Master's Thesis

Inexact Newton with Iterated Sensitivities for Optimization of Partial Differential Equation

Justin Asbjørn Pearse-Danker

Albert-Ludwig-University Freiburg

Faculty of Mathematics and Physics

Department of Mathematics

Chair for System Theory, Control and Optimization

March 26th, 2020

Writing Period

26. 09. 2020 – 26. 03. 2020

Examiner

Prof. Dr. Moritz Diehl

Second Examiner

Prof. Dr. Soeren Bartels

Adviser

Jonathan Frey

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Acknowledgments

First and foremost, I would like to thank my supervisor Prof. Dr. Moritz Diehl for giving me the opportunity to write this thesis, providing guidance and sharing his fruitful ideas.
I gratefully acknowledge my advisor Jonathan Frey for his continuous support, helpful advice and careful proofreading of this thesis. Moreover, I would like to thank all the people in the syscop office for creating an atmosphere which made working on this thesis a pleasant experience.
Most importantly, I wish to thank my loving and supportive wife, Sarah: The whole journey here in Freiburg was a challenging, exciting and life-changing experience which would have never been possible with out you. Furthermore, I would like to thank my son Ben, who is a true inspiration and important part of my life.
I also want to give thanks to my parents, sister and brother, who continuously supported me and always had an open ear for my dreams and ideas.
Last but not least, I want to thank all my friends, who made the time here in Freiburg such a great experience.

Abstract

Solving optimal control problems subject to partial differential equations (PDEs) is a challenging but important task, for instance, optimal heating or cooling.
The problem arising from the discretization can easily have 10^6 decision variables and therefore it is important to provide methods that can solve such problems efficiently. Thus, the interaction between the optimization method and the numerical simulation is crucial for such problems.
The Inexact Newton with Iterated Sensitivities (INIS) method solves a particular class of nonlinear programming (NLP) problems, which arise from optimal control formulations where a subset of the variables are implicitly defined by nonlinear equality constraints. The system of nonlinear equality constraints is called the forward problem. In contrast to other inexact Newton-type optimization methods, the INIS method preserves the local convergence properties and the asymptotic contraction rate of the inexact Newton-type method with the same Jacobian approximation applied to the forward problem. The INIS method is especially suited for problems where the number of states is significantly larger than the number of controls. This is the case for PDE constrained optimal control problems with boundary controls, which we regard in this thesis.
Moreover, we consider a forward problem that arises from the discretization of a PDE defined on a 2-dimensional domain. An efficient method for solving linear systems that result from the discretization of PDEs is the multi-grid (MG) method.
This thesis shows that the MG method can be combined with the INIS method resulting in the INIS-MG algorithm. The algorithm is applied to a test problem and compared to `ipopt`, a software package for large-scale nonlinear optimization. Furthermore, the theoretical properties of the INIS method are verified for the PDE constrained case.

The numerical experiments show that even a `MATLAB` implementation of the INIS-MG algorithm can outperform state of the art NLP solvers like `ipopt`.

Zusammenfassung

Das Lösen von Optimalsteuerungsproblemen mit Nebenbedingungen aus partiellen Differentialgleichungen (PDE) ist eine herausfordernde aber wichtige Aufgabe, etwa für optimales Heizen oder Kühlen. Das Problem, das sich aus der Diskretisierung ergibt, kann leicht 10^6 Variablen beinhalten, daher ist es wichtig, dass es Methoden gibt, die solche Problem effizient lösen. Somit ist das Zusammenspiel der Optimierungsmethode und der numerischen Simulation wichtig.

Die inexakte Newton Methode mit iterierten Sensitivitäten (INIS) löst eine bestimmte Klasse von nichtlinearen Optimierungsproblemen (NLP), die durch Optimalsteuerungsproblemen entstehen, in denen eine Teilmenge der Variablen implizit durch nichtlineare Gleichheitsbedingungen definiert ist. Diese nichtlinearen Gleichheitsbedingungen werden als Vorwärtsproblem bezeichnet. Im Gegensatz zu anderen inexakten Newton Methoden, erhält die INIS Methode lokale Konvergenzeigenschaften und die asymptotische Kontraktionsrate der inexakten Newton Methode, die mit derselben Approximation der Jakobischen auf das Vorwärtsproblem angewendet wird. Das INIS Verfahren ist vorallem für Probleme geeignet bei denen die Anzahl der Zustände deutlich größer ist als die der Kontrollen. Dies ist der Fall für Optimalsteuerungsprobleme mit Nebenbedingungen aus PDEs und Randkontrollen, welche wir in dieser Arbeit betrachten.

Darüber hinaus betrachten wir ein Vorwärtsproblem, das durch die Diskretisierung einer zweidimensionalen PDE entsteht. Ein effizientes Verfahren zum Lösen von Gleichungssystemen, die durch die Diskretisierung von PDEs entstehen, ist das Mehrgitterverfahren. Diese Arbeit zeigt, dass das Mehrgitterverfahren mit der INIS Methode kombiniert werden kann und vereint beide Verfahren in dem INIS-MG Algorithmus. Dieser Algorithmus wird auf ein Testproblem angewendet und mit `ipopt`, ein Softwarepaket für nichtlineare Optimierung, verglichen. Darüber hinaus

werden theoretische Eigenschaften des INIS Verfahrens für Optimierungsprobleme mit PDEs als Nebenbedingung nachgewiesen.
Die numerischen Experimente zeigen, dass sogar eine `MATLAB` Implementierung des INIS-MG Algorithmus aktuelle NLP Löser wie `ipopt` schlagen kann.

Contents

List of Figures

List of Algorithms

1 Introduction

1.1 Motivation

Optimal control problems with PDEs constraints often occur in the context of industrial and medical applications. In the production process of steel mills, selective intermediate cooling of steel profiles aims on reducing the total heat content while simultaneously equalizing the interior temperature distribution. Reducing the temperature as uniformly as possible results in a higher quality steel. Furthermore, expensive cooling beds can be replaced due to accelerated cooling of the steel [1, 2, 3].
Another application is the local heating of tumor tissues in the field of cancer therapy. Here, the aim is to generate a temperature distribution in the human body such that mainly the tumor is heated and not any other tissue. High temperatures can kill tumor cells but also injure healthy cells that are not effected by the tumor. Hence, the temperature distribution has to be carefully controlled [4].
Proper discretization of PDE constrained optimization problems can lead to large-scale problems with optimization variables between 10^3 and 10^{10} [5]. Due to the constantly improving computing power such problems can be attacked. However, it is important to further improve efficient methods and algorithms for solving large scale problems.

1.2 Outline and Contribution

- **Chapter 2** gives some background to numerical optimization and repeats Newton-type optimization for equality constrained optimization problems.

- In **Chapter 3** we introduce the forward problem and a condensed version of an inexact Newton method. Furthermore we in present the INIS method in its simplest form.
- The starting point of **Chapter 4** is the discretization of the Poisson equation.We then use this discretization to explain and apply the MG method. Completing this chapter with a linearity result of the MG method.
- In **Chapter 5** we introduce a special optimization problem where we can apply the INIS method in combination with the MG method resulting in the INIS-MG algorithm.
- **Chapter 6** shows experimental results testing the MG method and the INIS-MG algorithm on a test problem.
- Concluding with **Chapter 7** where we summarize the results and give ideas for future research.

2 Numerical Optimization for Equality Constrained Optimization

In this chapter we consider a nonlinear programming (NLP) problem of the form

$$\min_{y \in \mathbb{R}^{n_\mathrm{y}}} \quad f(y), \tag{2.1a}$$

$$\text{subject to} \quad g(y) = 0. \tag{2.1b}$$

The function $f : \mathbb{R}^{n_\mathrm{y}} \to \mathbb{R}$ is called the *objective function* and $g : \mathbb{R}^{n_\mathrm{y}} \to \mathbb{R}^{n_g}$ are the *equality constraints*. Both functions are assumed to be twice continuously differentiable. The goal is to minimize the objective function using the *decision variables y*. Due to the constraints they can not be chosen completely freely. We collect all points that fulfil these constraints in a set.

Definition 1 (Feasible set)**.** The *feasible set* is defined as

$$\Omega := \{y \in \mathbb{R}^{n_\mathrm{y}} \mid g(y) = 0\}. \tag{2.2}$$

Definition 2 (Local minimizer)**.** The point $y^* \in \mathbb{R}^{n_\mathrm{y}}$ is called *local minimizer*, if and only if $y^* \in \Omega$ and there exists a neighbourhood $\mathcal{N}$ of y^* such that for all $y \in \Omega \cap \mathcal{N}$ holds $f(y^*) \leq f(y)$.

