed using a full dataset $\mathcal{D}$, while the local phase utilizes subsets of dataset $\mathcal{D}$. Throughout the following, we use the subscript pair (e, b) to denote quantities associated with global and local phases, e.g., $\boldsymbol{\theta}_{e,b}$ denotes parameters obtained during e-th epoch using mini-batch b. Since the dynamic sampling strategy acts only on the finest level, we omit superscripts denoting a given level.

Local phase The local phase starts by generating a set of mini-batches $\{\mathcal{D}_b\}_{b=1}^{n_e}$, where $n_e \geq 1$. Samples of each mini-batch $\mathcal{D}_b$ are extracted from the dataset $\mathcal{D}$, such that each $\mathcal{D}_b$ contains mbs_e samples. Particular details about construction of $\{\mathcal{D}_b\}_{b=1}^{n_e}$ will be provided in Section 6.2.4.

Once the mini-batches $\{\mathcal{D}_b\}_{b=1}^{n_e}$ have been created, we construct a set of local optimization problems. Each local optimization problem has the same form as the global minimization problem (6.3), but the loss/objective function is evaluated using only samples from one mini-batch. We denote the sub-sampled objective functions associated with local optimization problems collectively as $\{\mathcal{L}_b\}_{b=1}^{n_e}$. These local optimization problems are then approximately solved, using one V-cycle of the RMTR method, in a successive manner. Thus, the parameters $\boldsymbol{\theta}_{e,b}$, obtained by minimiznig $\mathcal{L}_b$, are used as an initial guess for the minimization of the function $\mathcal{L}_{b+1}$. A local phase terminates once we have iterated over all mini-batches. We note that the local phase can be interpreted as one step of a nonlinear block Gauss-Seidel method.

Global phase In a global phase, the DSS-RMTR method determines the quality of a trial point $\boldsymbol{\theta}_{e,n_b}$ obtained as a result of the local phase. This is achieved by using global trust-region ratio ρ_e^G, defined as

$$\rho_e^G = \frac{\mathcal{L}(\boldsymbol{\theta}_{e,0}) - \mathcal{L}(\boldsymbol{\theta}_{e,\mathrm{n}_e})}{\frac{1}{\mathrm{n}_e}\sum_{b=1}^{\mathrm{n}_e} \mathcal{L}_b(\boldsymbol{\theta}_{e,b}) - \mathcal{L}_b(\boldsymbol{\theta}_{e,b+1})} = \frac{\text{global reduction}}{\text{average local reduction}}.$$

Thus, the global trust-region ratio ρ_e^G describes the difference between actual reduction observed in the global objective function $\mathcal{L}$ and an average local reduction, obtained while minimizing the local objective functions $\{\mathcal{L}_b\}_{b=1}^{n_e}$.

As custom in trust-region algorithms, the trial point $\boldsymbol{\theta}_{e,n_b}$ is accepted only, if $\rho_e^G > \zeta_1$, where $\zeta_1 > 0$. Otherwise, we reject the trial point. In addition, the global trust-region ratio ρ_e^G is used to adjust the mini-batch size. Since small values of ρ_e^G indicate that $\{\mathcal{L}_i\}_{i=1}^{n_e}$ do not approximate $\mathcal{L}$ well, we increase the mini-batch size. Thus, we decrease the number of mini-batches, but each mini-batch will contain larger portion of samples from $\mathcal{D}$, i.e., $\text{mbs}_{e+1} > \text{mbs}_e$. In contrast, large values of ρ_e^G suggest that the averaged sub-sampled objective functions $\{\mathcal{L}_b\}_{b=1}^{n_e}$ approximate $\mathcal{L}$ well and can be used during the next epoch. The described process is summarized in Algorithm 9.

Remark 24. *Numerical evaluation of the global trust-region ratio ρ_e^G is an expensive operation, especially if the number of samples in the dataset $\mathcal{D}$ is large. We can decrease the computational cost by performing the local phase multiple times before a global phase takes place.*

Generation of mini-batches and multilevel limited-memory quasi-Newton updates

The DSS-RMTR method incorporates curvature information by approximating the Hessian on each level of the multilevel hierarchy. The approximation is performed using

Algorithm 9 DSS-RMTR($\mathcal{L}$, $\boldsymbol{\theta}^L_{0,0}$, $\Delta_{0,0}$, $\text{epoch}_{\max}$, mbs_0)

Require: $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{\theta}_{0,0} \in \mathbb{R}^n$, $\Delta_0 \in \mathbb{R}$, $\text{epoch}_{\max} \in \mathbb{N}$, $\text{mbs}_0 \in \mathbb{N}$
Constants: $o \in \mathbb{R}$, $L \in \mathbb{N}$

```
for e = 0, ..., epoch_max do
    {D_b}_{b=1}^{n_e} = GenMiniBatches(D, mbs_e, o)        ▷ Construct mini-batches with overlap o

    for b = 1, ..., n_e do
        Construct L_b using D_b                            ▷ Generate mini-batch objective function
        [θ_{e,b}, Δ_{e,b}] = RMTR(L, L_b, θ_{e,b-1}, __, Δ_{e,b-1})   ▷ Call RMTR method
        red_b = L_b(θ_{e,b}) − L_b(θ_{e,b+1})              ▷ Compute mini-batch reduction
    end for

    if mbs_e < |D| then                                   ▷ Detect mini-batch (stochastic) regime
        ρ_e^G = (L(θ_{e,0}) − L(θ_{e,n_e})) / ((1/n_e) Σ_{b=1}^{n_e} red_b)   ▷ Compute global (batch) TR ratio

        [θ_{e+1,0}, mbs_{e+1}] = Global_conv_control(ρ_e^G, θ_{e,0}, θ_{e,n_e}, mbs_e)   ▷ Call Algorithm 10

    else                                                  ▷ Detect deterministic regime
        θ_{e+1,0} = θ_{e,n_e}
    end if

    Δ_{e+1,0} = Δ_{e,n_e}                                 ▷ Initialize TR radius for next epoch
end for
return θ_{e+1,0}, Δ_{e+1,0}
```

limited-memory secant methods. Therefore each level l is associated with memory $\{\boldsymbol{s}^l_i, \mathbf{y}^l_i\}_{i=1}^M$ of M secant pairs. Recall Section 1.3.1, each secant pair $\{\boldsymbol{s}^l_i, \mathbf{y}^l_i\}$ consists of search direction $\boldsymbol{s}^l_i$ and the variation of the gradient $\mathbf{y}^l_i$ along this direction. Typically, the pairs $\{\boldsymbol{s}^l_i, \mathbf{y}^l_i\}_{i=1}^M$ are collected over last M iterations. In the context of our DSS-RMTR method, this would mean that the pair $\{\boldsymbol{s}^l_i, \mathbf{y}^l_i\}$ is obtained as

$$\begin{aligned} \boldsymbol{s}^l_i &= \boldsymbol{\theta}^l_{b,i+1} - \boldsymbol{\theta}^l_{b,i}, \\ \mathbf{y}^l_i &= \nabla \mathcal{L}^l_b(\boldsymbol{\theta}^l_{b,i+1}) - \nabla \mathcal{L}^l_b(\boldsymbol{\theta}^l_{b,i}), \end{aligned} \tag{6.9}$$

where $\boldsymbol{s}^l_i$ is a search-direction computed at level l, during i-th iterate, while minimizing a local objective function $\mathcal{L}_b$, associated with a mini-batch $\mathcal{D}_b$. The vector $\mathbf{y}^l_i$ expresses the difference between the gradients of the local objective function $\mathcal{L}^l_b$ evaluated at $\boldsymbol{\theta}^l_{b,i+1}$ and $\boldsymbol{\theta}^l_{b,i+1}$.

Unfortunately, obtaining $\mathbf{y}^l_i$ as described in (6.9) increases the computational cost of our multilevel method. For example, let us assume that the RMTR method is configured with one pre- and one post-smoothing step on a level l. One V-cycle then requires two gradient evaluations per smoothing step, i.e. four gradient evaluations per level. In contrast, if we restrict ourselves to use only the first-order information, then one V-cycle of the RMTR method would necessitate only two gradient evaluations per level.

Algorithm 10 Global_conv_control(ρ_e^G, $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_{e+1}$, mbs_e)

Require: $\rho_e^G \in \mathbb{R}$, $\boldsymbol{\theta}_e \in \mathbb{R}^n$, $\boldsymbol{\theta}_{e+1} \in \mathbb{R}^n$, $\text{mbs}_e \in \mathbb{N}$
Constants: $\zeta_1, \zeta_2, \omega \in \mathbb{R}$, where $\zeta_1 > 0$, $0 < \zeta_2 \leq 0.2$, $\omega > 1.0$

```
if ρ_e^G > ζ_1 then
    θ_* = θ_{e+1}                          ▷ Accept of trial point
else
    θ_* = θ_e                              ▷ Reject of trial point
end if

if ρ_e^G < ζ_2 then
    mbs_* = ω mbs_e                        ▷ Increase mini-batch size
else
    mbs_* = mbs_e                          ▷ Preserve mini-batch size
end if
return θ_*, mbs_*
```

To decrease computational cost of our method, we generate mini-batches $\{\mathcal{D}_b\}_{b=1}^{n_e}$ using the overlapping sampling strategy [BT20, EGMO20]. This sampling strategy was originally developed to ensure the stability of limited-memory quasi-Newton updates in stochastic settings [BT20]. The sampling strategy splits a shuffled dataset $\mathcal{D}$ into n_e mini-batches of size mbs_e. Each mini-batch $\mathcal{D}_b$ is constructed as $\mathcal{D}_b = \{O_{b-1}, S_b, O_b\}$, where S_b denotes samples unique to the mini-batch $\mathcal{D}_b$. Symbols O_{b-1}, O_b denote samples of mini-batch $\mathcal{D}_b$, which are shared with mini-batches $\mathcal{D}_{b-1}$ and $\mathcal{D}_{b+1}$, respectively. The number of overlapping samples contained in O_{b-1}, O_b is usually fairly low. In this work, we prescribe 20% overlap between samples in $\mathcal{D}_b$ and $\mathcal{D}_{b+1}$, for all $b \in \{1, \ldots, n_e - 1\}$ during the first epoch. This determines the size of O_{b-1}, and O_b, which we keep constant during the whole training. Thus, the ratio between an overlapping and a non-overlapping portion of the samples in a mini-batch increases during training. Figure 6.5 illustrates the construction of mini-batches using the overlapping sampling strategy.

Remark 25. *Samples drawn by an overlapping sampling strategy are not independent. An alternative sampling strategy, which draws samples in an independent manner, but with higher computational cost, can be found in Reference [BT20].*

Using the overlapping sampling strategy allows to construct secant pair $\{\boldsymbol{s}_i^l, \mathbf{y}_i^l\}$ on a level l as follows:

$$\begin{aligned} \boldsymbol{s}_i^l &= \boldsymbol{\theta}_{b,i+1}^l - \boldsymbol{\theta}_{b,i}^l, \\ \mathbf{y}_i^l &= \nabla \mathcal{L}_{O_b}^l(\boldsymbol{\theta}_{b,i+1}^l) - \nabla \mathcal{L}_{O_b}^l(\boldsymbol{\theta}_{b,i}^l). \end{aligned} \tag{6.10}$$

The symbol $\nabla \mathcal{L}_{O_b}^l$ denotes a gradient of (6.3), evaluated using only samples contained in O_b. Given that $|O_b| < |\mathcal{D}_b|$, the evaluation of $\mathbf{y}_i^l$ using (6.10) is computationally cheaper than using formula (6.9).

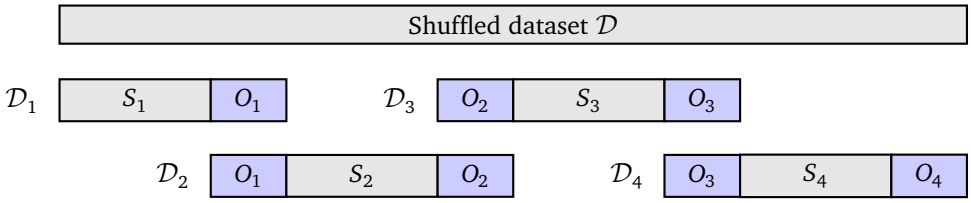

Figure 6.5: Example of four mini-batches created with overlap.

Table 6.1: Choice of parameters used inside TR/RMTR algorithms.

Parameter	η_1	η_2	γ_1	γ_2	ζ_1	ζ_2	ω	μ_1	μ_2	μ^1
Value	0.1	0.75	0.5	2.0	0.1	0.0	2.0	1	0	1

6.3 Implementation

Our multilevel framework for deep residual networks requires an efficient evaluation of the objective function (6.3) and reduced gradient (6.7). To this aim, we implement ResNets using Keras [C+15] and Tensorflow [AAB+15] libraries. Our implementation of solution strategies is based on the python library NumPy [WCV11]. During all experiments, we consider a fixed set of parameters. In particular, trust-region methods are setup using parameters summarized in Table 6.1. The single-level DSS-TR method is obtained by calling the DSS-RMTR algorithm with $L = 1$. Unless specified differently, DSS-TR and DSS-RMTR methods increase the mini-batch size by a factor of two, when requested by the global convergence control strategy, summarized in Algorithm 10. The DSS-RMTR method is configured with one pre-smoothing step, one coarse-level step, and zero post-smoothing steps. To decrease the computational cost of the DSS-RMTR method, we make use of the gradient lagging strategies, see Section 2.6 for a detailed description.

Our implementation of limited-memory secant methods is based on a compact matrix representation of quasi-Newton updates, as discussed in Section 1.3.1. The stability of quasi-Newton updates is ensured by discarding the secant pair whenever necessary, see also Section 1.3.3 for specifics. An initial approximation of reduced Hessian is obtained by solving an eigenvalue problem as proposed in Reference [RM18]. Furthermore, we solve the trust-region subproblems on all levels of multilevel hierarchy using the orthonormal basis (OBS) method [BEM17].

Compared to the first-order stochastic methods, the limited-memory secant methods have a higher computational cost per iteration. However, this additional cost becomes marginal as the size of mini-batches increases [BCN18]. For this reason, we set the memory size to $M = 1$, at the beginning of the training process. The value of M is then increased by one, every time the mini-batch size is enlarged by the DSS strategy.

All presented experiments are performed at the Swiss National Supercomputing Cen-

tre (CSCS) using XC50 compute nodes of the Piz Daint supercomputer. Each XC50 compute node consists of one Intel Xeon E5-2690 v3 processor and an NVIDIA Tesla P100 graphics card. The memory of a node is 64 GB, while the memory of a graphics card is 16 GB.

6.4 Numerical experiments

In this section, we describe numerical examples, that we use in order to assess the convergence properties of the proposed RMTR method. All examples are associated with solving a discretized optimal control problem (6.3). The discretization is performed using varying discretization parameters (time-step), which gives rise to the multilevel hierarchy of networks. Below, we provide a description of network architecture associated with level $l = 1$, i.e., the shallowest network. Deeper networks are then obtained by uniform refinement with a factor of two, as described in Section 6.2.1. All residual networks employ Tikhonov regularization, thus $\mathcal{R}(\cdot) := \beta \|\cdot\|_F^2$, where $\|\cdot\|_F^2$ denotes the Frobenius norm. On the finest level (deepest network), we prescribe the regularization parameter $\beta = 10^{-4}$. On the coarser levels, the value of the regularization parameter β is scaled by a coarsening factor 2^{L-l}, where l denotes a given level.

Our numerical examples can be categorized into two groups, depending on the form of employed residual blocks. More precisely, we consider networks with dense and convolutional residual blocks. We use dense networks to classify particles in two/three-dimensions into distinct classes, given their spatial location. To this aim, we consider three artificially created datasets. Although these datasets do not capture real-life applications, they allow us to investigate the convergence properties of the proposed DSS-RMTR method at a lower computational cost. The second group of our numerical examples considers convolutional networks and image-recognition tasks. Here, we consider three datasets of images, which are often used in the literature for assessing the performance of novel algorithms.

6.4.1 Dense networks

We consider ResNets with residual blocks, which have the form of the simple perceptron, as defined in (6.5). We prescribe operators $\{\mathbf{W}_k\}_{k=0}^K$ to be dense matrices and use the *tanh* activation function. An initial operator $\mathbf{Q}$ is also a dense matrix and its values are learned during the training process. We consider three artificially created datasets, which contain particles located in 2D/3D. Thus, input features describe co-ordinates of the particle, while the output vector prescribes an affiliation to a given class. In particular, we employ the following datasets:

- **Blobs:** Blobs dataset considers particles of a two-dimensional plane grouped into 15 classes. Each class is represented by Gaussian blob [PVG+11] with center point in $[-10, 10]^2$, see Figure 6.6 on the left. The center points are generated randomly

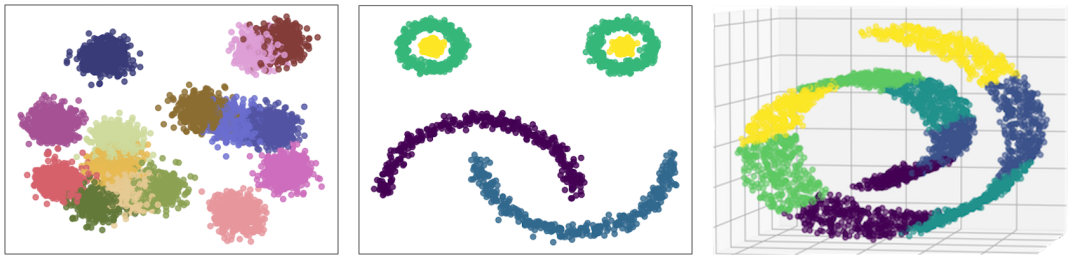

Figure 6.6: *Left:* Blobs dataset consisting of 15 classes (illustrated by different colors). *Middle:* Smiley dataset consisting of 4 classes (illustrated by different colors). *Right:* Spiral dataset consisting of 5 classes (illustrated by different colors).

with a standard deviation of one. The dataset consists of $10,000$ samples, which we split into $7,000$ for training and $3,000$ for testing. We choose the ResNet architecture to have a fixed width of size 3. The discretization in time is performed on interval $[0, T]$, where $T = 1$. We use $K = 7$, thus the coarsest network consists of 7 layers.

- **Smiley:** Smiley dataset contains particles of two-dimensional plane $[-5, 5]^2$ categorized into 4 classes. Each class is related to a particular part of Smiley, see Figure 6.6 in the middle. More precisely, the first and second class consists of samples describing the inner and outer part of Smiley's eyes, respectively. The third and fourth class contains samples describing Smiley's smile, which incorporates two interleaving half circles. This dataset has $7,000$ samples, which are divided into $5,000$ for training and $2,000$ for testing. We use network architecture with a fixed width of 10 and a depth of $K = 7$ layers on the coarsest level. During this experiment, we set the value of T as $T = 1$.
- **Spiral:** The spiral dataset incorporates particles in a three-dimensional hyperplane $[-1.5, 1.5]^3$, classified to 5 classes. All particles are located on spiral geometry, which is generated as described in Reference [Mar15, Chapter 10] and implemented within the Sklearn library [PVG+11]. The position of particles on a spiral then defines 10 unique chunks. Each chunk is assigned randomly to one of 5 classes, such that each class consists of two unique chunks. The spiral dataset contains $7,000$ samples, where $5,000$ are used for training and $2,000$ for testing purposes. During this experiment, we use ResNet with a fixed width of 5. The network depth is defined by $T = 7$ and $K = 7$ on the coarsest level.

6.4.2 Convolutional networks

As a second step, we demonstrate the convergence properties of our DSS-RMTR method on image classification tasks. We consider ResNet architectures with residual blocks, which have a form of the simple perceptron, as defined by (6.5), with *ReLu* activation

Figure 6.7: Example of images contained in Fashion, SVHN, and Cifar-10 datasets (from left to right).

function. The matrices $\{\mathbf{W}_k\}_{k=0}^K$ in (6.5) are now considered to represent sparse convolutional operators.

We consider following datasets of images:

- **Fashion:** Fashion-MNIST dataset contains Zalando's article images of cloth pieces grouped into 10 classes [XRV17], see Figure 6.7 on the left. The dataset consists of 60,000 samples for training and 10,000 samples for testing. Each sample is represented by 28×28 grayscale image. We perform the classification of images using ResNet with constant width. Each residual block utilizes 32 convolutional filters with a kernel size equal to 3. The time discretization employs $K = 7$ and $T = 7$.
- **SVHN:** SVHN dataset is a database of 32×32 color images [NWC+11]. Each image represents a digit from 0 to 9. The database was created from house numbers collected by Google Street View. It contains images, which lack normalization and contain overlapping digits, see Figure 6.7 in the middle. The SVHN dataset consists of 73,257 training samples and 26,032 testing samples. We classify images using three-stage ResNet. At the beginning of each stage, we double the number of filters and halve the size of the feature map. The dimensionality of the feature map is then preserved for all layers within a given stage. We employ the same time discretization parameters for all stages. Thus, the depth of the coarse-level network is defined by $T_i = 3$ and $K_i = 7$, where $i = \{1, 2, 3\}$. A number of filters is chosen to be 16, 32, and 64. Operators $\{\mathbf{Q}_i\}_{i=1}^3$, which change dimensionality between different stages, represent average pooling operation [GBC16].
- **Cifar-10:** Cifar-10 dataset consists of 60,000 color images, where we use 50,000 for training and 10,000 for testing [KH+09] . Each 32×32 image belongs to one of 10 classes, see also Figure 6.7. We employ the same ResNet architecture as for the SVHN dataset.

All datasets are pre-processed by standardizing the images, so pixel values lie in the range $[0, 1]$. Furthermore, we carry out centering by subtracting the mean from each pixel.

6.5 Numerical results

In this section, we study the performance of the RMTR method. Firstly, we focus on deterministic (batch) settings and validate the proposed multilevel framework. Here, we also investigate the performance of our RMTR with respect to the choice of a coarse level model and cycling scheme. Secondly, we consider hybrid stochastic-deterministic settings, and investigate convergence properties of the DSS-RMTR method, as presented by Algorithm 9.

To assess the performance of the (DSS-)RMTR method, we provide a comparison with the single-level (DSS-)TR method. Since the computational cost of one V-cycle of the RMTR method is higher than the computational cost of one TR iteration, we need to devise a suitable metric to perform a fair comparison. We focus on the most expensive part of the training, i.e., the cost associated with an evaluation of the gradients. To this aim, we define one work unit W^L to represent a computational cost associated with an evaluation of the gradient on the finest level, using a full dataset $\mathcal{D}$. Given that the computational cost of the back-propagation algorithm scales linearly with a number of samples and a number of the layers, we can define the total computational cost W as follows:

$$W = \sum_{e=1}^{e_{\text{tot}}} \sum_{b=1}^{n_e} \sum_{l=1}^{L} \frac{n_b}{p} 2^{l-L} Q_b^l W^L, \tag{6.11}$$

where e_{tot} denotes a number of epochs required by the RMTR method to converge. Symbol Q_b^l denotes the number of gradient calls performed on level l, using mini-batch $\mathcal{D}_b$. Given an epoch e, the computational cost is obtained by summing up gradient evaluations performed on all levels using all mini-batches. Since the computational cost of a gradient evaluation on level $l < L$ using mini-batch $\mathcal{D}_b$ is lower than one work unit W^L, we need to rescale quantities in (6.11) accordingly. In particular, the scaling factor 2^{l-L} accounts for a difference between the computational cost on a level l and the finest level L. Please note, this scaling factor assumes uniform coarsening in 1D by a factor of two. The scaling factor $\frac{n_b}{p}$ takes into consideration the difference between a number of samples contained in the dataset $\mathcal{D}$ and the mini-batch $\mathcal{D}_b$.

Remark 26. *If we set $L = 1$, then (6.11) determines the total computational cost of the single-level (DSS-)TR method.*

6.5.1 Deterministic (batch) settings

In this section, we investigate the convergence properties of the TR and the RMTR method within deterministic settings. Here, we consider only the dense ResNets and artificially created datasets, described in Section 6.4. We train all networks, with following stopping criteria:

$$\text{acc}_{\text{train}} > 0.98 \quad \text{or} \quad \text{acc}_{\text{val}} > 0.98$$

Table 6.2: The average total computational cost required by the RMTR method, configured with different coarse-level models and cycling schemes. The RMTR method is configured with 4 levels and the L-SR1 Hessian approximation strategy. The results are obtained by averaging over 10 independent runs.

Example	Cycle	Multiplicative	Additive	Hybrid
Blobs	V	$23.2 \pm 9.3\%$	$33.7 \pm 11.2\%$	$21.8 \pm 7.2\%$
	F	$17.1 \pm 5.4\%$	$15.3 \pm 5.4\%$	$14.8 \pm 5.5\%$
Smiley	V	$35.2 \pm 10.2\%$	$39.9 \pm 13.2\%$	$33.5 \pm 8.4\%$
	F	$31.2 \pm 6.1\%$	$28.5 \pm 6.3\%$	$28.4 \pm 6.1\%$
Spiral	V	$58.1 \pm 9.9\%$	$82.6 \pm 10.3\%$	$50.2 \pm 8.3\%$
	F	$42.1 \pm 5.3\%$	$35.2 \pm 6.5\%$	$29.9 \pm 5.3\%$

is satisfied. The symbols $\text{acc}_{\text{train}}$ and acc_{val} denote train and validation accuracy, respectively.

Convergence properties with respect to the coarse-level model and cycling scheme

As a first step, we study how the choice of cycling scheme and coarse-level model influences the performance of the RMTR method. More precisely, we consider V- and F-cycles and three types of the first-order consistent coarse-level models, namely additive, multiplicative, and hybrid, as defined in Section 2.4. Table 6.2 summarizes obtained results. The results are provided in terms of average total computational cost and standard deviation (std) obtained over 10 independent runs. As we can see, the F-cycle is computationally less expensive than V-cycle. Besides, using F-cycle helps to reduce the variability of the obtained results. In particular, the std is approximately two times lower, when the F-cycle is employed.

We also observe that the additive coarse-level model performs better than the multiplicative one if the F-cycle is used. In contrast, the multiplicative coarse-level model performs better, if the V-cycle is used. This is not surprising, as the additive model is known to represent fine-level objective function more accurately when close to a solution. The multiplicative coarse-level model might introduce new minima, which can help to speed up an escape to more desirable regions of the energy landscape. This can be especially beneficial if the initial guess is chosen far away from a solution. We point out that the hybrid coarse-level model is computationally the most efficient, independently of the choice of a cycling scheme.

For the remainder of this work, we use the RMTR method in the form of F-cycle and employ hybrid first-order consistent level-dependent objective functions.

Table 6.3: The average total computational cost required by the deterministic TR and the RMTR method using Blobs, Smiley, and Spiral datasets. The results are obtained by averaging 10 independent runs. Symbol —— indicates that no convergence was reached within 1,000 work units.

Example	Method		Levels (Residual blocks) 3 (25)	4 (49)	5 (97)	6 (193)
Blobs	L-BFGS	TR	$68.4 \pm 32\%$	$68.7 \pm 27\%$	$72.5 \pm 31\%$	$85.6 \pm 32\%$
		RMTR	$24.4 \pm 7\%$	$14.6 \pm 2\%$	$8.7 \pm 1\%$	$6.0 \pm 1\%$
	L-SR1	TR	$74.0 \pm 22\%$	$83.3 \pm 33\%$	$83.4 \pm 24\%$	$82.9 \pm 24\%$
		RMTR	$24.2 \pm 5\%$	$14.8 \pm 6\%$	$8.9 \pm 2\%$	$6.4 \pm 1\%$
Smiley	L-BFGS	TR	$182.4 \pm 35\%$	$531.0 \pm 37\%$	$676.2 \pm 36\%$	——
		RMTR	$54.2 \pm 11\%$	$32.4 \pm 8\%$	$18.6 \pm 4\%$	$10.4 \pm 2\%$
	L-SR1	TR	$383.9 \pm 43\%$	$618.4 \pm 44\%$	$828.4 \pm 48\%$	——
		RMTR	$61.4 \pm 16\%$	$28.4 \pm 6\%$	$18.7 \pm 5\%$	$11.1 \pm 2\%$
Spiral	L-BFGS	TR	$149.8 \pm 42\%$	$217.7 \pm 42\%$	$302.7 \pm 43\%$	$392.7 \pm 41\%$
		RMTR	$52.7 \pm 7\%$	$31.2 \pm 8\%$	$23.6 \pm 7\%$	$15.7 \pm 3\%$
	L-SR1	TR	$157.8 \pm 33\%$	$231.3 \pm 35\%$	$332.2 \pm 36\%$	$412.3 \pm 36\%$
		RMTR	$56.5 \pm 8\%$	$29.9 \pm 5\%$	$21.4 \pm 5\%$	$15.6 \pm 3\%$

Comparison of the RMTR method with the TR method

In this section, we compare the performance of the RMTR method with its single-level counterpart. The presented study involves both, the L-BFGS and the L-SR1, Hessian approximation schemes. Table 6.3 demonstrates the obtained results with respect to a different number of levels. As we can see, the total computational cost required by the TR method grows rapidly with network depth. This is as expected since it is known that the deeper networks are more difficult to train than shallow networks [HR17]. Figure 6.8 on the left depicts typical convergence behavior of the TR method, used for the training of ResNets. We observe, that the method encounters a certain plateau region, where only a small decrease in the value of energy is obtained.