2.1 Optimality Conditions

Next, we state necessary and sufficient conditions for optimality to be able to rate a feasible point either as a hot candidate for a local minimizer or no candidate at all

[6, 7].

First, we characterize the set of feasible directions for a feasible point $x^* \in \Omega$. We are interested in all directions $p \in \mathbb{R}^{n_\mathrm{y}}$ such that moving in that direction contains exclusively feasible points. We call these directions *tangent vectors* which are defined as follows.

Definition 3 (Tangent Vector)**.** For a feasible point $y^* \in \Omega$ a vector $p \in \mathbb{R}^{n_\mathrm{y}}$ is called a *tangent vector* to Ω at y^* if there exist a smooth curve

$$\bar{y}(t) : [0, \varepsilon) \to \mathbb{R}^{n_\mathrm{y}} \tag{2.3}$$

with $\bar{y}(0) = y^*$, $\bar{y}(t) \in \Omega$ and $\frac{\mathrm{d}\bar{y}}{\mathrm{d}t}(0) = p$.

Definition 4 (Tangent Cone)**.** The *tangent cone* $\mathrm{T}_\Omega(y^*)$ of Ω at a feasible point $y^* \in \Omega$ is the set of all tangent vectors at y^*.

The tangent cone can often be obtained by an algebraic description, but this is only possible under some condition which we call *linear independence constraint qualification.*

Definition 5. (LICQ) The *linear independence constraint qualification* (LICQ) holds at a point $y^* \in \Omega$ if and only if the gradients $\nabla g_i(y^*)$ for $i \in \{1, \dots, n_g\}$ are linearly independent.

Definition 6 (Linearised Feasible Cone)**.** For a point $y^* \in \Omega$ the *linearised feasible cone* is defined as

$$\mathcal{F}(y^*) = \{p \in \mathbb{R}^{n_\mathrm{y}} \,|\, \nabla g_i(y^*)^\top p = 0 \;\text{ for } i = 1, \dots, n_g\}. \tag{2.4}$$

The following theorem states under which assumptions the linearised feasible cone contains infeasible directions and gives an algebraic description of the tangent cone.

Theorem 1. At any $y^* \in \Omega$ the following statements hold:

1: $\mathrm{T}_\Omega(y^*) \subset \mathcal{F}(y^*)$,

2: If LICQ holds at y^* then $\mathrm{T}_\Omega(y^*) = \mathcal{F}(y^*)$.

Proof. See [6]. □

With the *Lagrangian function* defined as

$$\mathcal{L}(y, \lambda) = f(y) + \lambda^\top g(y), \tag{2.5}$$

with so called *Lagrange multipliers* $\lambda \in \mathbb{R}^{n_g}$ we can formulate the *Karush-Kuhn-Tacker* (KKT) conditions which are also known as the *first order necessary conditions* (FONC) for optimality-

Theorem 2 (KKT Conditions)**.** If y^* is a local minimizer of the NLP (2.1) and the LICQ holds at y^*, then there exists a multiplier $\lambda^* \in \mathbb{R}^{n_g}$ such that

$$\nabla_y \mathcal{L}(y^*, \lambda^*) = 0, \tag{2.6a}$$

$$g(y^*) = 0. \tag{2.6b}$$

Proof. A proof can be found in [6] where different variants of FONC are formulated which simplify the conditions step by step and lead to the KKT conditions. □

A point $(y^*, \lambda^*) \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_g}$ where LICQ and the KKT conditions hold is called *KKT point.*
To formulate the KKT conditions we have only used first order information. Because we assumed f and g to be twice continuous differentiable we can also state *second order necessary conditions* (SONC) and *second order sufficient conditions* (SOSC).

Theorem 3 (Second Order Optimality Conditions)**.** If (y^*, λ^*) is a KKT point the the following statements holds:

SONC: If y^* is a local minimizer, then

$$p^\top \nabla_y^2 \mathcal{L}(y^*, \lambda^*) p \geq 0, \tag{2.7}$$

for all $p \in \mathcal{F}(y^*)$.

SOSC: If

$$p^\top \nabla_y^2 \mathcal{L}(y^*, \lambda^*) p > 0, \tag{2.8}$$

for all $p \in \mathcal{F}(y^*)$ with $p \neq 0$, then y^* is a local minimizer which is unique in its neighbourhood.

Definition 7 (Regular KKT Point)**.** A minimizer of an equality constrained NLP is called a *regular KKT point*, if both LICQ and SOSC are satisfied at this KKT point.

2.2 Newton-Type Optimization

The basic idea of Newton-type optimization to solve NLPs of the form (2.1) locally is to apply Newton-type methods to find a solution of the KKT conditions (2.6).

2.2.1 Quadratic Model Interpretation derived by the Framework of Sequential Quadratic Programming (SQP) Methods

The SQP method approximates the NLP (2.1) by a quadratic model and iteratively solves a sequence of optimization subproblems [8].

For a current iterate (y^k, λ^k) the SQP method solves the QP

$$\min_{\delta y \in \mathbb{R}^{n_y}} \quad \frac{1}{2}\delta y^\top \nabla_y^2 \mathcal{L}(y^k, \lambda^k)\delta y + \nabla_y \mathcal{L}(y^k, \lambda^k)\delta y \tag{2.9a}$$

$$\text{subject to} \quad \nabla g(y^k)\delta y + g(y^k) = 0. \tag{2.9b}$$

and updates the iterates by the scheme

$$\begin{aligned} y^{k+1} &= y^k + \delta y, \\ \lambda^{k+1} &= \lambda^k + \delta\lambda. \end{aligned} \tag{2.10}$$

where δy and $\delta\lambda$ are obtained as the solution of the KKT conditions

$$\begin{bmatrix} \nabla_y^2 \mathcal{L}(y^k, \lambda^k)\delta y + \nabla_y \mathcal{L}(y^k, \lambda^k) + \nabla g(y^k)\delta\lambda \\ \nabla g(y^k)\delta y + g(y^k) \end{bmatrix} = 0. \tag{2.11}$$

Rewritting these conditions as matrix vector product leads to

$$\begin{bmatrix} \nabla_y^2 \mathcal{L}(y^k, \lambda^k) & \nabla g(y^k) \\ \nabla g(y^k) & \mathbb{0} \end{bmatrix} \begin{bmatrix} \delta y \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_y \mathcal{L}(y^k, \lambda^k) \\ g(y^k) \end{bmatrix}. \tag{2.12}$$

2.2.2 Local contraction

We now want to formulate conditions for convergence of Newton-type optimization. The linear system (2.12) can also be derived by applying Netwon's method to the KKT conditions

$$F(y, \lambda) := \begin{bmatrix} \nabla_y \mathcal{L}(y, \lambda) \\ g(y) \end{bmatrix} = 0, \tag{2.13}$$

of the NLP (2.1). For the exact Jacobian $\frac{\partial F}{\partial(y,\lambda)}$ we introduce the notation

$$J(y, \lambda) := \frac{\partial F}{\partial(y, \lambda)}(y, \lambda). \tag{2.14}$$

Consequently, the iterates (2.10) of the SQP method are exactly those generated by the Newton iteration

$$\begin{bmatrix} y^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} y^{k} \\ \lambda^{k} \end{bmatrix} - J(y^k, \lambda^k)^{-1} F(y^k, \lambda^k). \tag{2.15}$$

In order to reduce computational cost we could use an approximation $\tilde{J}(y, \lambda) \approx J(y, \lambda)$ of the exact Jacobian. This leads to a so called *Newton-type iteration*

$$\begin{bmatrix} y^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} y^{k} \\ \lambda^{k} \end{bmatrix} - \tilde{J}(y^k, \lambda^k)^{-1} F(y^k, \lambda^k). \tag{2.16}$$

We define the *spectral radius* $\rho(B)$ of a square matrix B as follows

$$\rho(B) = \max\{|\mu| \mid \mu \text{ is Eigenvalue of } B\}. \tag{2.17}$$

The following theorem states sufficient and necessary conditions for local convergence of Newton-type iterations.

Theorem 4 (Local Newton-type contraction [6])**.** We consider the twice continuously differentiable function $F(y, \lambda)$ from (2.13) and a regular KKT point (y^*, λ^*) with $F(y^*, \lambda^*) = 0$. We then apply the Newton-type iteration with $\tilde{J}(y^k, \lambda^k) \approx J(y^k, \lambda^k)$. We assume $\tilde{J}(\cdot, \cdot)$ to be continuously differentiable and invertible in a neighbourhood of the solution. If all eigenvalues of the iteration matrix have a modulus smaller than 1, i.e., if the spectral radius satisfies

$$\kappa^* := \rho(\tilde{J}(y^*, \lambda^*)^{-1} J(y^*, \lambda^*) - \mathbb{1}_{n_F}) < 1 \tag{2.18}$$

then this fixed point (y^*, λ^*) is asymptotically stable, where $n_F = n_y + n_g$. Additionally, the iterates (y^k, λ^k) converge linearly to the KKT point (y^*, λ^*) with the asymptotic contraction rate κ^* when initialized sufficiently close. On the other hand, if $\kappa^* > 1$, then the fixed point (y^*, λ^*) is unstable.

3 Inexact Newton with Iterated Sensitivities (INIS)

In this chapter we introduce the INIS method and results from [9]. We apply the INIS method to a particular class of NLP problems

$$\min_{z\in\mathbb{R}^{n_\mathrm{z}},\, w\in\mathbb{R}^{n_\mathrm{w}}} \quad f(z,w) \tag{3.1a}$$

$$\text{subject to} \quad g(z,w)=0 \tag{3.1b}$$

where $f:\mathbb{R}^{n_\mathrm{z}}\times\mathbb{R}^{n_\mathrm{w}}\to\mathbb{R}$ and $g:\mathbb{R}^{n_\mathrm{z}}\times\mathbb{R}^{n_\mathrm{w}}\to\mathbb{R}^{n_g}$ are again assumed to be twice continuously differentiable. Furthermore we assume $n_g=n_\mathrm{z}$ and that the Jacobian $g_z:=\frac{\partial g(z,w)}{\partial z}$ is invertible. We introduce the vector of the decision variables $y\in\mathbb{R}^{n_\mathrm{y}}$ which consist of the *states* z and the *controls* w, i.e. $y=[z^\top,w^\top]^\top$ and $n_\mathrm{y}=n_\mathrm{z}+n_\mathrm{w}$. The INIS method can be generalized for NLPs that contains inequality constraints and additional equality constraints [9, 10] but the main focus of this thesis are problems of the form (3.1).

Due to the invertibility of g_z it follows that the variables z are implicitly defined via the equality constraints $g(z,w)=0$, i.e. for a given w^* we can solve

$$g(z,w^*)=0 \tag{3.2}$$

to obtain the solution $z^*(w^*)$. This subproblem is called the *forward problem.*

We solve it by applying Newton's root finding method to obtain

$$\delta z^k=-g_z(z^k,w^*)^{-1}g(z^k,w^*). \tag{3.3}$$

Instead of using the exact Jacobian g_z we can use a full-rank approximation, $M \in \mathbb{R}^{n_\mathrm{z} \times n_\mathrm{z}}$

$$M \approx g_z, \tag{3.4}$$

and apply an inexact Newton method with updates of the form

$$\delta z^k = -M^{-1} g(z^k, w^*). \tag{3.5}$$

3.1 Condensed Inexact Newton (IN) for Equality Constrained Optimization

In order to solve (3.1) we can apply the SQP method. With the Lagrangian $\mathcal{L}(y, \lambda) = f(y) + \lambda^\top g(y)$ the QP that has to be solved in each iteration reads as

$$\min_{\delta y \in \mathbb{R}^{n_\mathrm{y}}} \quad \frac{1}{2} \delta y^\top \nabla_y^2 \mathcal{L}(y^k, \lambda^k) \delta y + \nabla_y \mathcal{L}(y^k, \lambda^k) \delta y, \tag{3.6a}$$

$$\text{subject to} \quad g_z(y^k)\delta z + g_w(y^k)\delta w + g(y^k) = 0, \tag{3.6b}$$

with the decision variables $y^k = \left[(z^k)^\top, \, (w^k)^\top \right]^\top$.

Instead of solving the KKT conditions of this QP directly we apply a null space approach. For a given $\delta \bar{w}$ we can pre-solve the forward problem (3.6b)

$$\delta \bar{z} = -g_z(y^k)^{-1}(g(y^k) + g_w(y^k)\delta \bar{w}) \tag{3.7}$$

and obtain a feasible direction $\delta \bar{y} = \left[\delta \bar{z}^\top, \, \delta \bar{w}^\top \right]^\top$.

With a basis

$$Z^\top = \begin{bmatrix} -g_w(y^k)^\top g_z(y^k)^{-\top}, & \mathbb{1}_{n_\mathrm{w}} \end{bmatrix} \in \mathbb{R}^{n_\mathrm{w} \times n_\mathrm{y}}, \tag{3.8}$$

of the nullspace of the constraint Jacobian, i.e. $g_y(y^k)Z = \mathbb{0}$, we can locally express any other feasible direction in the form

$$\delta y = \delta \bar{y} + Z\delta w \tag{3.9}$$

for $\delta w \in \mathbb{R}^{n_\mathrm{w}}$. Substituting δy in (3.6) yields the unconstrained problem

$$\min_{\delta w \in \mathbb{R}^{n_\mathrm{w}}} \quad \frac{1}{2}\delta w^\top Z^\top \nabla_y^2 \mathcal{L}(y^k, \lambda^k) Z\delta w + (\nabla_y^2 \mathcal{L}(y^k, \lambda^k)\delta\bar{y} + \nabla_y \mathcal{L}(y^k, \lambda^k))^\top Z\delta w. \tag{3.10}$$

The solution is then obtained by solving the system

$$(Z^\top \nabla_y^2 \mathcal{L}(y^k, \lambda^k) Z)\delta w = -Z^\top (\nabla_y^2 \mathcal{L}(y^k, \lambda^k)\delta\bar{y} + \nabla_y \mathcal{L}(y^k, \lambda^k)), \tag{3.11}$$

and can be expanded back to the full variable space $\mathbb{R}^{n_\mathrm{y}}$ using Equation (3.6b). In order to calculate $\delta\lambda$ and update λ^k we apply an inexact Newton method to the KKT conditions for the NLP (3.1)

$$\begin{bmatrix} \nabla_y^2 \mathcal{L}(y^k, \lambda^k) & \begin{pmatrix} g_z(y^k)^\top \\ g_w(y^k)^\top \end{pmatrix} \\ \begin{pmatrix} g_z(y^k) & g_w(y^k) \end{pmatrix} & \mathbb{0} \end{bmatrix} \begin{bmatrix} \delta z \\ \delta w \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_y \mathcal{L}(y^k, \lambda^k) \\ g(y^k) \end{bmatrix}. \tag{3.12}$$

We can write down the first n_z equations as

$$g_z(y^k)^\top \delta\lambda = -[\mathbb{1}_{n_\mathrm{z}} \quad \mathbb{0}](\nabla_y \mathcal{L}(y^k, \lambda^k) + \nabla_y^2 \mathcal{L}(y^k, \lambda^k)\delta y) \tag{3.13}$$

and thus we have

$$\delta\lambda = -[g_z(y_k)^{-\top} \quad \mathbb{0}](\nabla_y \mathcal{L}(y^k, \lambda^k) + \nabla_y^2 \mathcal{L}(y^k, \lambda^k)\delta y). \tag{3.14}$$

This idea is summarized in Algorithm 1 where we used the approximations $\tilde{H} \approx \nabla_y^2 \mathcal{L}(y^k, \lambda^k)$, (3.4) and

$$\tilde{Z}^\top = \begin{bmatrix} -g_w(y^k)^\top M^{-\top}, & \mathbb{1}_{n_\mathrm{w}} \end{bmatrix} \approx Z^\top. \tag{3.15}$$

Algorithmus 1 : `inexact_newton`(y^k, λ^k, M)

```
  // Get feasible direction
1 $\delta\bar{z} = M^{-1} g(y_k)$
  // Solve condensed QP
2 $b = -\tilde{Z}^\top \nabla_y \mathcal{L}(y^k, \lambda^k) + \tilde{Z}^\top \tilde{H} \begin{bmatrix} \delta\bar{z} \\ \mathbb{0} \end{bmatrix}$
3 $\delta w = (\tilde{Z}^\top \tilde{H} \tilde{Z})^{-1} b$
  // Expand to full variable space
4 $\delta z = -\delta\bar{z} - M^{-1} g_w(y^k, \lambda^k) \delta w$
5 $\delta\lambda = -\begin{bmatrix} M^{-\top} & \mathbb{0} \end{bmatrix} (\nabla_y \mathcal{L}(y^k, \lambda^k) + \tilde{H} \delta y)$
  // Update variables
6 $y^{k+1} = y^k + (\delta z^\top, \delta w^\top)^\top$
7 $\lambda^{k+1} = \lambda^k + \delta\lambda$
```