In contrast to the TR method, the computational cost required by the RMTR method decreases with the number of layers. This is due to the fact that the initialization of the network parameters, provided by the F-cycle, produces an initial guess, which is relatively close to a solution. The plateau regions are typically encountered on the coarser levels, where the computational cost is low. Typical convergence behavior of the RMTR method is illustrated in Figure 6.8 on the right.

We also remark that the TR method is significantly more sensitive to the choice of an initial guess than the RMTR method. The relative std of the obtained results varies from 30% to 40% for the TR method. In contrast, the relative std for the RMTR method decreases with the number of levels and it is below 3.5% for networks with 6 levels for all datasets.

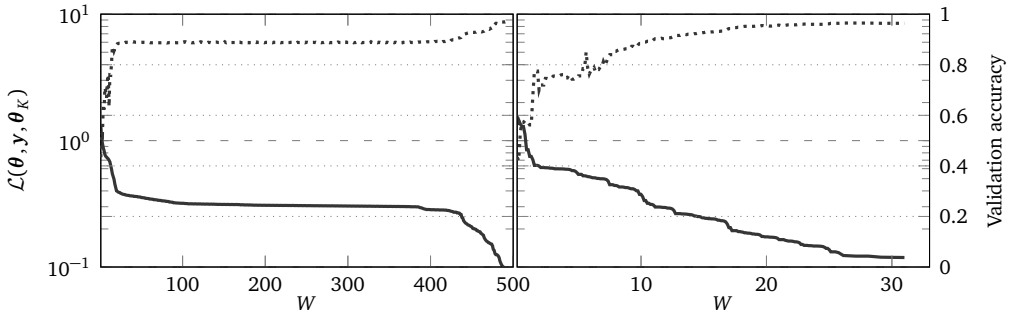

Figure 6.8: Typical convergence behavior of the TR and the RMTR method when used for training of dense ResNets. The example considers a network with 25 residual blocks, Smiley dataset, and L-BFGS Hessian approximation scheme. *Left:* The TR method. *Right:* The four-level RMTR method.

6.5.2 Hybrid (stochastic-deterministic) settings

In this section, we compare the performance of the DSS-TR method and the DSS-RMTR method. Comparison is divided into two parts, related to dense and convolutional ResNets. As in the deterministic settings, we also employ the DSS-RMTR method in the form of an F-cycle. We reset the mini-batch size mbs_0 to its initial value, every time a new level is taken into consideration.

Dense networks

We compare the performance of the DSS-TR method and the DSS-RMTR method using dense ResNets with Blobs, Smiley, and Spiral datasets. The performed study considers three different initial mini-batch sizes. More precisely, parameter mbs_0 takes on a value from $\{250, 500, 1000\}$. Tables 6.4 and 6.5 report obtained results. As we can see, hybrid trust-region methods perform better than their deterministic counterparts. We also note the hybrid methods require lower computational cost when mini-batch size is initialized to a smaller value. This is due to the fact that small-batch methods tend to be more exploratory, which allows them to escape plateau regions. We also highlight the fact that the hybrid methods are less sensitive to the choice of the initial guess than the deterministic methods.

The obtained results imply that the DSS-RMTR method performs significantly better than the DSS-TR method, in terms of total computational cost and the sensitivity to the initial guess. Similarly to the results obtained for the deterministic methods, the total computational cost required by the DSS-TR method increases with network depth. In contrast, the total computational cost required by the DSS-RMTR method decreases with network depth and the number of levels. For example, for the Spiral dataset with 6 levels and 193 residual blocks, the DSS-RMTR method requires 10 times lower computational

Table 6.4: The average total computational cost of the DSS-TR and DSS-RMTR methods required for training dense ResNets. The methods employ the L-BFGS scheme in order to approximate Hessian. The results are obtained by averaging 10 independent runs.

Example	mbs_0	Method	Levels (Residual blocks) 2 (13)	4 (49)	6 (193)
Blobs	250	DSS-TR	$6.8 \pm 4.1\%$	$10.2 \pm 6.1\%$	$31.3 \pm 15.0\%$
		DSS-RMTR	$4.7 \pm 1.7\%$	$4.3 \pm 1.5\%$	$3.9 \pm 1.2\%$
	500	DSS-TR	$11.2 \pm 3.5\%$	$19.1 \pm 6.3\%$	$14.6 \pm 3.5\%$
		DSS-RMTR	$5.6 \pm 1.3\%$	$4.3 \pm 1.6\%$	$4.1 \pm 2.1\%$
	1,000	DSS-TR	$13.1 \pm 6.5\%$	$21.2 \pm 22.1\%$	$78 \pm 24.3\%$
		DSS-RMTR	$9.8 \pm 2.5\%$	$7.3 \pm 2.4\%$	$4.2 \pm 1.8\%$
Smiley	250	DSS-TR	$43.7 \pm 25.1\%$	$53.9 \pm 30.3\%$	$61.2 \pm 40.3\%$
		DSS-RMTR	$19.2 \pm 1.5\%$	$11.2 \pm 3.8\%$	$7.3 \pm 2.6\%$
	500	DSS-TR	$49.6 \pm 25.2\%$	$57.2 \pm 22.2\%$	$70.1 \pm 39.4\%$
		DSS-RMTR	$13.5 \pm 1.9\%$	$11.0 \pm 5.9\%$	$9.1 \pm 1.7\%$
	1,000	DSS-TR	$53.8 \pm 24.1\%$	$63.8 \pm 25.8\%$	$66.6 \pm 32.0\%$
		DSS-RMTR	$25.5 \pm 1.8\%$	$14.3 \pm 4.5\%$	$11.9 \pm 3.1\%$
Spiral	250	DSS-TR	$37.0 \pm 16.4\%$	$54.3 \pm 17.3\%$	$76.8 \pm 22.3\%$
		DSS-RMTR	$30.2 \pm 4.8\%$	$10.2 \pm 2.8\%$	$5.9 \pm 0.5\%$
	500	DSS-TR	$48.3 \pm 27.2\%$	$83.0 \pm 33.6\%$	$119.8 \pm 30\%$
		DSS-RMTR	$36.1 \pm 13.8\%$	$12.1 \pm 3.18\%$	$8.2 \pm 0.8\%$
	1,000	DSS-TR	$61.2 \pm 32.2\%$	$88.2 \pm 35.1\%$	$135.5 \pm 31.1\%$
		DSS-RMTR	$29.1 \pm 6.0\%$	$16.1 \pm 1.92\%$	$13.6 \pm 0.2\%$

cost than the DSS-TR method, if the L-BFGS Hessian approximation scheme is used.

Our numerical results also indicate that the trust-region methods configured with the L-SR1 Hessian approximation scheme require lower computational cost than the methods, which employ the L-BFGS scheme. This can be attributed to the fact that the L-SR1 scheme does not enforce the Hessian approximations to be positive definite and, therefore, allows for a more accurate approximation.

Convolutional networks

Our last set of experiments considers image classification and convolutional neural networks. During all tests, we prescribe an initial mini-batch size mbs_0 to be 128, i.e., we use $\text{mbs}_0 = 128$. Since convolutional ResNets are more challenging to train than dense ResNets, we employ following stopping criterium:

$$\begin{aligned} &\text{acc}_{\text{train}} > 0.98 \quad \text{or} \quad \left(\sum_{i=1}^{10} (\text{acc}_{\text{train}})_e - (\text{acc}_{\text{train}})_{e-i} \right) < 0.001 \quad \text{or} \\ &\text{acc}_{\text{val}} > 0.98 \quad \text{or} \quad \left(\sum_{i=1}^{10} (\text{acc}_{\text{val}})_e - (\text{acc}_{\text{val}})_{e-i} \right) < 0.001. \end{aligned} \tag{6.12}$$

Table 6.5: The average total computational cost of the DSS-TR and DSS-RMTR methods required for training dense ResNets. The methods employ the L-SR1 scheme in order to approximate Hessian. The results are obtained by averaging 10 independent runs.

Example	mbs_0	Method	Levels (Residual blocks) 2 (13)	4 (49)	6 (193)
Blobs	250	DSS-TR	$11.2 \pm 3.4\%$	$19.4 \pm 6.2\%$	$24.6 \pm 3.5\%$
		DSS-RMTR	$4.8 \pm 1.2\%$	$3.2 \pm 1.0\%$	$3.8 \pm 0.7\%$
	500	DSS-TR	$12.4 \pm 4.3\%$	$26.0 \pm 20.1\%$	$27.6 \pm 20.2\%$
		DSS-RMTR	$6.3 \pm 1.6\%$	$4.7 \pm 1.0\%$	$4.1 \pm 0.9\%$
	1,000	DSS-TR	$16.5 \pm 22.8\%$	$29.2 \pm 21.1\%$	$39.3 \pm 24\%$
		DSS-RMTR	$7.9 \pm 1.5\%$	$5.4 \pm 1.2\%$	$4.3 \pm 0.9\%$
Smiley	250	DSS-TR	$20.1 \pm 4.8\%$	$21.5 \pm 4.7\%$	$23.1 \pm 5.0\%$
		DSS-RMTR	$11.7 \pm 2.9\%$	$5.4 \pm 1.7\%$	$4.5 \pm 1.3\%$
	500	DSS-TR	$25.2 \pm 6.5\%$	$25.8 \pm 5.2\%$	$26.2 \pm 5.3\%$
		DSS-RMTR	$16.5 \pm 3.76\%$	$6.4 \pm 1.7\%$	$5.0 \pm 0.8\%$
	1,000	DSS-TR	$31.6 \pm 8.2\%$	$33.4 \pm 7.2\%$	$36.2 \pm 9.1\%$
		DSS-RMTR	$18.2 \pm 4.3\%$	$7.7 \pm 1.4\%$	$6.1 \pm 1.2\%$
Spiral	250	DSS-TR	$17.2 \pm 4.5\%$	$21.4 \pm 5.3\%$	$23.5 \pm 5.5\%$
		DSS-RMTR	$13.8 \pm 4.0\%$	$7.2 \pm 4.2\%$	$4.4 \pm 1.5\%$
	500	DSS-TR	$31.8 \pm 9.4\%$	$32.4 \pm 6.2\%$	$39.3 \pm 4.3\%$
		DSS-RMTR	$25.1 \pm 8.2\%$	$16.3 \pm 3.5\%$	$13.5 \pm 3.23\%$
	1,000	DSS-TR	$34.2 \pm 9.5\%$	$43.9 \pm 13.5\%$	$55.4 \pm 23.2\%$
		DSS-RMTR	$23.2 \pm 6.1\%$	$16.0 \pm 3.5\%$	$13.7 \pm 2.8\%$

The stopping criterium (6.12) verifies whether a train, or validation accuracy of 98% is achieved. Besides, it incorporates early stopping, which halts the training process when there is no improvement in train or validation accuracy within the last 10 epochs. This stopping criterium is commonly used by machine learning practitioners [GBC16].

In this section, we investigate the convergence properties of the solution strategies by measuring the total computational cost and achieved train and validation accuracy. Table 6.6 summarizes the obtained results. All results are obtained by training the network 10 times and then selecting the result with the highest validation accuracy, i.e., which generalizes the best. We report only results obtained using the L-SR1 Hessian approximation scheme, as its performance is superior to the L-BFGS scheme.

The results indicate that the DSS-RMTR method outperforms the DSS-TR method in all scenarios, while it maintains approximately the same validation accuracy. The speed up obtained by the DSS-RMTR method varies for different network architectures and datasets. The lowest speed up, approximately by a factor of 2, is obtained for the SVNH dataset with two levels. The highest speed up, approximately by a factor of 6.5, is obtained for the Fashion dataset with five levels.

We note that the obtained results could be improved in terms of the validation accuracy by incorporating normalization techniques, such as dropout [GG16] or batch-normalization [IS15], into the design of the residual blocks. We plan to integrate these

Table 6.6: Total computational cost of the DSS-TR and DSS-RMTR methods required for training convolutional ResNets. The methods employ the L-SR1 scheme in order to approximate Hessian. The result with the highest validation accuracy is selected from 10 independent runs.

Example	Levels (Residual blocks)	DSS-TR			DSS-RMTR		
		W	acc_{train}	acc_{test}	W	acc_{train}	acc_{test}
Fashion	2 (13)	253	94.7%	92.0%	88.2	93.9%	92.2%
	3 (25)	260	95.0%	92.8%	60.4	91.3%	92.7%
	4 (49)	262	94.9%	92.9%	46.8	93.9%	92.7%
	5 (97)	263	94.8%	93.1%	39.2	91.4%	92.9%
SVNH	2 (15)	133	99.1%	93.1%	68.0	92.7%	93.3%
	3 (27)	123	99.2%	93.3%	58.0	94.7%	93.6%
	4 (51)	122	99.0%	93.8%	51.8	93.9%	93.7%
	5 (99)	130	95.4%	93.9%	45.2	94.7%	94.3%
Cifar-10	2 (15)	102	98.8%	83.6%	46.5	99.1%	83.8%
	3 (27)	107	99.1%	84.5%	40.4	99.1%	84.3%
	4 (51)	113	98.4%	84.7%	35.8	99.3%	85.1%
	5 (99)	139	94.5%	85.5%	31.8	97.4%	85.3%

techniques into our multilevel framework as part of future work.

6.6 Conclusion and outlook

In this chapter, we proposed a variant of the RMTR method for training deep residual networks. Our multilevel framework utilizes a hierarchy of auxiliary networks with different depths to speed up the training process of the original network. The proposed RMTR method operates in a hybrid (stochastic-deterministic) regime and dynamically adjusts mini-batch sizes during the training process. Furthermore, we incorporated curvature information on each level of the multilevel hierarchy using limited-memory secant methods. Our numerical examples demonstrated the robustness of the proposed RMTR variant. A comparison with a single-level TR method was performed and illustrated a significant reduction in terms of total computational cost. Moreover, we showed that the RMTR method is considerably less sensitive to the choice of an initial guess than the TR method.