3.2 INIS for Equality Constrained Optimization

The algorithm introduced in this section introduces an additional variable $D \in \mathbb{R}^{n_\mathrm{z} \times n_\mathrm{w}}$ called the sensitivity matrix. This matrix is defined implicitly by the equation

$$g_z(y) D - g_w(y) = 0. \tag{3.16}$$

We solve this equation by applying Newtons method with the approximation $M \approx g_z$ and the current iterates y_k of the SQP method, resulting in

$$\delta D = -M^{-1}(g_z(y^k)D^k - g_w(y^k)), \tag{3.17}$$

and the updates

$$D^{k+1} = D^k + \delta D. \tag{3.18}$$

With the sensitivity matrix we can approximate the Jacobian $g_w(y^k)$ via

$$MD^k \approx g_w(y^k). \tag{3.19}$$

Hence, the QP solved in each SQP iteration reads as

$$\min_{\delta y \in \mathbb{R}^{n_\mathrm{y}}} \quad \frac{1}{2}\delta y^\top \tilde{H} \delta y + \nabla_y \mathcal{L}(y^k, \lambda^k)\delta y \tag{3.20a}$$

$$\text{subject to} \quad M\delta z + MD^k \delta w + g(y^k) = 0. \tag{3.20b}$$

Applying the condensing procedure introduced for the inexact Newton method with $\tilde{Z}^\top := \begin{bmatrix} -D^\top & \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$ and $\delta \bar{w} = \mathbb{0}$ results in the condensed system

$$\tilde{Z}^\top \tilde{H} \tilde{Z}\, \delta w = -\tilde{Z}^\top \nabla_y \mathcal{L}(y^k, \lambda^k) - \tilde{Z}^\top \tilde{H} \begin{bmatrix} -M^{-1} g(y^k) \\ \mathbb{0} \end{bmatrix}. \tag{3.21}$$

and the expansion step

$$\delta z = -M^{-1} g(y^k) - D^k \delta w \tag{3.22a}$$

$$\delta \lambda = -[M^{-\top} \quad \mathbb{0}](\nabla_y \mathcal{L}(y^k, \lambda^k) + \tilde{H}\delta y). \tag{3.22b}$$

Subsequently, we update the sensitivity matrix via Equations (3.17) and (3.18). These steps lead us to Algorithm 2.

Algorithmus 2 : `INIS`(y^k, λ^k, D^k, M)

```
   // Get feasible direction
1  $\delta \bar{z} = M^{-1} g(y^k)$
   // Solve condensed QP
2  $b = -Z^\top \nabla_y \mathcal{L}(y^k, \lambda^k) + Z^\top \tilde{H} \begin{bmatrix} \delta \bar{z} \\ \mathbb{0} \end{bmatrix}$
3  $\delta w = (Z^\top \tilde{H} Z)^{-1} b$
   // Expand to full varaiable space
4  $\delta z = \delta \bar{z} - D^k \delta w$
5  $\delta \lambda = -\begin{bmatrix} M^{-\top} & \mathbb{0} \end{bmatrix} (\nabla_y \mathcal{L}(y^k, \lambda^k) + \tilde{H} \delta y)$
   // Update iterates
6  $y^{k+1} = y^k + (\delta z^\top, \delta w^\top)^\top$
7  $\lambda^{k+1} = \lambda^k + \delta \lambda$
   // Update sensitivities
8  $\delta D = -M^{-1}(g_z(y^k) D^k - g_w(y^k))$
9  $D^{k+1} = D^k + \delta D$
```

Remark 1. Because of the condensing procedure and the introduced approximation $M \approx g_z$ the INIS algorithm is best suited for problems where $n_\mathrm{z} \gg n_\mathrm{w}$. For small n_w step 3 of the Algorithm 2 is computational cheap and the costs for solving steps 5 and 7 are reduced due to the approximation M which should be a matrix that is computational cheap to invert.

3.2.1 Local contraction

In order to analyze the local convergence properties of the INIS method we state the augmented FONC

$$F_{\mathrm{IS}}(y,\lambda,D) = \begin{bmatrix} \nabla_y \mathcal{L}(y,\lambda) \\ g(y) \\ \mathrm{vec}(g_z(y)D - g_w(y)) \end{bmatrix} = 0, \tag{3.23}$$

where $\mathrm{vec}(\cdot)$ vectorizes a given matrix in column major order. Thus $F_{\mathrm{IS}}(y,\lambda,D) \in \mathbb{R}^{n_{\mathrm{IS}}}$, with $n_{\mathrm{IS}} = 2n_{\mathrm{z}} + n_{\mathrm{z}}n_{\mathrm{w}} + n_{\mathrm{w}}$. Since they contain the FONC conditions of the original formulation (3.1) a solution of the augmented FONC is also a solution to the original conditions.

Proposition 1. A regular KKT point (y^*,λ^*,D^*) of the augmented conditions (3.23) corresponds to a regular KKT point (y^*,λ^*) of the NLP (3.1).

Proof. The KKT conditions of (3.1) are

$$\begin{bmatrix} \nabla_y \mathcal{L}(y,\lambda) \\ g(y) \end{bmatrix} = 0 \tag{3.24}$$

which are equal to the first two equations of $F_{\mathrm{IS}}(y,\lambda,D) = 0$. □

Applying Newton's method to (3.23) results in the system

$$\underbrace{\begin{bmatrix} \nabla_y^2 \mathcal{L}(y^k,\lambda^k) & \begin{pmatrix} g_z(y^k)^\top \\ g_w(y^k)^\top \end{pmatrix} & \mathbb{0} \\ \begin{pmatrix} g_z(y^k) & g_w(y^k) \end{pmatrix} & \mathbb{0} & \mathbb{0} \\ \frac{\partial}{\partial y}\mathrm{vec}(g_z(y^k)D^k - g_w(y^k)) & \mathbb{0} & \mathbb{1}_{n_{\mathrm{w}}} \otimes g_z(y^k) \end{bmatrix}}_{=:J_{\mathrm{IS}}(y^k,\lambda^k,D^k)} \begin{bmatrix} \delta y \\ \delta\lambda \\ \mathrm{vec}(\delta D) \end{bmatrix} = -F_{\mathrm{IS}}(y^k,\lambda^k,D^k) \tag{3.25}$$

where $\otimes$ denotes the Kronecker product of matrices. We approximate $J_{\mathrm{IS}}(y^k, \lambda^k, D^k)$ now be by the matrix

$$J_{\mathrm{INIS}}(y^k, \lambda^k, D^k) := \begin{bmatrix} \tilde{H} & \begin{pmatrix} M^\top \\ D^{k\top} M^\top \end{pmatrix} & \mathbb{0} \\ M \quad MD^k & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes M \end{bmatrix} \tag{3.26}$$

where we again used the approximations $\tilde{H}$, M and MD^k introduced in the previous sections. We assume these matrices to be such that $J_{\mathrm{INIS}}(\cdot)$ is continuously differentiable and invertible.

Theorem 5. For the augmented linear system (3.23) on the NLP in (3.1), the eigenspectrum of the INIS-type iteration matrix at the solution (y^*, λ^*, D^*) reads as

$$\sigma(J_{\mathrm{INIS}}^{-1} J_{\mathrm{IS}} - \mathbb{1}_{n_{\mathrm{IS}}}) = \sigma(M^{-1} g_z - \mathbb{1}_{n_\mathrm{z}}) \cup \sigma(\tilde{H}_Z^{-1} H_Z - \mathbb{1}_{n_\mathrm{w}}) \tag{3.27}$$

where $Z \in \mathbb{R}^{n_y \times n_\mathrm{w}}$ denotes a basis for the null space of the Jacobian $g_y(y)$, such that the reduced Hessians $H_Z := Z^\top H Z$ and $\tilde{H}_Z := Z^\top \tilde{H} Z$ are defined, with the exact Hessian $H := \nabla_y^2 \mathcal{L}(y, \lambda)$ and an approximation $\tilde{H} \approx H$.
More specifically, the iteration matrix has the n_w eigenvalues of the matrix $\tilde{H}_Z^{-1} H_Z - \mathbb{1}_{n_\mathrm{w}}$ and the n_z eigenvalues of $M^{-1} g_z - \mathbb{1}_{n_\mathrm{z}}$ with an algebraic multiplicity of $(2 + n_\mathrm{w})$.

Proof. The proof is based on the observation that the eigenvalues μ of the iteration matrix

$$J_{\mathrm{INIS}}^{-1} J_{\mathrm{IS}} - \mathbb{1}_{n_{\mathrm{IS}}} \tag{3.28}$$

are the zeros of

$$\det(J_{\mathrm{INIS}}^{-1} J_{\mathrm{IS}} - \mathbb{1}_{n_{\mathrm{IS}}} - \mu \mathbb{1}_{n_{\mathrm{IS}}}) = \det(J_{\mathrm{INIS}}^{-1} J_{\mathrm{IS}} - (\mu + 1) \mathbb{1}_{n_{\mathrm{IS}}}) = 0. \tag{3.29}$$

This matrix can be rewritten as product of block matrices where properties of the determinant can be used to determine the eigenvalues and their algebraic multiplicity. A detailed proof can be found in [9]. □

Based on this Theorem we can formulate a dependence of the contraction rate of the forward problem and the INIS algorithm.

Corollary 1 (Local INIS-type contraction)**.** The local rate of convergence for the INIS-type optimization algorithm is defined by

$$\kappa^*_{\mathrm{INIS}} = \rho(J^{-1}_{\mathrm{INIS}} J_{\mathrm{IS}} - \mathbb{1}_{n_{\mathrm{IS}}}) = \max\left(\kappa^*_F, \rho\left(\tilde{H}_Z^{-1} H_Z - \mathbb{1}_{n_{\mathrm{w}}}\right)\right) \tag{3.30}$$

where $\kappa^*_F = \rho\left(M^{-1} g_z - \mathbb{1}_{n_z}\right)$ is defined as the local contraction rate of the forward problem (3.5).

Proof. Follows directly by Theorem 5. □

From this corollary, it follows that local contraction of the forward problem is a necessary condition for local contraction of the INIS-type algorithm. If an exact Hessian is used in the INIS method the local contraction of the forward problem is even a sufficient condition. More precisely, if the Hessian approximation is good enough, i.e. $\rho\left(\tilde{H}_Z^{-1} H_Z - \mathbb{1}_{n_{\mathrm{w}}}\right) \leq \kappa^*_F$ we have $\kappa^*_{\mathrm{INIS}} = \kappa^*_F$. This connection between the contraction properties of the forward problem and the INIS method is a special property that makes the INIS method stand out fundamentally from the family of IN methods.

That the contraction of the forward problem is neither sufficient or necessary for convergence of the inexact Newton method in Algorithm 1 was discussed in [11] where also an example is stated to further support this statement.

4 Multi-Grid for Simulation of Partial Differential Equations

Partial differential equations are often used to describe physical processes like fluid dynamics, electricity, magnetism, heat, diffusion, etc., which are important in many fields of physics and engineering sciences. Therefore, the numerical solution of PDEs is an important task. In this chapter, we want to introduce the multi-grid method [12, 13] which is particularly well suited for solving very fine discretized PDEs.

Definition 8 (Partial Differential Equation (PDE))**.** Let $\Omega \subset \mathbb{R}^n$ be an open domain and $z \in C^k(\Omega)$ than a partial differential equation is a mapping

$$G : \Omega \times \mathbb{R}^n \times \mathbb{R}^{n^2} \times \cdots \times \mathbb{R}^{n^k} \to \mathbb{R}, \tag{4.1}$$

that defines a relation between the partial derivatives of z via the equation

$$G(t, z(t), Dz(t), D^2 z(t), \ldots, D^k z(t)) = 0, \tag{4.2}$$

for all $z \in \Omega$.

Functions that satisfy this equation are called solution of the partial differential equation. Generally, PDEs have got infinitely many solutions. To single out relevant solutions, additional conditions are imposed in form of boundary value or initial value conditions.

Definition 9 (Boundray Value Problem)**.** Let $\Omega \subset \mathbb{R}^n$ be an open domain and $z \in C^k(\bar{\Omega})$. A boundray value problem is a PDE $G(z) = 0$ together with boundray conditions that are defined via an equation

$$H(t, z(t), Dz(t), D^2z(t), \dots, D^{k-1}z(t)) = 0, \tag{4.3}$$

for all $t \in \partial\Omega$ with a mapping

$$H : \partial\Omega \times \mathbb{R}^n \times \mathbb{R}^{n^2} \times \cdots \times \mathbb{R}^{n^{k-1}} \to \mathbb{R}. \tag{4.4}$$

4.1 Numerical Solution of Poisson Equation

Let us regard the example of the Poisson equation which we restrict for simplicity to the 2-dimensional square $\Omega = (0,1)^2$, although Ω could be a more general domain or even of higher dimension. The following PDE will be part of our optimization problem in chapter 5 and is well suited to explain the basic principle of the MG method. Let us start with the simple Dirichlet boundary value problem

$$\begin{aligned} -\Delta z &= f \quad t \in \Omega, \\ z &= 0 \quad t \in \partial\Omega, \end{aligned} \tag{4.5}$$

with $f : \Omega \to \mathbb{R}$. For the numerical solution the negative Laplace operator $-\Delta$ will be discretized by finite differences [13]. This discretization is characterised by the choice of the grid size h and the difference scheme. For an integer $J \geq 1$ we set $h = 1/J$ and define the grid points $t_{i,j} = (ih, jh)$ for $0 \leq i, j \leq J$. The partial derivatives of the negative Laplace operator will be discretized by central difference quotients

$$\frac{\partial^2 z(t_i, t_j)}{\partial t_1 \partial t_1} \approx \partial^+_{t_1}\partial^-_{t_1} z_{i,j} := \frac{z_{i-1,j} - 2z_{i,j} + z_{i+1,j}}{h^2}. \tag{4.6}$$

where $z_{i,j}$ are approximations of the unknowns $z(t_{i,j})$.

These approximations result in the discretized Poisson problem

$$\begin{aligned} -\partial_{t_1}^{+}\partial_{t_1}^{-}z_{i,j} - \partial_{t_2}^{+}\partial_{t_2}^{-}z_{i,j} &= f(t_{i,j}) \qquad && \text{for } 1 \leq i,j \leq J-1, \\ z_{0,j} = z_{J,j} = z_{i,0} = z_{i,J} &= 0 && \text{for } 0 \leq i,j \leq J. \end{aligned} \tag{4.7}$$

Rewriting the left hand side of the first equation with the definition of the central difference quotients we get

$$-\partial_{t_1}^{+}\partial_{t_1}^{-}z_{i,j} - \partial_{t_2}^{+}\partial_{t_2}^{-}z_{i,j} = -h^{-2}(z_{i,j-1} + z_{i-1,j} - 4z_{i,j} + z_{i+1,j} + z_{i,j+1}) \tag{4.8}$$

The right hand side is often referred to the 5-point stencil which helps to get a visual intuition of the discretized laplace operator, see Figure 1.

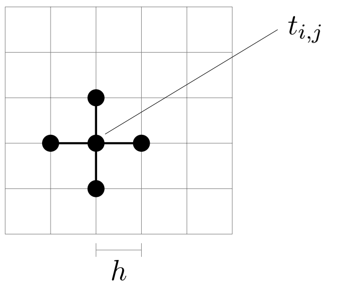

Figure 1: 5-point-stencil on uniform grid with gridsize h.

The goal is to iteratively determine the coefficients $(z_{i,j} : i,j = 0,\dots,J)$ with a linear system of equations resulting from the descretized Poisson problem. In order to simplify the equations we introduce the *lexicographic enumeration* for the interior grid points by identifying

$$(i,j) \equiv i + (j-1)(J-1) = m, \tag{4.9}$$

for $i,j = 1,\dots,J-1$ and $m = 1,\dots,N$ with $N = (J-1)^2$. An illustration is found in Figure 2.

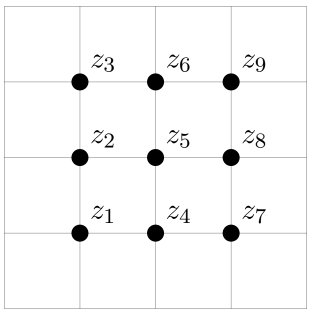

Figure 2: lexicographic enumeration of the interior grid points on uniform grid with gridsize 1/4.

.