Outlook

The work presented in this chapter can be extended in several ways. For instance, it would be beneficial to incorporate normalization strategies, such as batch-normalization, into the design of the residual blocks. However, normalization strategies tend to break the finite-sum structure of the loss function. Therefore, trust-region methods, which rely on the monotonic decrease of the objective function can not be readily applied.

We constructed a multilevel hierarchy by coarsening in time. In the case of convolutional neural networks, we could also explore a coarsening in space (image resolution) in order to additionally lower the computational cost of the coarse-level models.

Conclusion and outlook

In this thesis, we have discussed the algorithmic and software developments of the RMTR methods.

In Part I, we provided a description of trust-region based minimization techniques. Particular focus was put on the recursive multilevel trust-region (RMTR) method. We described a generic nonlinear multilevel framework and discussed how to build multilevel hierarchies and transfer operators. We also provided an overview of the additive, multiplicative, and hybrid approaches, which can be used to construct coarse-level models. Furthermore, we discussed how to treat trust-region and variable bounds in multilevel settings. In addition, we proposed an active-set variant of the RMTR method. The resulting method was designed for constrained optimization problems arising from the finite-element discretization of PDEs and utilized truncated basis functions in order to construct suitable coarse-level models. Using numerical examples, we demonstrated the efficiency of the proposed active-set RMTR method.

Later, we presented an open-source C++ library UTOPIA, which incorporates a parallel implementation of the trust-region methods described in this work. Single-level and multi-level trust-region methods were developed such that they allow for a representation of a trust-region radius by the ℓ_2- and the ℓ_∞-norm. In addition, we provided an implementation of several trust-region subproblem solvers as well as limited-memory secant methods. Our implementation targets CPU computing environments and allows for matrix-based and matrix-free computations. We investigated the strong and weak scaling properties of our RMTR implementation with up to $12,000$ processors and a billion degrees of freedom.

In Part II, we proposed three novel variants of the RMTR method. Our first variant was designed for phase-field fracture simulations. In particular, we focused on a monolithic solution scheme and designed RMTR for coupled constrained non-convex optimization problems. The proposed RMTR method employed novel coarse-level models, which combine coarse-level discretizations with the representation of fractures from the finest level. We demonstrated the performance and robustness of our RMTR method using several numerical examples, including pressurized fracture networks with $1,000$ fractures in 2D and 100 fractures in 3D. The obtained results demonstrate that the novel RMTR method provides significant improvements in terms of computational time compared to the staggered solution scheme and the single-level trust-region method.

Our second variant of the RMTR method was designed for the thin-shell cloth simulations. The developed RMTR method utilized subdivision surfaces in order to construct prolongation and restriction operators. We also leveraged a reverse subdivision operator in order to project iterates between subsequent levels of the multilevel hierarchy. Our numerical examples demonstrated the robustness of the proposed RMTR method.In addition, a comparison with a single-level trust-region method was made, where we demonstrated a reduction in the number of the required iterations by several orders of magnitude.

Our third variant of the RMTR method was developed for training deep residual networks (ResNets). The developed method builds a multilevel hierarchy and transfer operators by leveraging the dynamical system's view-point, which casts ResNet as the discretization of an initial value problem. The proposed RMTR method operated in a hybrid (stochastic-deterministic) regime, which dynamically adjusted the mini-batch sizes during training. In addition, we incorporated curvature information by means of the limited-memory secant method, on each level of the multilevel hierarchy. We demonstrated the performance of our RMTR method using various ResNets architectures, with dense and convolutional residual blocks. A comparison with the single-level trust-region method was made and indicated a significant reduction in terms of total computational cost. Furthermore, it is evident from the obtained results, that the RMTR method is significantly less sensitive to the choice of initial guess than the single-level trust-region method.

Outlook

The work presented in this thesis could be extended in several ways, for example:

- The scaling properties of the RMTR could be further enhanced by reducing the communication cost associated with the evaluation of objective function and its derivatives on the coarser levels.
- An implementation of the RMTR method within our UTOPIA library could be ported to GPU-based architectures. Although our implementation of the RMTR methods can already be used using GPU architectures, the memory requirements of our nonlinear multilevel framework should to be reduced in order to enable large-scale GPU simulations. Furthermore, several GPU specific performance optimizations have to be performed to achieve desirable performance.
- The smoothing properties of the successive coordinate minimization (SCM) are superior to the smoothing properties of Krylov-subspace methods. However, the performance of the SCM method deteriorates in parallel. Therefore, it would be beneficial to develop scalable solution strategies for solving trust-region subproblems. An interesting possibility could be to utilize polynomial smoothers, such as the Chebyshev method.

- From an algorithmic point of view, the RMTR method could be further extended to handle linear equality and inequality constraints. This would enhance the applicability of the method to a larger class of nonlinear optimization problems.

Appendix A

Algorithmic details

A.1 Projection of point-wise constraints onto the feasible set

Let us consider the feasible set $\mathcal{F} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{l} \leq \boldsymbol{x} \leq \mathbf{u}\}$, where the inequalities are meant component-wise. The projection $\mathcal{P}(\mathbf{y})$ of vector $\mathbf{y}$ onto $\mathcal{F}$ is also defined component-wise as

$$(\mathcal{P}(\mathbf{y}))_k := \begin{cases} (\mathbf{l})_k, & \text{if} \quad (\mathbf{y})_k \leq (\mathbf{l})_k, \\ (\mathbf{y})_k, & \text{if} \quad (\mathbf{l})_k \leq (\mathbf{y})_k \leq (\mathbf{u})_k, \\ (\mathbf{u})_k, & \text{if} \quad (\mathbf{u})_k \leq (\mathbf{y})_k, \end{cases}$$

for all $k \in \{1, \ldots, n\}$.

A.2 Test problems used for testing active-set RMTR method

This section describes test problems used to generate numerical examples in Part I of this thesis. Through this section, we denote the computational domain by $\Omega \subset \mathbb{R}^d$, where $d = \{1, 2, 3\}$. Furthermore, we use symbol $H^1(\Omega)$ to denote the standard Sobolev space of weakly differentiable functions in L^2 with one weak derivative also in L^2.

A.2.1 Membrane

Let Ω be a unit square, e.g., $\Omega = [0,1]^2$ with boundary Γ, decomposed into three parts: $\Gamma_l = \{0\} \times [0,1]$, $\Gamma_r = \{1\} \times [0,1]$ and $\Gamma_f = [0,1] \times \{0,1\}$. Following Reference [DD07], we consider the minimization problem defined as

$$\begin{aligned} &\underset{u \in \mathcal{X}}{\text{minimize}} \;\; f(u) := \frac{1}{2} \int_\Omega \|\nabla u(\boldsymbol{x})\|^2 \, d\boldsymbol{x} + \int_\Omega u(\boldsymbol{x}) \, d\boldsymbol{x}, \\ &\text{subject to} \;\; \text{lb}(\boldsymbol{x}) \leq u, \quad \text{a.e. in } \Omega. \end{aligned} \tag{A.1}$$

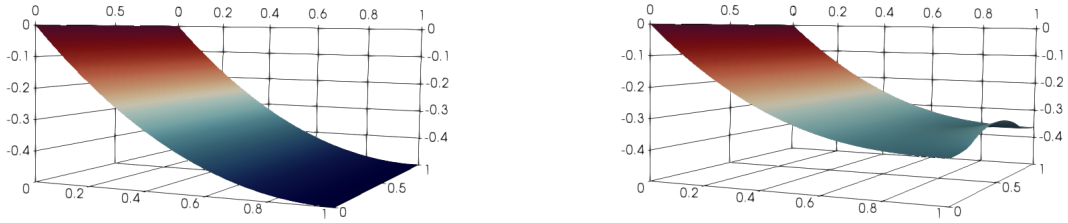

Figure A.1: Membrane test problem. *Left:* unconstrained solution. *Right*: constrained solution.

The function lb defining a lower bound is defined on the right part of the boundary, Γ_r, by the upper part of the circle with the radius, $r = 1$, and the centre, $C = (1; -0.5; -1.3)$. The minimization (A.1) is performed over space $\mathcal{X} := \{u \in H^1(\Omega) \mid u = 0 \text{ on } \Gamma_l\}$. We discretize the problem (A.1) using the finite element method with $\mathbb{Q}_1$ Lagrange elements. The simulation result is depicted in Figure A.1.

A.2.2 Combustion

Let Ω be a unit square, e.g., $\Omega = [0,1]^2$ with boundary Γ. Following [BM+00, KM16], we consider following minimization problem:

$$\begin{aligned} &\underset{u \in \mathcal{X}}{\text{minimize}} \;\; f(u) := \frac{1}{2}\int_\Omega \|\nabla u(\boldsymbol{x})\|^2 - (ue^u - e^u)\, d\boldsymbol{x} - \int_\Omega f(x) u \, d\boldsymbol{x}, \\ &\text{subject to} \;\; \text{lb}(\boldsymbol{x}) \le u \le \text{ub}(\boldsymbol{x}), \quad \text{a.e. in } \Omega. \end{aligned} \tag{A.2}$$

The variable bounds and right hand side in (A.2) are defined with respect to spatial coordinates $\boldsymbol{x}$ as

$$\begin{aligned} \text{lb}(\boldsymbol{x}) &= -8(x_1 - 7/16)^2 - 8(x_2 - 7/16)^2 + 0.2, \\ \text{ub}(\boldsymbol{x}) &= 0.5, \\ f(\boldsymbol{x}) &= (9\pi^2 + e^{(x_1^2 - x_1^3)\sin(3\pi x_2)}(x_1^2 - x_1^3) + 6x_1 - 2)\sin(3\pi x_1), \end{aligned}$$

where $f \in L^2(\Omega)$ and x_1, x_2 are coordinates of $\boldsymbol{x}$. The minimization (A.2) is performed over space $\mathcal{X} := \{u \in H^1(\Omega) \mid u = 0 \text{ on } \Gamma_l\}$. We discretize the problem (A.2) using the finite element method with $\mathbb{Q}_1$ Lagrange elements. The simulation result is depicted in Figure A.2.

A.3 Phase-field fracture model

A.3.1 Additive decomposition of elastic energy

Following Reference [MWH10], we additively decompose the elastic energy density $\psi_e(\boldsymbol{\varepsilon})$ into $\psi_e^+(\boldsymbol{\varepsilon})$ due to the tension and $\psi_e^-(\boldsymbol{\varepsilon})$ due to the compression by using

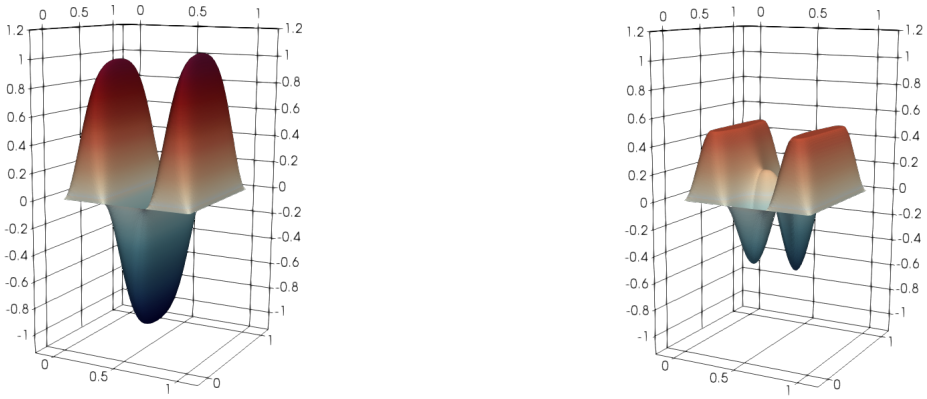

Figure A.2: Combustion test problem. *Left:* Unconstrained solution. *Right:* Constrained solution.

following relation

$$\psi_e^{\pm}(\boldsymbol{\varepsilon}) := \frac{1}{2}\lambda\Big(\big\langle \mathrm{tr}(\boldsymbol{\varepsilon})\big\rangle^{\pm}\Big)^2 + \mu\, \boldsymbol{\varepsilon}^{\pm} : \boldsymbol{\varepsilon}^{\pm}\,. \tag{A.3}$$

The symbols μ and λ in (A.3) denote the Lamé constants. The bracket symbol is defined as $\big\langle x\big\rangle^{\pm} := \frac{(|x|\pm x)}{2}$. The strain tensor related to the desirable mode (tension, or compression) is obtained by the spectral decomposition of the principal strains

$$\boldsymbol{\varepsilon}^{\pm} := \sum_{i=1}^{d} \big\langle \varepsilon_i \big\rangle^{\pm} \boldsymbol{n}_i \otimes \boldsymbol{n}_i,$$

where $\boldsymbol{n}_i$ are the principal directions. The Cauchy stress tensor $\boldsymbol{\sigma} := \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}}$ is also decomposed in an additive manner as

$$\boldsymbol{\sigma} := \big((1-c)^2(1-k)+k\big)\boldsymbol{\sigma}^{+} - \boldsymbol{\sigma}^{-},$$

with

$$\boldsymbol{\sigma}^{\pm} := \sum_{i=1}^{d} \Big(\lambda\big\langle \varepsilon_1+\varepsilon_2+\varepsilon_3\big\rangle^{\pm} + 2\mu\big\langle \epsilon_i\big\rangle^{\pm}\Big)\boldsymbol{n}_i \otimes \boldsymbol{n}_i.$$

A.4 Subdivision surfaces

A.4.1 Catmull–Clark subdivision: Boundary rules

We consider two types of boundary subdivision rules; see Figure A.3. The first type computes the midpoint $\boldsymbol{e}_I^{l+1}$ of a given edge, i.e.,

$$\boldsymbol{e}_I^{l+1} = 0.5\,(\boldsymbol{v}_I^{l} + \boldsymbol{v}_{I+1}^{l}).$$

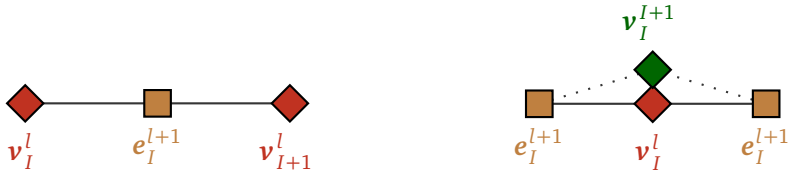

Figure A.3: Catmull–Clark boundary subdivision. *Left:* Edge midpoint construction (brown square). *Right:* Reposition of coarse-level point $\boldsymbol{v}_I^l$ (red diamond) into $\boldsymbol{v}_I^{l+1}$ (green diamond).