Because of the vanishing coefficients $z_{0,j} = z_{J,j} = z_{i,0} = z_{i,J} = 0$ for $i, j = 0, \dots, J$ the linear system of equations can be reduced and written as

$$\underbrace{h^{-2}\begin{bmatrix} X & -\mathbb{1} & & \\ -\mathbb{1} & \ddots & \ddots & \\ & \ddots & \ddots & -\mathbb{1} \\ & & -\mathbb{1} & X \end{bmatrix}}_{=:A} \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}}_{=:Z} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{=:F}, \tag{4.10}$$

with the matrix $X \in \mathbb{R}^{(J-1)\times(J-1)}$ defined by

$$X = \begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix}, \tag{4.11}$$

and $f_i = f(t_i)$ for $i = 1, \dots, N$. This linear system of equations

$$AZ = F \tag{4.12}$$

can be solved iteratively with methods like the Richardson, Gauss-Seidel or Jacobi method [14]. However, the drawback of suchlike iterative solvers is a decreasing convergence rate for finer grids. Therefore, a single iteration does not only get more expensive to compute due to the larger linear system, but also more iterations steps are needed to converge to the solution. In the next sections, we will introduce the MG method which does not suffer from this problem but instead has a convergence rate independent of the gridsize and therefore is well suited to solve large systems resulting from a fine grid. More details to Finite Difference Methods can be found in [13].

4.2 Multi-Grid (MG) Method

The multi-grid method is an algorithm for linear systems resulting from the discretization of PDEs. In contrast to basic iterative solvers for linear systems, the MG method not only uses the linear system itself but also linear systems of lower dimension that structurally depend on the system that actual has to be solved, which can be seen as discretization on a coarser grid.
In the following, we summarized the basic concept of the multi-grid method to give a first intuition.

4.2.1 Overview: Basic Concept of MG

We will limit ourself to the geometric approach of the MG method, in this concept the structural dependence of the linear systems will be established with a hierarchy of grids.
The MG method starts by calculating a *fine grid approximation* of the solution on the initial/finest grid. The basic idea of the MG method is to improve this approximation through a correction term which is obtained as solution of a *defect problem* on a coarser grid. This problem is formulated with the help of the fine grid approximation.

In order to formulate the defect problem on the coarser grid we have to transfer the functions between the different refinement levels called *gridlevels*. The transfer from a fine grid onto a coarse grid is called *restriction* and the other way around *prolongation*.
It is important to note that the correction of the fine grid approximation with the solution of the defect problem is not sufficient for convergence of the MG method. The key component is the so called *smoother*. Its purpose is to reduce certain parts of the error resulting in a "smoother" error. The error of a current approximation can be divided into low- and high-frequency errors. The goal of the coarse grid correction is to reduce the low-frequency error and the smoother reduces the high-frequency error, in combination they lead to the MG method. Typically basic iterative methods are used as smoother. Even tough these methods are not suited to solve the system directly as mentioned above, they do quickly remove high frequency errors.
These series of steps are commonly referred to one MG-Cycle or one iteration of the MG method.

4.2.2 Grid Hierarchy

Recalling the Poisson test problem

$$\begin{aligned} -\Delta z &= f \quad t \in \Omega, \\ z &= 0 \quad t \in \partial\Omega, \end{aligned} \tag{4.13}$$

we will know introduce a hierarchy of grids starting with the largest grid size $h_0 = 1/2$ and corresponding grid $\Omega_0 = \{1/2\}$. A hierarchy of refined grids is know established by successively halving the grid sizes

$$h_l = \frac{1}{2} h_{l-1}, \tag{4.14}$$

resulting in the sequence

$$h_0 > h_1 > \ldots > h_l > \ldots > h_L \quad \text{with } h_l = 2^{-l-1} \tag{4.15}$$

for given $L > 0$. The index l is called *gridlevel* with corresponding interior grid $\Omega_l = \{(ih_l, jh_l) : 1 \leq i, j \leq J_l\}$ with $J_l = h_l^{-1} - 1$. This hierarchy is illustrated in Figure 3 with the lexicographic enumeration introduced in Equation (4.9).

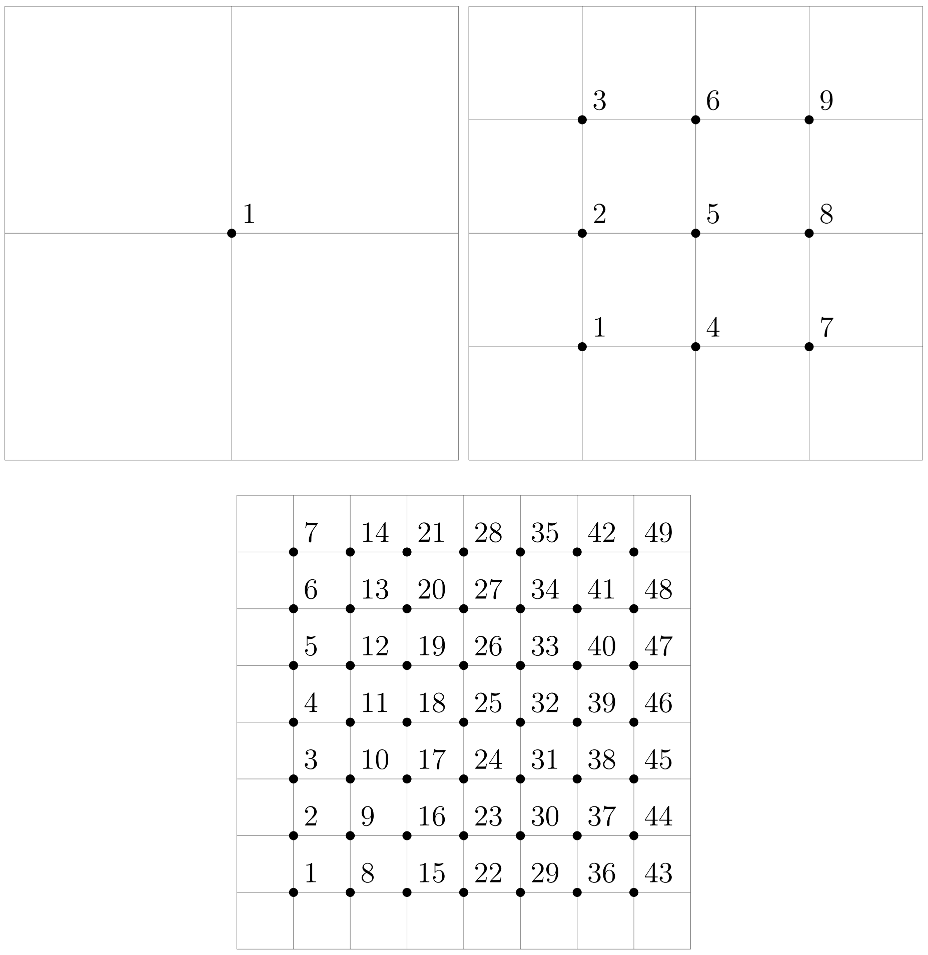

Figure 3: Grid hierarchy for gridlevels $l = 0, 1$ and 2 with interior grid point enumeration

From now on, we will formulate the discretized Poisson problem for a given gridlevel

$l \in \mathbb{N}$, i.e. we write Equation (4.12) as

$$A_l Z_l = F_l. \tag{4.16}$$

Based on this linear system of equations, we investigate the smoothing effect of the smoother.

4.2.3 Richardson Method as Smoother for the MG Method

Different iterative methods has been proved to be effective smoothers for the Poisson equation, such as the iterations Gauß-Seidel, damped Jacobi, alternating direction implicit iteration (ADI) and the Richardson method [12]. For simplicity, we restrict ourselves in the following to the Richardson iteration as introduced in [13],

$$Z_l^k = Z_l^{k-1} - \omega(A_l Z_l^{k-1} - F_l). \tag{4.17}$$

For a given number $\nu \in \mathbb{N}$ and an inital guess Z_l^0, Algorithm 3 performs $\nu \in \mathbb{N}$ Richardson iterations.

Algorithmus 3 : `richardson`$(A_l, F_l, Z_l^0, \omega, \nu)$

```
1 for $k \leftarrow 1$ to $\nu$ do
2     $Z_l^k = Z_l^{k-1} - \omega(A_l Z_l^{k-1} - F_l)$
3 end
4 return $Z_l^\nu$
```

Condition Number of Discretization Matrix

For convergence analysis we want to investigate the eigenvectors of the discretization matrix A_l, [12, 15].