The second type repositions the coarse-level control point $\boldsymbol{v}_I^l$ into $\boldsymbol{v}_I^{l+1}$ as

$$\boldsymbol{v}_I^{l+1} = 0.125\ \boldsymbol{e}_I^{l+1} + 0.75\ \boldsymbol{v}_I^l + 0.125\ \boldsymbol{e}_{I+1}^{l+1}.$$

A.4.2 Catmull–Clark reverse subdivision: Reconstruction of coarse-level ghost control points

We assume that the coarse-level inner regular and extraordinary control points have already been constructed. The reverse subdivision scheme for a regular ghost control point reformulates the edge point equation

$$\boldsymbol{e}_I^{l+1} = \frac{(\boldsymbol{e}_I^l + \boldsymbol{v}_I^l + \boldsymbol{f}_I^{l+1} + \boldsymbol{f}_{I+1}^{l+1})}{4}$$

as

$$\boldsymbol{v}_I^l = 4\boldsymbol{e}_I^{l+1} - \boldsymbol{e}_I^l - \boldsymbol{f}_I^{l+1} - \boldsymbol{f}_{I+1}^{l+1}. \tag{A.4}$$

Equation (A.4) states that the $\boldsymbol{v}_I^l$ can be obtained as a weighted sum of the fine-level control points $\boldsymbol{f}_I^{l+1}$, $\boldsymbol{f}_{I+1}^{l+1}$, $\boldsymbol{e}_I^{l+1}$ together with the already reconstructed coarse-level inner control point $\boldsymbol{e}_I^l$.

After the regular ghost points have been reconstructed, the corner ghost control point $\boldsymbol{v}_I^l$ can be found by adjusting the face point equation

$$\boldsymbol{f}_I^{l+1} = \frac{1}{|F_I|}\left(\boldsymbol{v}_I^l + \sum_{i=1}^{N-1} \boldsymbol{v}_{I+i}^l\right)$$

for the given face F_I. This leads to the following reverse subdivision rule:

$$\boldsymbol{v}_I^l = |F_I|\boldsymbol{f}_I^{l+1} - \sum_{i=1}^{N-1} \boldsymbol{v}_{I+i}^l.$$

Figure A.4 demonstrates the described steps.

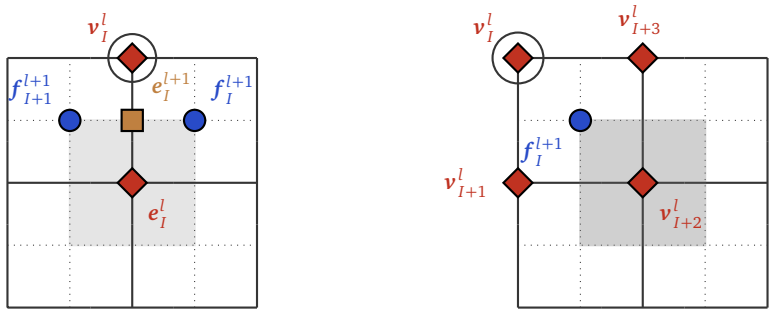

Figure A.4: Reverse subdivision rules. *Left:* Reconstruction of regular ghost point (in the circle). *Right:* Reconstruction of corner ghost point (in the circle).

A.4.3 Catmull–Clark reverse subdivision: Extraordinary control points with valence 3

Assume that the control cage has just one extraordinary control point per face. If necessary, this can be achieved by performing one subdivision step in order to obtain the control cage $\mathcal{T}^1$. The reverse subdivision scheme for an extraordinary control point with $\vartheta = 3$ then follows from (A.4) in Appendix A.4.2.

Appendix B

Software details

B.1 Trust-region subproblem solvers

Table B.1 provides a summary of the trust-region subproblem solvers currently available within UTOPIA library. Solvers implemented using UTOPIA primitives can be executed using linear algebra provided by any backend. Solvers offered directly by the PETSC backend can be used only with the PETSC datatypes.

B.2 RMTR specializations

Table B.2 provides a summary of specializations, which can be used to configure the RMTR classes. Table B.3 provides a summary of specializations, which can be used to determine how to handle constraints in multilevel settings.

B.3 Examples of using UTOPIA trust-region solvers

B.3.1 Optimization problem context

Figure B.1 illustrates interfaces provided by the class `Function`. The optimization problem, i.e. class `MyOptimizationProblem` is then used by the nonlinear solver in order to evaluate function value and its derivatives.

Table B.1: Summary of the trust-region subproblem solvers available in the UTOPIA.

	Solver type	Impl. type		Matrix-free
		PETSc	Frontend	
ℓ_2-norm	Cauchy Point [NW06]	✗	✓	✓
	Moré and Sorensen [Gay83,EM12]	✗	✓	✓
	Dogleg [Pow70a]	✗	✓	✓
	Steihaug-Toint Conjugate Gradient [Ste83,Toi81]	✓	✓	✓
	Generalized Lanczos Conjugate Gradient [GLRT99]	✓	✗	✗
ℓ_∞-norm	Generalized Cauchy Point [CGT00]	✗	✓	✓
	Active Set [Ulb11]	✗	✓	✗
	Successive Coordinate Minimization [GMTW08]	✗	✓	✗
	Modified Proportioning with Gradient Projections [DD07]	✗	✓	✓
	Prolongated Truncated Conjugate Gradient [Yua00]	✓	✗	✗

Table B.2: Summary of specializations for defining level-dependent objective functions.

Specialization	Matrix-based	Matrix-free	Coarse-level model
`FIRST_ORDER_ADDITIVE`	✓	✗	(2.19)
`MF_FIRST_ORDER_ADDITIVE`	✗	✓	(2.19)
`SECOND_ORDER_ADDITIVE`	✓	✗	(2.23)
`MF_SECOND_ORDER_ADDITIVE`	✗	✓	(2.23)
`GALERKIN`	✓	✗	(2.25)
`FIRST_ORDER_MULTIPLICATIVE`	✓	✗	(2.29)
`MF_FIRST_ORDER_MULTIPLICATIVE`	✗	✓	(2.29)
`SECOND_ORDER_MULTIPLICATIVE`	✓	✗	(2.30)
`MF_SECOND_ORDER_MULTIPLICATIVE`	✗	✓	(2.30)
`HYBRID_FIRST_ORDER`	✓	✗	(2.32)
`MF_HYBRID_FIRST_ORDER`	✗	✓	(2.32)

Table B.3: Summary of specializations types for multilevel treatment of constraints. Symbol NA stands for not applicable.

Specialization	Projection rules TR bounds	Projection rules variable bounds
`TRBoundsF`	(2.37)	NA
`TRBoundsGR`	(2.41)	NA
`TRBoundsGM`	(2.44), (2.45)	NA
`TRBoundsF_VRBoundsF`	(2.37)	(2.37)
`TRBoundsGR_VRBoundsGM`	(2.41)	(2.44), (2.45)
`TRBoundsGR_VRBoundsAS`	(2.41)	Active-set method, Section 2.5.4
`TRBoundsGM_VRBoundsGM`	(2.44), (2.45)	(2.44), (2.45)
`TRBoundsGM_VRBoundsAS`	(2.44), (2.45)	Active-set method, Section 2.5.4

```
template<class Matrix, class Vector>
class MyOptimizationProblem : public Function<Matrix, Vector> {
public:
  typedef UTOPIA_SCALAR(Matrix) Scalar;

  bool value(const Vector &x, Scalar &f) const override {
        // evaluation routine for objective function
        return true;
  }

  bool gradient(const Vector &x, Vector &g) const override {
        // evaluation routine for gradient
        return true;
  }

  bool hessian(const Vector &x, Matrix &H) const override {
        // evaluation routine for Hessian
        return true;
  }
};
```

Figure B.1: Implementation of optimization problem, following interfaces of class `Function`.

B.3.2 Trust-region

Figure B.2 provides an example of solving optimization problem using matrix-free trust-region solver, implemented by `MF_TR_2` class.

```
template<class Matrix, class Vector>
void example(const Input & input, const SizeType & n){

  // instantiate the trust-region subproblem solver (TRSS)
  auto subproblem_solver = std::make_shared<SteihaugToint<Matrix, Vector> >();
  // configure the TRSS to use Jacobi preconditioner
  subproblem->set_preconditioner(std::make_shared<Jacobi<Vector> >());
  // instantiate the L-BFGS Hessian approximation scheme
  auto hes_approx = std::make_shared<LBFGS<Vector> >();

  // instantiate the matrix-free trust-region solver (TR radius defined by $\ell_2$-norm)
  MF_TR_2<Vector> tr_solver(hes_approx, subproblem_solver);

  // instantiate the optimization problem
  MyOptimizationProblem<Matrix, Vector> opt_problem;
  // initialize initial guess
  Vector x0(serial_layout(n), 0.0);

  // configure the trust-region solver with user-specified parameters
  tr_solver.read(input);
  // solve optimization problem
  tr_solver.solve(opt_problem, x0);
}
```

Figure B.2: Solving an optimization problem using UTOPIA's TR solver.

B.3.3 Recursive multilevel trust-region

Figure B.3 provides an example of solving optimization problem using RMTR solver, provided by `RMTR_inf` class. The RMTR method is configured with `SECOND_ORDER_ADDITIVE` level-dependent objective functions. The trust-region constraints are handled by means of (2.41).

```
template<class Matrix, class Vector>
void example(const Input& input, const SizeType & L,
          const std::vector<std::shared_ptr<Function<Matrix, Vector>>> & level_functions,
          const std::vector<std::shared_ptr<Transfer<Matrix, Vector>>> & transfers,
          Vector & x){

        // instantiate the trust-region subproblem solver (Successive Coordinate Minimization)
        auto tr_strategy_fine = std::make_shared<utopia::SCM<Matrix, Vector>>();

        // instantiate the trust-region subproblem solver (MPGRP)
        auto tr_strategy_coarse =  std::make_shared<utopia::MPGRP<Matrix, Vector>>();

        // instantiate the RMTR solver (TR radius defined by $\ell_infty}$-norm), with
        // second-order consistent coarse level models of additive type
        // the trust-region bounds are projected using constraint projection rules from (2.41)
        auto rmtr = std::make_shared< RMTR_inf<Matrix, Vector, TRBoundsGR<Matrix, Vector>,
                                                              SECOND_ORDER_ADDITIVE>>(L);

        // set trust-region subproblem solver for the coarsest level
        rmtr->set_coarse_tr_strategy(tr_strategy_coarse);
        // set trust-region subproblem solver for all other levels
        rmtr->set_fine_tr_strategy(tr_strategy_fine);

        // set transfer operators
        rmtr->set_transfer_operators(transfers);

        // set low-cost objective functions (including obj. function of the original opt. problem)
        rmtr->set_functions(level_functions);

        // configure the RMTR with user-specified parameters
        rmtr->read(input);

        // solve optimization problem with initial guess x
        rmtr->solve(x);
}
```

Figure B.3: Solving an optimization problem using UTOPIA's RMTR solver.

Bibliography

[AA12] A. Abdollahi and I. Arias. Phase-field modeling of crack propagation in piezoelectric and ferroelectric materials with different electromechanical crack conditions. *Journal of the Mechanics and Physics of Solids*, 60(12):2100–2126, 2012. 68

[AAB+15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org. 136

[ABB+99] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, et al. *LAPACK Users' guide*. SIAM, 1999. 79

[ABHT03] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss–Seidel. *Journal of Computational Physics*, 188(2):593–610, 2003. 29

[ADL16] M. Ambati and L. De Lorenzis. Phase-field modeling of brittle and ductile fracture in shells with isogeometric NURBS-based solid-shell elements. *Computer Methods in Applied Mechanics and Engineering*, 312:351–373, 2016. 68

[ADLK00] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. MUMPS: a general purpose distributed memory sparse solver. In *International Workshop on Applied Parallel Computing*, pages 121–130. Springer, 2000. 87

[AFMP15] M. Artina, M. Fornasier, S. Micheletti, and S. Perotto. Anisotropic Mesh Adaptation for Crack Detection In Brittle Materials. *SIAM Journal on Scientific Computing*, 37(4):B633–B659, 2015. 68

[AGDL15] M. Ambati, T. Gerasimov, and L. De Lorenzis. A review on phase-field models of brittle fracture and a new fast hybrid formulation. *Computational Mechanics*, 55(2):383–405, 2015. 71

[AINT17] S. Adachi, S. Iwata, Y. Nakatsukasa, and A. Takeda. Solving the trust-region subproblem by a generalized eigenvalue problem. *SIAM Journal on Optimization*, 27(1):269–291, 2017. 12

[AL01] N. M. Alexandrov and R. M. Lewis. An overview of first-order model management for engineering optimization. *Optimization and Engineering*, 2(4):413–430, 2001. 24, 25

[Ale96] N. Alexandrov. Multilevel and multiobjective optimization in multidisciplinary design. In *6th Symposium on Multidisciplinary Analysis and Optimization*, page 4122, 1996. 24

[AMM09] H. Amor, J.-J. Marigo, and C. Maurini. Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. *Journal of the Mechanics and Physics of Solids*, 57(8):1209–1229, 2009. 71

[AMS+14] F. Amiri, D. Millán, Y. Shen, T. Rabczuk, and M. Arroyo. Phase-field modeling of fracture in linear thin shells. *Theoretical and Applied Fracture Mechanics*, 69:102–109, 2014. 68

[AT90] L. Ambrosio and V. M. Tortorelli. Approximation of functional depending on jumps by elliptic functional via Γ-convergence. *Communications on Pure and Applied Mathematics*, 43(8):999–1036, 1990. 67, 71

[Aue93] G. Auer. Numerische behandlung von trust region problemen, 1993. Master's thesis. 13

[BAA+14] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, et al. PETSc users manual revision 3.5. *Argonne National Laboratory (ANL)*, 2014. 60

[BAA+19] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al. PETSc users manual. 2019. 79

[Bab73] I. Babuška. The finite element method with penalty. *Mathematics of computation*, 27(122):221–228, 1973. 102

[Bak66] N. S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966. 21

[BBEM19] J. Brust, O. Burdakov, J. B. Erway, and R. F. Marcia. A dense initialization for limited-memory quasi-Newton methods. *Computational Optimization and Applications*, 74(1):121–142, 2019. 18

[BC83] A. Brandt and C. W. Cryer. Multigrid algorithms for the solution of linear complementarity problems arising from free boundary problems. *SIAM journal on scientific and statistical computing*, 4(4):655–684, 1983. 49

[BC18] K. Bandara and F. Cirak. Isogeometric shape optimisation of shell structures using multiresolution subdivision surfaces. *Computer-Aided Design*, 95:62 – 71, 2018. 103

[BCN18] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018. 136

[BE86] J. W. Barrett and C. M. Elliott. Finite element approximation of the Dirichlet problem using the boundary penalty method. *Numerische Mathematik*, 49(4):343–366, Jul 1986. 102

[Bel99] F. B. Belgacem. The mortar finite element method with Lagrange multipliers. *Numerische Mathematik*, 84(2):173–197, 1999. 75

[BEM17] J. Brust, J. B. Erway, and R. F. Marcia. On solving L-SR1 trust-region subproblems. *Computational Optimization and Applications*, 66(2):245–266, 2017. 17, 136

[BFA02] R. Bridson, R. Fedkiw, and J. Anderson. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Trans. Graph.*, 21(3):594–603, July 2002. 100

[BFKY11] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Multigrid smoothers for ultraparallel computing. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, 2011. 29

[BFM00] B. Bourdin, G. A. Francfort, and J.-J. Marigo. Numerical experiments in revisited brittle fracture. *Journal of the Mechanics and Physics of Solids*, 48(4):797–826, 2000. 68, 71, 88

[BFM08] B. Bourdin, G. A. Francfort, and J.-J. Marigo. The variational approach to fracture. *Journal of elasticity*, 91(1-3):5–148, 2008. 70

[BGZY17] O. Burdakov, L. Gong, S. Zikrin, and Y.-x. Yuan. On efficiently combining limited-memory and trust-region techniques. *Mathematical Programming Computation*, 9(1):101–134, 2017. 17

[BHH06] M. Benzi, E. Haber, and L. Hanson. Multilevel algorithms for large-scale interior point methods in bound constrained optimization. *Preprint*, 2006. 22

[BHNS16] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016. 16

[BJT19] A. S. Berahas, M. Jahani, and M. Takáč. Quasi-Newton methods for deep learning: Forget the past, just sample. *arXiv preprint arXiv:1901.09997*, 2019. 16

[BKK20] V. Braglia, A. Kopaničáková, and R. Krause. A multilevel approach to training. *arXiv preprint arXiv:2006.15602*, 2020. 35

[BKKW17] C. Bilgen, A. Kopaničáková, R. Krause, and K. Weinberg. A phase-field approach to pneumatic fracture. *PAMM*, 17(1):71–74, 2017. 68

[BKKW18] C. Bilgen, A. Kopaničáková, R. Krause, and K. Weinberg. A phase-field approach to conchoidal fracture. *Meccanica*, 53, Apr 2018. 69, 84

[BKKW19] C. Bilgen, A. Kopaničáková, R. Krause, and K. Weinberg. A detailed investigation of the model influencing parameters of the phase-field fracture approach. *GAMM-Mitteilungen*, 0(0):e202000005, 2019. 84

[BLNZ95] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. 16

[BLR11] B. Bourdin, C. J. Larsen, and C. L. Richardson. A time-discrete model for dynamic fracture based on crack regularization. *International journal of fracture*, 168(2):133–143, 2011. 68

[BM+00] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*. Siam, 2000. 22, 28, 56, 154

[BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36. Eurographics Association, 2003. 100

[BMM+10] S. Benson, L. C. McInnes, J. Moré, T. Munson, and J. Sarich. Toolkit for advanced optimization (*tao*) web page. *URL http://www. mcs. anl. gov/tao*, 2010. 87

[BMMR87] R. C. Brower, K. Moriarty, E. Myers, and C. Rebbi. The multigrid method for fermion calculations in quantum chromodynamics. *Multigrid Methods: Theory, Applications, and Supercomputing, SF McCormick, ed*, 110:85–100, 1987. 21

[BNS94] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3):129–156, 1994. 16

[BOS13] S. Burke, C. Ortner, and E. Süli. An adaptive finite element approximation of a generalized Ambrosio–Tortorelli functional. *Mathematical Models and Methods in Applied Sciences*, 23(09):1663–1697, 2013. 71

[BR82] R. E. Bank and D. J. Rose. Analysis of a multilevel iterative method for nonlinear finite element equations. *Mathematics of Computation*, 39(160):453–465, 1982. 22

[Bra73] A. Brandt. Multi-level adaptive technique ($mlat$) for fast numerical solution to boundary value problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, pages 82–89. Springer, 1973. 21

[Bra76] A. Brandt. *Multi-level Adaptive Techniques (MLAT), 1; the Multi-grid Method*. IBM Thomas J. Watson Research Division, 1976. 21

[Bra77] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977. 21, 22, 37

[Bra94] J. H. Bramble. On the development of multigrid methods and their analysis. In *Proceedings of Symposia in Applied Mathematics (W. Gautschi, eds.)*, volume 48, pages 5–19, 1994. 22

[Bro67] C. G. Broyden. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967. 15

[Bro70] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. 15

[BS09] A. Borzì and V. Schulz. Multigrid methods for pde optimization. *SIAM review*, 51(2):361–395, 2009. 23

[BT20] A. S. Berahas and M. Takáč. A robust multi-batch L-BFGS method for machine learning. *Optimization Methods and Software*, 35(1):191–219, 2020. 135

[BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM. 100, 115

[BWX08] J. V. Burke, A. Wiegmann, and L. Xu. Limited memory BFGS updating in a trust-region framework. *SIAM Journal on Optimization*, 2008. 17

[BZ00] J. H. Bramble and X. Zhang. The analysis of multigrid methods. *Handbook of numerical analysis*, 7:173–415, 2000. 22

[C+15] F. Chollet et al. Keras. https://keras.io, 2015. 136

[CC78] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978. 104

[CFI16] S. Conti, M. Focardi, and F. Iurlano. Phase field approximation of cohesive fracture models. In *Annales de l'Institut Henri Poincare (C) Non Linear Analysis*, volume 33, pages 1033–1067. Elsevier, 2016. 68

[CGS19] E. C. Cyr, S. Günther, and J. B. Schroder. Multilevel Initialization for Layer-Parallel Deep Neural Network Training. *arXiv preprint arXiv:1912.08974*, 2019. 125

[CGST93] A. R. Conn, N. Gould, A. Sartenaer, and P. L. Toint. Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. *SIAM Journal on Optimization*, 3(1):164–221, 1993. 8

[CGT91] A. R. Conn, N. I. Gould, and P. L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical programming*, 50(1-3):177–195, 1991. 18

[CGT00] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*. MOS-SIAM Series on Optimization. SIAM, 2000. 1, 8, 12, 13, 14, 17, 22, 74, 124, 160

[CGT12] C. Cartis, N. I. M. Gould, and P. L. Toint. An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. *IMA Journal of Numerical Analysis*, 32(4):1662–1695, 2012. 1

[CGT13] A. R. Conn, G. Gould, and P. L. Toint. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, volume 17. Springer Science & Business Media, 2013. 13

[CHW19] L. Chen, X. Hu, and S. M. Wise. Convergence Analysis of the Fast Subspace Descent Methods for Convex Optimization Problems. *arXiv:1810.04116 [cs, math]*, October 2019. 23

[Cla05] F. Clarke. The maximum principle in optimal control, then and now. *Control and Cybernetics*, 34(3):709, 2005. 126

[CLPS03] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM journal on scientific computing*, 24(3):1076–1089, 2003. 128

[Cly17] D. Clyde. *Numerical Subdivision Surfaces for Simulation and Data Driven Modeling of Woven Cloth*. PhD thesis, UCLA, 2017. 103

[CMH+17] B. Chang, L. Meng, E. Haber, F. Tung, and D. Begert. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*, 2017. 123, 125

[CO01] F. Cirak and M. Ortiz. Fully c^1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 51(7):813–833, 2001. 100

[COS00] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000. 100, 102, 103, 104

[CPK+17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 123

[CR95] Y. Chauvin and D. E. Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology press, 1995. 124, 128

[CTT17a] D. Clyde, J. Teran, and R. Tamstorf. Modeling and Data-driven Parameter Estimation for Woven Fabrics. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 17:1–17:11, New York, NY, USA, 2017. ACM. 100, 101, 102, 103

[CTT17b] D. Clyde, J. Teran, and R. Tamstorf. Simulation of nonlinear Kirchhoff-Love thin shells using subdivision finite elements. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, New York, NY, USA, 2017. ACM. 101, 103

[Dav68] W. C. Davidon. Variance algorithm for minimization. *The Computer Journal*, 10(4):406–410, 1968. 15

[DD07] M. Domorádová and Z. Dostál. Projector preconditioning for partially bound-constrained quadratic optimization. *Numerical Linear Algebra with Applications*, 14(10):791–806, 2007. 13, 153, 160

[dDDB01] J.-R. de Dreuzy, P. Davy, and O. Bour. Hydraulic properties of two-dimensional random fracture networks following a power law length distribution: 1. Effective connectivity. *Water Resources Research*, 37(8):2065–2078, 2001. 85

[Deu11] P. Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35. Springer Science & Business Media, 2011. 69

[Dic10] T. Dickopf. *Multilevel methods based on non-nested meshes*. PhD thesis, University of Bonn, 2010, 2010. 32, 76

[DM79] J. E. Dennis and H. Mei. Two new unconstrained optimization algorithms which use function and gradient values. *Journal of optimization theory and applications*, 28(4):453–482, 1979. 12

[DMS00] T. Dreyer, B. Maar, and V. Schulz. Multigrid optimization in applications. *Journal of Computational and Applied Mathematics*, 120(1-2):67–84, 2000. 22

[Dos09] Z. Dostál. *Optimal quadratic programming algorithms: with applications to variational inequalities*, volume 23. Springer Science & Business Media, 2009. 13, 96

[DPLM07] G. Del Piero, G. Lancioni, and R. March. A variational model for fracture mechanics: numerical experiments. *Journal of the Mechanics and Physics of Solids*, 55(12):2513–2537, 2007. 68

[DS96] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, 1996. 1, 13

[DW98] P. Deuflhard and M. Weiser. Global inexact Newton multilevel FEM for nonlinear elliptic problems. In *Multigrid Methods V*, pages 71–89. Springer, 1998. 22

[EGC04] M. Eldred, A. Giunta, and S. Collis. Second-order corrections for surrogate-based optimization with model hierarchies. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 4457, 2004. 41, 42

[EGG09] J. B. Erway, P. E. Gill, and J. D. Griffin. Iterative methods for finding a trust-region step. *SIAM Journal on Optimization*, 20(2):1110–1131, 2009. 12

[EGMO20] J. B. Erway, J. Griffin, R. F. Marcia, and R. Omheni. Trust-region algorithms for training responses: machine learning methods using indefinite hessian approximations. *Optimization Methods and Software*, 35(3):460–487, 2020. 18, 135

[EK15] H. Esmaeili and M. Kimiaei. An efficient implementation of a trust region method for box constrained optimization. *Journal of Applied Mathematics and Computing*, 48(1-2):495–517, 2015. 17

[EM12] J. B. Erway and R. F. Marcia. Limited-memory BFGS systems with diagonal updates. *Linear algebra and its applications*, 437(1):333–344, 2012. 17, 160

[EM14] J. B. Erway and R. F. Marcia. Algorithm 943: MSS: MATLAB software for L-BFGS trust-region subproblems for large-scale optimization. *ACM Transactions on Mathematical Software (TOMS)*, 40(4):1–12, 2014. 17

[EM17] J. B. Erway and R. F. Marcia. On solving large-scale limited-memory quasi-Newton equations. *Linear Algebra and its Applications*, 515:196–225, 2017. 15, 16

[Fah00] M. Fahl. *Trust-region methods for flow control based on reduced order modeling*. PhD thesis, 2000. 24

[FCZ+17] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2017. 123

[Fed62] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4):1092–1096, 1962. 21

[Fed64] R. P. Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964. 21

[FGB17] C. C. Fischer, R. V. Grandhi, and P. S. Beran. Bayesian low-fidelity correction approach to multi-fidelity aerospace design. In *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0133, 2017. 42

[FGB18] C. C. Fischer, R. V. Grandhi, and P. S. Beran. Bayesian-Enhanced Low-Fidelity Correction Approach to Multifidelity Aerospace Design. *AIAA Journal*, 56(8):3295–3306, 2018. 42

[FJS98] F. Facchinei, J. Júdice, and J. Soares. An active set Newton algorithm for large-scale nonlinear programs with box constraints. *SIAM Journal on Optimization*, 8(1):158–186, 1998. 68

[Fle70] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970. 15

[Flü13] S. Flügge. *Elasticity and Plasticity/Elastizität und Plastizität*. Springer, 2013. 70

[FM90] A. V. Fiacco and G. P. McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*. SIAM, 1990. 15

[FM98] G. A. Francfort and J.-J. Marigo. Revisiting brittle fracture as an energy minimization problem. *Journal of the Mechanics and Physics of Solids*, 46(8):1319–1342, 1998. 68

[FM17] P. Farrell and C. Maurini. Linear and nonlinear solvers for variational phase-field models of brittle fracture. *International Journal for Numerical Methods in Engineering*, 109(5):648–667, 2017. 69

[For06] A. Forsgren. On the behavior of the conjugate-gradient method on ill-conditioned problems. Technical Report TRITA-MAT-2006-OS1, Department of Mathematics, Royal Institute of Technology, January 2006. 114

[FP14] E. Frandi and A. Papini. Coordinate search algorithms in multilevel optimization. *Optimization Methods and Software*, 29(5):1020–1041, 2014. 23

[Gay83] D. M. Gay. Algorithm 611: Subroutines for unconstrained minimization using a model/trust-region approach. *ACM Transactions on Mathematical Software (TOMS)*, 9(4):503–524, 1983. 12, 160

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016. 126, 128, 130, 139, 145

[GG16] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 145

[GG18] W. Gao and D. Goldfarb. Block BFGS methods. *SIAM Journal on Optimization*, 28(2):1205–1231, 2018. 16

[GGM88] W. Gander, G. H. Golub, and U. v. Matt. A constrained eigenvalue problem. *ETH, Eidgen "o sische Technische Hochschule Z ü rich, Institute for Computer Science, Scientific Computing Section*, 92, 1988. 12

[GGR16] R. Gower, D. Goldfarb, and P. Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *International Conference on Machine Learning*, pages 1869–1878, 2016. 16