We note that the eigenvectors of A_l are

$$\mathrm{v}_l^{i,j} = v_l^i \otimes v_l^j \tag{4.18}$$

for $i, j = 1, \ldots, J_l$ with

$$v_l^i = [\sin(i\pi h_l), \sin(2i\pi h_l), \ldots, \sin(J_l i\pi h_l)]^T \tag{4.19}$$

for $i = 1, \ldots, J_l$. The corresponding eigenvalues are

$$\xi_l^{i,j} = 4h^{-2}(\sin(i\pi h_l/2)^2 + \sin(j\pi h_l/2)^2). \tag{4.20}$$

$i, j = 1, \ldots, J_l$. With the eigenvalues

$$\begin{aligned} \xi_{\max} &= 8h^{-2}\sin((h_l^{-1}-1)\pi h_l/2)^2, \\ \xi_{\min} &= 8h^{-2}\sin(\pi h_l/2)^2, \end{aligned} \tag{4.21}$$

we can obtain the condition number approximately for $h_l \ll 1$ by

$$\begin{aligned} \kappa_{\text{cond}} &= \frac{\xi_{\max}}{\xi_{\min}} \\ &= \frac{\sin((h_l^{-1}-1)\pi h_l/2)^2}{\sin(\pi h_l/2)^2} \\ &\approx \frac{\sin(\pi/2)^2}{\sin(\pi h_l/2)^2} \\ &= \frac{1}{\sin(\pi h_l/2)^2}. \end{aligned} \tag{4.22}$$

Taylors formula than yields

$$\kappa_{\text{cond}} \approx \frac{1}{(0 + \pi h_l/2 + 0 + \mathcal{O}(h_l^3))^2} \approx h_l^{-2}. \tag{4.23}$$

Hence the condition number is large for $h_l \ll 1$ and classical iterative solvers typically converge slowly [13]. An illustration of $\xi_{\max}$ and $\xi_{\min}$ can be found in Figure 4.

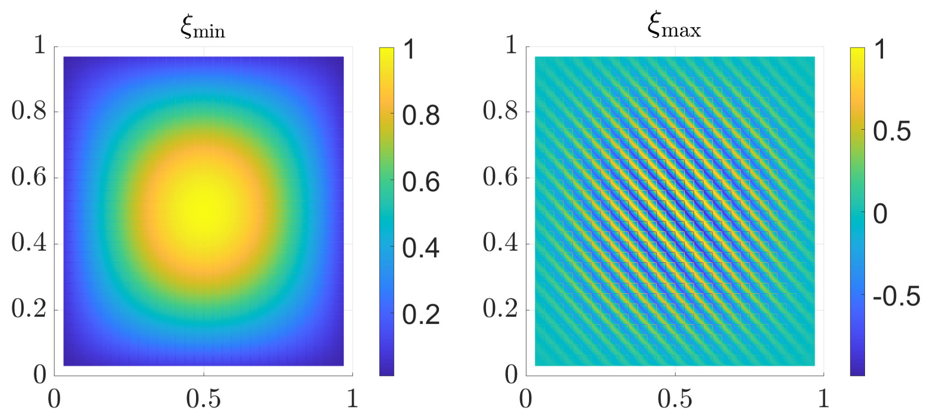

Figure 4: Illustration of the low and high frequency eigenvectors $\xi_{\min}$ and $\xi_{\max}$ of A_l.

Smoothing Effect of the Richardson Method

For the solution Z_l^* we analyse the error $e_l^k = Z_l^k - Z_l^*$,

$$\begin{aligned} e_l^k &= Z_l^k - Z_l^* \\ &= Z_l^{k-1} - \omega(A_l Z_l^{k-1} - F_l) - Z_l^* \\ &= e_l^{k-1} - \omega(A_l Z_l^{k-1} - A_l Z_l^*) \\ &= (\mathbb{1}_{J_l^2} - \omega A_l) e_l^{k-1}. \end{aligned} \tag{4.24}$$

Thus, the error converges to zero if

$$|\mathbb{1}_{J_l^2} - \omega \xi_l^{i,j}| < 1 \tag{4.25}$$

for $i, j = 1, \ldots, J_l$. This is guaranteed if we chose $\omega \in (0, 2/\xi_{\max})$, e.g. $\omega = 1/\xi_{\max}$. Using the eigenvectors of A_l we have

$$e_l^0 = \sum_{i=1}^{J_l} \sum_{j=1}^{J_l} \eta^{i,j} \mathrm{v}_l^{i,j} \tag{4.26}$$

for some coefficients $\eta^{i,j} \in \mathbb{R}$. ν steps of the Richardson method then yield

$$e_l^{\nu} = \sum_{i=1}^{J_l} \sum_{j=1}^{J_l} \theta^{i,j} \mathrm{v}_l^{i,j} \tag{4.27}$$

with

$$\begin{aligned} \theta^{i,j} &= \eta^{i,j}(1 - \omega \xi_l^{i,j})^K \\ &= \eta^{i,j}(1 - \frac{\xi_l^{i,j}}{\xi_{\max}})^K. \end{aligned} \tag{4.28}$$

Consequently the high frequency parts of the error vanish much faster than low frequency parts. An illustration of this observation is provided in Figure 5. So we have seen convergence of the Richardson Iteration is only lacking in the low frequencies.

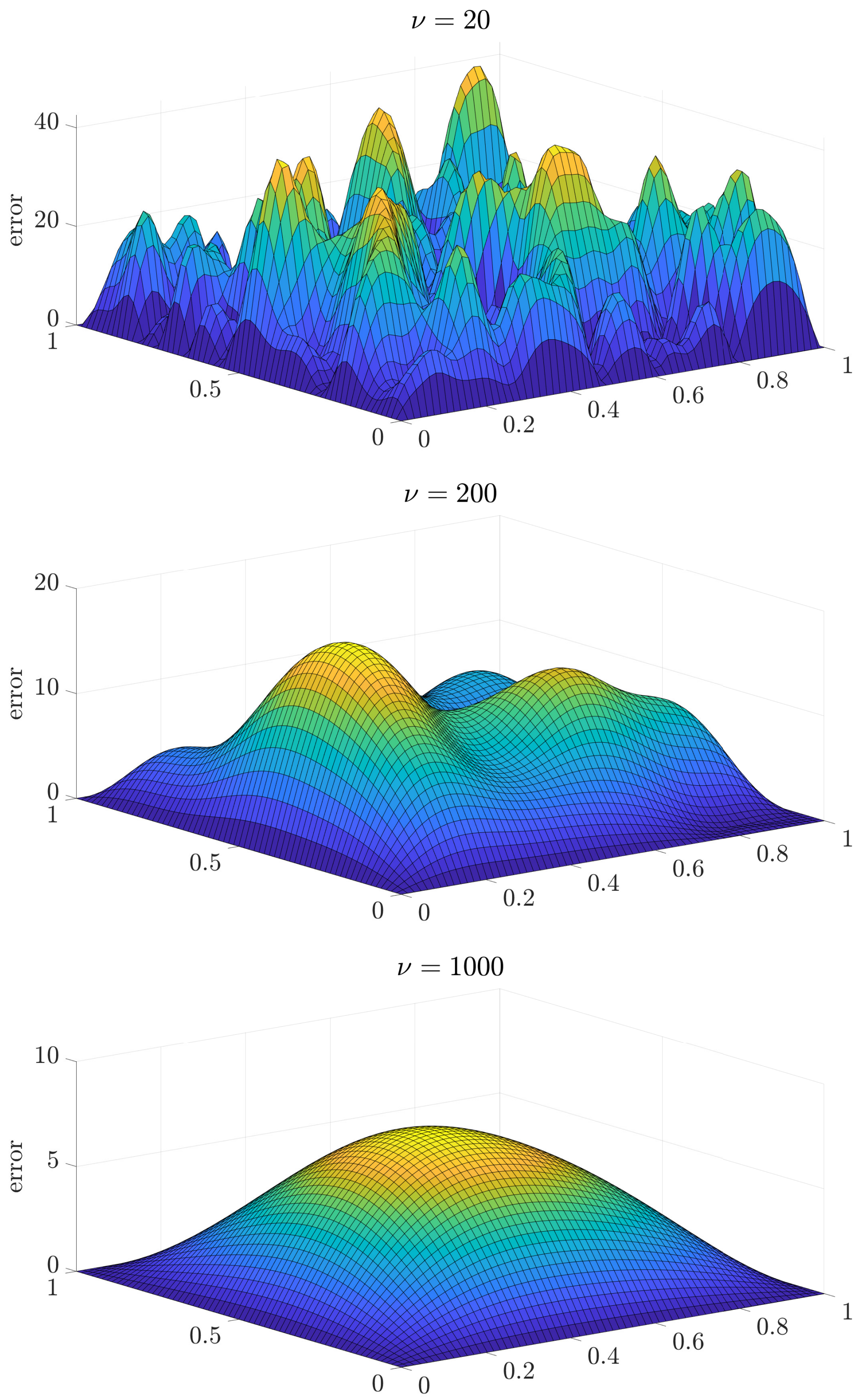

Figure 5: Error $e_l^\nu = |Z_l^\nu - Z_l^*|$ after $\nu = 20, 200, 1000$ iterations with the Richardson method on gridlevel $l = 6$ to show the smoothing effect.