[GHM80] B. Garbow, K. Hillstrom, and J. More. MINPACK project. *Argonne National Laboratory*, 1980. 13

[Gia05] A. Giacomini. Ambrosio-Tortorelli approximation of quasi-static evolution of brittle fractures. *Calculus of Variations and Partial Differential Equations*, 22(2):129–172, 2005. 68, 71

[GK09a] C. Gross and R. Krause. A new class of non-linear additively preconditioned Trust-Region strategies: Convergence results and applications to non-linear mechanics. *Institute for Numerical Simulation, University of Bonn, INS preprint*, 904, 2009. 23

[GK09b] C. Gross and R. Krause. On the convergence of recursive trust-region methods for multiscale nonlinear optimization and applications to nonlinear mechanics. *SIAM Journal on Numerical Analysis*, 47(4):3044–3069, 2009. 2, 23, 24, 30, 35, 103, 113, 114, 123, 132

[GK09c] C. Gross and R. Krause. A recursive trust-region method for non-convex constrained minimization. In *Domain Decomposition Methods in Science and Engineering XVIII*, pages 137–144. Springer, 2009. 23, 74

[GKS20] C. Gräser, D. Kienle, and O. Sander. Truncated nonsmooth Newton multigrid for phase-field brittle-fracture problems. *arXiv preprint arXiv:2007.12290*, 2020. 69

[GL16] T. Gerasimov and L. D. Lorenzis. A line search assisted monolithic approach for phase-field computing of brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 312:276 – 303, 2016. 68, 69, 74, 90

[GLRT99] N. I. Gould, S. Lucidi, M. Roma, and P. L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999. 12, 160

[GM90] E. Gelman and J. Mandel. On multilevel iterative methods for optimization problems. *Mathematical Programming*, 48(1-3):1–17, 1990. 49, 50

[GMKK20] L. Gaedke-Merzhäuser, A. Kopaničáková, and R. Krause. Multilevel minimization for deep residual networks. *arXiv preprint arXiv:2004.06196*, 2020. 125

[GMS+10] S. Gratton, M. Mouffe, A. Sartenaer, P. L. Toint, and D. Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization. *Optimization Methods and Software*, 25(3):359–386, 2010. 23, 28, 56, 96

[GMT11] S. Gratton, M. Mouffe, and P. L. Toint. Stopping rules and backward error analysis for bound-constrained optimization. *Numerische Mathematik*, 119(1):163–187, 2011. 8

[GMT12] S. Gratton, V. Malmedy, and P. L. Toint. Using approximate secant equations in limited memory methods for multilevel unconstrained optimization. *Computational Optimization and Applications*, 51(3):967–979, 2012. 16

[GMTW08] S. Gratton, M. Mouffe, P. Toint, and M. Weber Mendonca. A recursive ℓ_∞-trust-region method for bound-constrained nonlinear optimization. *IMA Journal of Numerical Analysis*, 28(4):827–861, 2008. 2, 13, 23, 24, 35, 46, 47, 49, 50, 56, 74, 96, 160

[GMVB16] C. Gulcehre, M. Moczulski, F. Visin, and Y. Bengio. Mollifying networks. *arXiv preprint arXiv:1608.04980*, 2016. 123

[GNHLG09] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandie. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768–1778, 2009. 79

[Gol70] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970. 15

[GPR+04] S. Gano, V. Perez, J. Renaud, S. Batill, and B. Sanders. Multilevel variable fidelity optimization of a morphing unmanned aerial vehicle. In *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, page 1763, 2004. 41

[Gra06] S. Gratton. Second-order convergence properties of trust-region methods using incomplete curvature information, with an application to multigrid optimization. *Journal of Computational Mathematics*, pages 676–692, 2006. 56

[Gri21] A. A. Griffith. The phenomena of rupture and flow in solids. *Philosophical transactions of the royal society of london. Series A, containing papers of a mathematical or physical character*, 221:163–198, 1921. 68, 70

[Gro09] C. Gross. *A Unifying Theory for Nonlinear Additively and Multiplicatively Preconditioned Globalization Strategies: Convergence Results and Examples From the Field of Nonlinear Elastostatics and Elastodynamics*. PhD thesis, Universität Bonn, 2009. 30

[GRS05] S. E. Gano, J. E. Renaud, and B. Sanders. Hybrid variable fidelity optimization by using a kriging-based scaling function. *Aiaa Journal*, 43(11):2422–2433, 2005. 42

[GRS+18] S. Günther, L. Ruthotto, J. B. Schroder, E. Cyr, and N. R. Gauger. Layer-parallel training of deep residual neural networks. *arXiv preprint arXiv:1812.04352*, 2018. 125

[GRT10] N. I. Gould, D. P. Robinson, and H. S. Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1):21–57, 2010. 12

[GST06] S. Gratton, A. Sartenaer, and P. L. Toint. Numerical experience with a recursive trust-region method for multilevel nonlinear optimization. *Mathematical Programming, Series A*, 2006. 28, 56

[GST08] S. Gratton, A. Sartenaer, and P. L. Toint. Recursive Trust-Region Methods for Multiscale Nonlinear Optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008. 2, 23, 24, 31, 35, 103, 113, 117, 123, 132

[GT04] S. Green and G. Turkiyyah. Second-order accurate constraint formulation for subdivision finite element simulation of thin shells. *International Journal for Numerical Methods in Engineering*, 61(3):380–405, 2004. 102

[GT06] M. W. Gee and R. S. Tuminaro. Nonlinear Algebraic Multigrid for Constrained Solid Mechanics Problems Using Trilinos. Technical Report SAND2006-2256, Sandia National Laboratories, April 2006. 103

[GTS02] S. Green, G. Turkiyyah, and D. Storti. Subdivision-based Multilevel Methods for Large Scale Engineering Simulation of Thin Shells. In K. Lee and N. M. Patrikalakis, editors, *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, SMA '02, pages 265–272, New York, NY, USA, 2002. ACM. 103

[GVY+18] N. Golmant, N. Vemuri, Z. Yao, V. Feinberg, A. Gholami, K. Rothauge, M. W. Mahoney, and J. Gonzalez. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018. 124

[HA09] C. M. H. Amor, J. J. Marigo. Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. *Journal of the Mechanics and Physics of Solids*, 57:1209–1229, 2009. 68, 71

[Hac76] W. Hackbusch. An iterative method for the fast resolution of elliptic boundary value problems, rep. 76-12. *Institute for Applied Mathematics, University of Cologne, West Germany, Cologne*, pages 77–8, 1976. 21

[Hac85] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer-Verlag Berlin Heidelberg, 1985. 22, 31, 37

[Hag89] W. W. Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989. 15

[Hag01] W. W. Hager. Minimizing a quadratic over a sphere. *SIAM Journal on Optimization*, 12(1):188–208, 2001. 12

[HBH+05] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, et al. An overview of the trilinos project. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):397–423, 2005. 60

[HCB05] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005. 100, 104

[Heb73] M. Hebden. An algorithm for minimization using exact second derivatives. 1973. 12

[Hen18] Henderson, The State University. The Nature of Quartz, 2018. [Online; accessed August 30, 2018]. 83

[HG91] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991. 124

[HGO+17] C. Hesch, A. Gil, R. Ortigosa, M. Dittmann, C. Bilgen, P. Betsch, M. Franke, A. Janz, and K. Weinberg. A framework for polyconvex large strain phase-field methods to fracture. *Computer Methods in Applied Mechanics and Engineering*, 2017. 68

[HHS17] E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017. 124

[Hig02] N. J. Higham. *Accuracy and stability of numerical algorithms*, volume 80. SIAM, 2002. 113

[HK94] R. H. Hoppe and R. Kornhuber. Adaptive multilevel methods for obstacle problems. *SIAM journal on numerical analysis*, 31(2):301–323, 1994. 13, 49

[HKP19] C. P. Ho, M. Kočvara, and P. Parpas. Newton-type multilevel optimization method. *Optimization Methods and Software*, pages 1–34, 2019. 39

[HM83] W. Hackbusch and H. D. Mittelmann. On multi-grid methods for variational inequalities. *Numerische Mathematik*, 42(1):65–76, 1983. 49

[HM17] Y. Heider and B. Markert. A phase-field modeling approach of hydraulic fracture in saturated porous media. *Mechanics Research Communications*, 80:38 – 46, 2017. Multi-Physics of Solids at Fracture. 68

[HN92] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992. 129

[Hod61] R. A. Hodgson. Classification of structures on joint surfaces. *American journal of science*, 259(7):493–502, 1961. 82

[Hop87] R. H. Hoppe. Multigrid algorithms for variational inequalities. *SIAM journal on numerical analysis*, 24(5):1046–1065, 1987. 49

[HPZ16] V. Hovhannisyan, P. Parpas, and S. Zafeiriou. MAGMA: multilevel accelerated gradient mirror descent algorithm for large-scale convex composite minimization. *SIAM Journal on Imaging Sciences*, 9(4):1829–1857, 2016. 23

[HR17] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017. 126, 130, 142

[HRHJ18] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun. Learning Across Scales—Multiscale Methods for Convolution Neural Networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 123, 124

[HSL+16] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 123

[HW14] C. Hesch and K. Weinberg. Thermodynamically consistent algorithms for a finite-deformation phase-field approach to fracture. *International Journal for Numerical Methods in Engineering*, 99(12):906–924, 2014. 68

[HWW15] T. Heister, M. F. Wheeler, and T. Wick. A primal-dual active set method and predictor-corrector mesh adaptivity for computing fracture propagation using a phase-field approach. *Computer Methods in Applied Mechanics and Engineering*, 290:466–495, 2015. 69

[HZRS16a] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 126, 131

[HZRS16b] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 127

[Int07] M. Intel. Intel Math Kernel Library, 2007. 111

[IS15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 145

[JCJ+17] H. Jung, M.-K. Choi, J. Jung, J.-H. Lee, S. Kwon, and W. Young Jung. ResNet-based vehicle classification and localization in traffic surveillance systems. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 61–67, 2017. 123

[JLW20] D. Jodlbauer, U. Langer, and T. Wick. Parallel matrix-free higher-order finite element solvers for phase-field fracture problems. *arXiv preprint arXiv:2005.00331*, 2020. 69

[JS09] M. Juntunen and R. Stenberg. Nitsche's method for general boundary conditions. *Mathematics of computation*, 78(267):1353–1374, 2009. 102

[Kar15] B. Karasözen. Derivative free multilevel optimization. *TWMS Journal of Applied and Engineering Mathematics*, 5(1):46, 2015. 23

[Kau99] L. Kaufman. Reduced storage, quasi-Newton trust region approaches to function optimization. *SIAM Journal on Optimization*, 10(1):56–69, 1999. 17

[KB14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ArXiv*, 2014. 128

[KBLW09] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff-Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3902 – 3914, 2009. 101, 103, 104

[KGSK18] D. Kienle, C. Gräser, O. Sander, and M.-A. Keip. Efficient and reliable phase-field simulation of brittle fracture using a nonsmooth multigrid solution scheme. *PAMM*, 18(1):e201800126, 2018. 69

[KH+09] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. 139

[Kir50] G. Kirchhoff. Über das gleichgewicht und die bewegung einer elastischen scheibe. *Journal für die reine und angewandte Mathematik*, 40:51–88, 1850. 101

[KK01] R. Kornhuber and R. Krause. Adaptive multigrid methods for signorini's problem in linear elasticity. *Computing and Visualization in Science*, 4(1):9–20, 2001. 49

[KK17] A. Kopaničáková and R. Krause. Rook: Parallel framework for phase-field fracture simulations. Git repository, 2017. 79

[KK20a] A. Kopaničáková and R. Krause. Globally Convergent Multilevel Training of Deep Residual Networks. 2020. In preparation. 123

[KK20b] A. Kopaničáková and R. Krause. A recursive multilevel trust region method with application to fully monolithic phase-field models of brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 360:112720, 2020. 67

[KKT19] A. Kopaničáková, R. Krause, and R. Tamstorf. Subdivision-based nonlinear multiscale cloth simulation. *SIAM Journal on Scientific Computing*, 41(5):S433–S461, 2019. 99

[KM16] M. Kočvara and S. Mohammed. A first-order multigrid method for bound-constrained convex optimization. *Optimization Methods and Software*, 31(3):622–644, 2016. 49, 154

[KMN+16] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. 124

[Kor94a] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities i. *Numerische Mathematik*, 69(2):167–184, 1994. 13, 49, 50, 53

[Kor94b] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities i. *Numerische Mathematik*, 69(2):167–184, Dec 1994. 50

[Kor97] R. Kornhuber. *Adaptive monotone multigrid methods for nonlinear variational problems*. Vieweg+ Teubner Verlag, 1997. 51

[KPSC06] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3-4):237–254, 2006. 79

[Kra05] B. Kragel. *Streamline diffusion POD models in optimization*. PhD thesis, 2005. 23

[KSJ+20] A. C. Kirby, S. Samsi, M. Jones, A. Reuther, J. Kepner, and V. Gadepally. Layer-parallel training with gpu concurrency of deep residual neural networks via nonlinear multigrid. *arXiv preprint arXiv:2007.07336*, 2020. 125

[KSK97] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. *Version 1.0, Dept. of Computer Science, University of Minnesota*, page 22, 1997. 79

[KSM15] C. Kuhn, A. Schlüter, and R. Müller. On degradation functions in phase field fracture models. *Computational Materials Science*, 108:374–384, 2015. 71

[KY94] R. Kornhuber and H. Yserentant. Multilevel methods for elliptic problems on domains not resolved by the coarse grid. *Contemporary Mathematics*, 180:49–49, 1994. 49

[KZ16] R. Krause and P. Zulian. A Parallel Approach to the Variational Transfer of Discrete Fields between Arbitrarily Distributed Unstructured Finite Element Meshes. *SIAM Journal on Scientific Computing*, 38(3):C307–C333, 2016. 76

[Lew96] R. Lewis. A trust region framework for managing approximation models in engineering optimization. In *6th Symposium on Multidisciplinary Analysis and Optimization*, page 4101, 1996. 24

[LL80] L. Landau and E. Lifshitz. *Statistical Physics, Part 1: Volume 5*. Butterworth-Heinemann, 1980. 67

[LLMZ16] G. Liu, Q. Li, M. A. Msekh, and Z. Zuo. Abaqus implementation of monolithic and staggered schemes for quasi-static and dynamic fracture phase-field model. *Computational Materials Science*, 121:35 – 47, 2016. 90

[LMB+14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 123

[LN89] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 15

[LN06a] S. Lanquetin and M. Neveu. Reverse Catmull-Clark subdivision. In *Conference proceedings WSCG'2006*. UNION Agency - Science Press, 2006. 109

[LN06b] R. M. Lewis and S. G. Nash. Factors affecting the performance of optimization-based multigrid methods. In *Multiscale Optimization Methods and Applications*, pages 151–172. Springer, 2006. 56

[LN13] R. M. Lewis and S. G. Nash. Using inexact gradients in a multilevel optimization algorithm. *Computational Optimization and Applications*, 56(1):39–61, 2013. 57

[Loo87] C. Loop. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, August 1987. 104

[Man84] J. Mandel. A multilevel iterative method for symmetric, positive definite linear complementarity problems. *applied mathematics and optimization*, 11(1):77–95, 1984. 49

[Mar15] S. Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015. 138

[MAT10] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010. 59

[MHW10] C. Miehe, M. Hofacker, and F. Welschinger. A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. *Computer Methods in Applied Mechanics and Engineering*, 199(45):2765–2778, 2010. 68, 72

[MK16] R. Muller and C. Kuhn. Priority Programme 1748 - Reliable Simulation Techniques in Solid Mechanics: Benchmark: Phase Field Model for Fracture, 2016. oral. 83

[ML84] J. Mandel and J.-L. LIONS. Étude algébrique d'une méthode multigrille pour quelques problèmes de frontière libre. *Comptes rendus des séances de l'Académie des sciences. Série 1, Mathématique*, 298(18):469–472, 1984. 49

[ML91] D. Mahidhara and L. Lasdon. An *sqp* algorithm for large sparse nonlinear programs,". *Austin, MSIS Department–School of Business Administration, University of Texas*, 1991. 16

[ML01] C. Miehe and M. Lambrecht. Algorithms for computation of stresses and elasticity moduli in terms of Seth–Hill's family of generalized strain tensors. *Communications in numerical methods in engineering*, 17(5):337–353, 2001. 79

[MS83] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983. 12

[MS19] R. Mohr and O. Stein. An Adaptive Sample Size Trust-Region Method for Finite-Sum Minimization. *arXiv preprint arXiv:1910.03294*, 2019. 132

[MT91] J. J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991. 13

[MTT02] X. Marduel, C. Tribes, and J.-Y. Trépanier. Optimization using variable fidelity solvers: exploration of an approximation management framework for aerodynamic shape optimization. In *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, page 5595, 2002. 40

[MWH10] C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations. *International Journal for Numerical Methods in Engineering*, 83(10):1273–1311, 2010. 68, 71, 72, 154

[MWW14] A. Mikelic, M. F. Wheeler, and T. Wick. Phase-field modeling of pressurized fractures in a poroelastic medium. *ICES Report*, pages 14–18, 2014. 84

[MWW15a] A. Mikelić, M. F. Wheeler, and T. Wick. Phase-field modeling of a fluid-driven fracture in a poroelastic medium. *Computational Geosciences*, 19(6), Dec 2015. 84

[MWW15b] A. Mikelić, M. F. Wheeler, and T. Wick. A quasi-static phase-field approach to pressurized fractures. *Nonlinearity*, 28(5):1371, 2015. 84

[Nas00] S. G. Nash. A multigrid approach to discretized optimization problems. *Optimization Methods and Software*, 14(1-2):99–116, 2000. 22, 37

[NL11] S. G. Nash and R. M. Lewis. Assessing the performance of an optimization-based multilevel method. *Optimization Methods and Software*, 26(4-5):693–717, 2011. 56

[Noc80] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980. 16

[NW06] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006. 8, 12, 13, 15, 18, 22, 68, 74, 87, 115, 160

[NWC+11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. 139

[OW01] P. Oswald and B. Wohlmuth. On polynomial reproduction of dual FE bases. In *Thirteenth international conference on domain decomposition methods*, pages 85–96, 2001. 76

[PAMM11] K. Pham, H. Amor, J.-J. Marigo, and C. Maurini. Gradient damage models and their use to approximate brittle fracture. *International Journal of Damage Mechanics*, 20(4):618–652, 2011. 71

[Par17] P. Parpas. A multilevel proximal gradient algorithm for a class of composite optimization problems. *SIAM Journal on Scientific Computing*, 39(5):S681–S701, 2017. 23

[PLRR14] P. Parpas, D. V. Luong, D. Rueckert, and B. Rustem. A multilevel proximal algorithm for large scale composite convex optimization. *Submitted, March*, 2014. 23

[Pos15] L. Pospíšil. *Development of Algorithms for Solving Minimizing Problems with Convex Quadratic Function on Special Convex Sets and Applications*. PhD thesis, Vysoká škola báňská-Technická univerzita Ostrava, 2015. 112

[Pow68] M. J. Powell. A fortran subroutine for solving systems of nonlinear algebraic equations. Technical report, Atomic Energy Research Establishment, Harwell, England (United Kingdom), 1968. 13

[Pow70a] M. J. D. Powell. A hybrid method for nonlinear equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Equations*, volume 7, pages 87–114. Gordon & Breach Science Publishers, London, 1970. 12, 160

[Pow70b] M. J. D. Powell. A New Algorithm for Unconstrained Optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, 1970. 12

[Pow85] M. Powell. The performance of two subroutines for constrained optimization on some difficult test problems. *Numerical Optimization I*, 984:160–177, 1985. 18

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 137, 138

[PVN+18] C. Planta, D. Vogler, M. Nestola, P. Zulian, and R. Krause. Variational Parallel Information Transfer between Unstructured Grids in Geophysics-Applications and Solutions Methods. *PROCEEDINGS, 43rd Workshop on Geothermal Reservoir Engineering, Stanford, CA*, pages 1–13, 2018. 83

[PW00] F. A. Potra and S. J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000. 13

[QPK19] A. Quaglino, S. Pezzuto, and R. Krause. High-dimensional and higher-order multifidelity monte carlo estimators. *Journal of Computational Physics*, 388:300–315, 2019. 25

[RDS+15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 123

[Reu87] A. Reusken. Convergence of the multigrid full approximation scheme for a class of elliptic mildly nonlinear boundary value problems. *Numerische Mathematik*, 52(3):251–277, 1987. 22

[Reu88] A. Reusken. Convergence of the multilevel full approximation scheme including thev-cycle. *Numerische Mathematik*, 53(6):663–686, 1988. 22

[RM51] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. 123, 128

[RM18] J. Rafati and R. F. Marcia. Improving L-BFGS initialization for trust-region methods in deep learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 501–508. IEEE, 2018. 18, 136

[Roj07] M. Rojas. LSTRS 1.2: MATLAB Software for Large-Scale Trust-Regions Subproblems and Regularization. 2007. 12

[RSS01] M. Rojas, S. A. Santos, and D. C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM Journal on optimization*, 11(3):611–646, 2001. 12

[Saa03] Y. Saad. *Iterative methods for sparse linear systems*. Siam, 2003. 21, 87, 114

[Sar97] A. Sartenaer. Automatic determination of an initial trust region in nonlinear programming. *SIAM Journal on scientific Computing*, 18(6):1788–1803, 1997. 14

[SF08] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, second edition, 2008. 103

[SG04] O. Schenk and K. Gartner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475 – 487, 2004. Selected numerical algorithms. 114

[Sha70] D. F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970. 15

[SKBN18] J. M. Sargado, E. Keilegavlen, I. Berre, and J. M. Nordbotten. High-accuracy phase-field models for brittle fracture based on a new family of degradation functions. *Journal of the Mechanics and Physics of Solids*, 111:458–489, 2018. 71

[SKM14] A. Schluter, C. Kuhn, and R. Muller. Phase Field Approximation of Dynamic Brittle Fracture. *PAMM*, 14(1):143–144, 2014. 68

[SPSP03] F. Samavati, H.-R. Pakdel, C. Smith, and P. Prusinkiewicz. Reverse Loop Subdivision. Technical Report 2003-730-33, University of Calgary, November 2003. 110

[SS97] E. M. Simantiraki and D. F. Shanno. An infeasible-interior-point method for linear complementarity problems. *SIAM Journal on Optimization*, 7(3):620–640, 1997. 17

[Sta98] J. Stam. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 395–404. ACM, 1998. 107

[Ste83] T. Steihaug. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983. 12, 29, 160

[Stu17] P. A. Studios. OpenSubdiv, 2017. 104, 111

[SZ14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 124

[Tis01] F. Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1038–1057, 2001. 114

[TJM15] R. Tamstorf, T. Jones, and S. F. McCormick. Smoothed Aggregation Multigrid for Cloth Simulation. *ACM Trans. Graph.*, 34(6):245:1–245:13, October 2015. 103

[Toi81] P. L. Toint. Towards an Efficient Sparsity Exploiting Newton Method for Minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88. Academic Press, London, England, 1981. 12, 29, 160

[TOS00] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Elsevier, 2000. 22

[TTWM09] P. L. Toint, D. Tomanos, and M. Weber-Mendonca. A multilevel algorithm for solving the trust-region subproblem. *Optimization Methods & Software*, 24(2):299–311, 2009. 12

[TWS06] B. Thomaszewski, M. Wacker, and W. Straßer. A Consistent Bending Model for Cloth Simulation with Corotational Subdivision Finite Elements. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '06, pages 107–116. The Eurographics Association, 2006. 100

[TX98] X.-C. Tai and J. Xu. Subspace correction methods for convex optimization problems. In *Eleventh International Conference on Domain Decomposition Methods (London, 1998)*, pages 130–139, 1998. 23

[TX02] X.-C. Tai and J. Xu. Global and uniform convergence of subspace correction methods for some convex optimization problems. *Mathematics of Computation*, 71(237):105–124, 2002. 23

[Ulb11] M. Ulbrich. *Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces*, volume 11. SIAM, 2011. 13, 68, 87, 160

[UZ17] S. Ulbrich and J. C. Ziems. Adaptive multilevel trust-region methods for time-dependent pde-constrained optimization. *Portugaliae Mathematica*, 74(1):37–67, 2017. 23

[Val10] M. Vallejos. MGOPT with gradient projection method for solving bilinear elliptic optimal control problems. *Computing*, 87(1-2):21–33, 2010. 49

[VMDBV14] J. Vignollet, S. May, R. De Borst, and C. V. Verhoosel. Phase-field models for brittle and cohesive fracture. *Meccanica*, 49(11):2587–2601, 2014. 68, 69

[VRG09] C. V. Verhoosel, J. J. Remmers, and M. A. Gutiérrez. A dissipation-based arc-length method for robust simulation of brittle and ductile failure. *International Journal for Numerical Methods in Engineering*, 77(9):1290–1321, 2009. 69

[WCV11] S. v. d. Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011. 136

[Wei17] E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017. 124, 125

[Wes95] P. Wesseling. Introduction to multigrid methods. 1995. 22

[WG10] Z. Wen and D. Goldfarb. A line search multigrid method for large-scale nonlinear optimization. *SIAM Journal on Optimization*, 20(3):1478–1503, 2010. 22, 56

[WGH+20] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl. A multigrid method for efficiently training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–162, 2020. 125

[Wic17a] T. Wick. An Error-Oriented Newton/Inexact Augmented Lagrangian Approach for Fully Monolithic Phase-Field Fracture Propagation. *SIAM Journal on Scientific Computing*, 39(4):B589–B617, 2017. 69

[Wic17b] T. Wick. Modified Newton methods for solving fully monolithic phase-field quasi-static brittle fracture propagation. *Computer Methods in Applied Mechanics and Engineering*, 325:577–611, 2017. 69, 74

[WK03] B. I. Wohlmuth and R. H. Krause. Monotone multigrid methods on nonmatching grids for nonlinear multibody contact problems. *SIAM journal on scientific computing*, 25(1):324–347, 2003. 75

[Woh00] B. I. Wohlmuth. A mortar finite element method using dual spaces for the Lagrange multiplier. *SIAM journal on numerical analysis*, 38(3):989–1012, 2000. 74

[Won11] E. L. S. Wong. *Active-set methods for quadratic programming*. PhD thesis, UC San Diego, 2011. 13

[WSC+16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 123

[WWW14] M. Wheeler, T. Wick, and W. Wollner. An augmented-Lagrangian method for the phase-field approach for pressurized fractures. *Computer Methods in Applied Mechanics and Engineering*, 271:69 – 85, 2014. 68

[XB07] L. Xu and J. Burke. ASTRAL: An active set ℓ_∞-trust-region algorithm for box constrained optimization. *Optimization online, July*, 2007. 17

[XRV17] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-Mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 139

[XWA+18] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. The Microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE, 2018. 123

[YD06] I. Yavneh and G. Dardyk. A Multilevel Nonlinear Method. *SIAM Journal on Scientific Computing*, 28(1):24–46, 2006. 38, 39

[Yua00] Y. Yuan. On the truncated conjugate gradient method. *Mathematical Programming*, 87(3):561–573, 2000. 13, 29, 160

[YWN+18] M. Yu, C. Wei, L. Niu, S. Li, and Y. Yu. Calculation for tensile strength and fracture toughness of granite with three kinds of grain sizes using three-point-bending test. *PloS one*, 13(3), 2018. 84

[ZKN+16] P. Zulian, A. Kopaničáková, M. C. G. Nestola, A. Fink, N. Fadel, A. Rigazzi, V. Magri, T. Schneider, E. Botter, J. Mankau, and R. Krause. Utopia: A C++ embedded domain specific language for scientific computing. Git repository. https://bitbucket.org/zulianp/utopia, 2016. 3, 59, 79

[ZKN+20] P. Zulian, A. Kopaničáková, M. G. C. Nestola, A. Fink, N. A. Fadel, J. Vandevondele, and R. Krause. Large scale simulation of pressure induced phase-field fracture propagation using Utopia. *arXiv preprint arXiv:2007.12908*, 2020. 29, 58

[Zor00] D. Zorin. A Method for Analysis of c^1-Continuity of Subdivision Surfaces. *SIAM Journal on Numerical Analysis*, 37(5):1677–1708, 2000. 104

[ZS18] L.-H. Zhang and C. Shen. A nested Lanczos method for the trust-region subproblem. *SIAM Journal on Scientific Computing*, 40(4):A2005–A2032, 2018. 12

[ZU11] J. C. Ziems and S. Ulbrich. Adaptive multilevel inexact sqp methods for pde-constrained optimization. *SIAM Journal on Optimization*, 21(1):1–40, 2011. 23