Linearity of the Richardson Method

We want to close the section about the Richardson method with a linearity estimate, which we use later in this chapter to prove a similar result for the MG method.

Lemma 1 (Linearity of the Richardson Method)**.** For $\nu \in \mathbb{N}$ there exist matrices S_l^ν, T_l^ν such that ν Richardson iterations can compactly be written as

$$Z_l^\nu = S_l^\nu Z_l^0 + T_l^\nu F_l \tag{4.29}$$

for all $Z_l^0, F_l \in \mathbb{R}^{J_l^2}$. With

$$\begin{aligned} S_l^\nu &= \mathbb{1}_{J_l^2} - \omega A_l, \\ T_l^\nu &= \mathbb{1}_{J_l^2}\omega \end{aligned} \tag{4.30}$$

for $\nu = 1$ and

$$\begin{aligned} S_l^\nu &= (\mathbb{1}_{J_l^2} - \omega A_l) S_l^{\nu-1}, \\ T_l^\nu &= ((\mathbb{1}_{J_l^2} - \omega A_l) T_l^{\nu-1} - \omega \mathbb{1}_{J_l^2}) \end{aligned} \tag{4.31}$$

for $\nu > 1$.

Proof. The proof is performed by induction.
For $\nu = 1$ we can rewrite Equation (4.17) as

$$Z_l^\nu = (\mathbb{1}_{J_l^2} - \omega A_l) Z_l^0 - \omega F_l, \tag{4.32}$$

hence Equation (4.29) holds.
Lets assume Equation (4.29) holds for $\nu - 1$, hence, we can rewrite the Richardson iteration as

$$\begin{aligned} Z_l^\nu &= (\mathbb{1}_{J_l^2} - \omega A_l) Z_l^{\nu-1} - \omega F_l \\ &= (\mathbb{1}_{J_l^2} - \omega A_l)(S_l^{\nu-1} Z_l^0 + T_l^{\nu-1} F_l) - \omega F_l \\ &= (\mathbb{1}_{J_l^2} - \omega A_l) S_l^{\nu-1} Z_l^0 + ((\mathbb{1}_{J_l^2} - \omega A_l) T_l^{\nu-1} - \omega \mathbb{1}_{J_l^2}) F_l \end{aligned} \tag{4.33}$$

which proves the lemma. $\square$

4.2.4 Two-Grid Method

Based on the observation of the previous subsection, our next step is to introduce a second iteration which manly focuses on reducing the smooth part of the error in order to get a method efficiently reducing all parts of the error.
Given a grid Ω_l with grid size h_l and an approximation Z_l of the solution $Z_l^* = A_l^{-1}F_l$. As we have seen in the previous section, we can smooth this approximation with only a few Richardson iterations resulting in Z_l^ν where $\nu \in \mathbb{N}$ is the number of smoothing iterations. More precisely the error $e_l^\nu = Z_l^\nu - Z_l^*$ is smoother than the old error $e_l = Z_l - Z_l^*$. With the residuum $r_l = A_l Z_l^\nu - F_l$ the error e_l^ν is the solution of the *defect problem*

$$A_l d_l = r_l, \tag{4.34}$$

since

$$A_l e_l^\nu = A_l Z_l^\nu - A_l Z_l^* = A_l Z_l^\nu - F_l = r_l. \tag{4.35}$$

Although we have formulated a new linear system of equations with the same complexity as Equation (4.12), e_l^ν can be approximated on a coarse grid better than Z_l^*, because it is a smoothed function.
To formulate the defect problem on a coarse grid we define a *restriction operator*

$$\begin{aligned} R_l \colon \mathbb{R}^{J_l^2} &\to \mathbb{R}^{J_{l-1}^2} \\ r_l &\mapsto R_l\, r_l. \end{aligned} \tag{4.36}$$

There are different variants how to restrict a grid function to a coarser grid [12], we used a version introduced in [13]. R_l is a sparse matrix and its sparsity pattern is visualized in Figure 6 with repeating entries

$$\begin{bmatrix} 1/2 & 1/2 & 0 & \dots & 0 & 1/2 & 1 & 1/2 & 0 & \dots & 1/2 & 1/2 \end{bmatrix} \tag{4.37}$$

in each row.

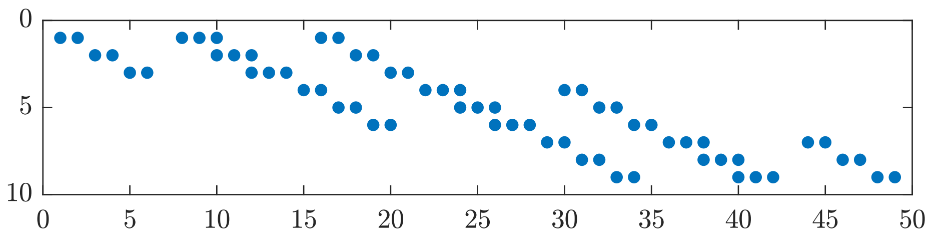

Figure 6: Sparsity pattern for restriction matrix $R_3 \colon \mathbb{R}^{49} \to \mathbb{R}^9$.

The effect of the restriction matrix R_l on the coefficient vector r_l is illustrated in Figure 7.

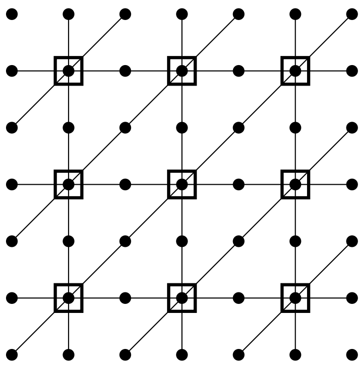

Figure 7: Restriction and prolongation for gridlevel $l = 3$.

The restricted defect problem is called *coarse grid equation* and reads as

$$A_{l-1}d_{l-1} = r_{l-1} = R_l r_l, \tag{4.38}$$

where A_{l-1} is the matrix from the discretized Poisson problem on gridlevel $l-1$. As solution we obtain $d_{l-1} = A_{l-1}^{-1} r_{l-1}$ which we expect to be an approximation of the error e_l^ν. Since d_{l-1} is only defined on the coarse grid Ω_{l-1}, we have to interpolate this solution back to the fine grid. Therefore we define the *prolongation operator*

$$\begin{aligned} P_l \colon \mathbb{R}^{J_{l-1}^2} &\to \mathbb{R}^{J_l^2} \\ d_{l-1} &\mapsto P_l \, d_{l-1} = R_l^\top d_{l-1}. \end{aligned} \tag{4.39}$$

This relates to a linear interpolation illustrated in Figure 7. Now, we can update our smoothed approximation Z_l^ν introducing the compact formula

$$Z_l^\nu \mapsto Z_l^\nu - P_l A_{l-1}^{-1} R_l (A_l Z_l^\nu - F_l), \tag{4.40}$$

called *coarse grid correction* . The combination of these steps is called *two-grid iteration* summarized in Algorithm 4.

Algorithmus 4 : `two_grid`$(A_l, F_l, Z_l^0, \omega, \nu)$

```
1 $Z_l^\nu =$ richardson$(A_l, F_l, Z_l^0, \omega, \nu)$            // smoothing inital guess
2 $r_l = A_l Z_l^\nu - F_l$                                  // calculation of the residuum
3 $r_{l-1} = R_l r_l$                                        // restriction of the residuum
4 $d_{l-1} = A_{l-1}^{-1} r_{l-1}$              // exact solution of the coarse-grid equation
5 $Z_l = Z_l^\nu - R_l^\top d_{l-1}$                                         // correction step
6 return $Z_l$
```

A common modification to the two-grid method is to apply *pre-smoothing* step before the coarse-grid correction and a *post-smoothing* step afterwards [12].

4.2.5 Multi-Grid Method

For a very fine discretization of the Poisson problem (4.5) or other multi-dimensional boundary value problems the computation of the exact solution $d_{l-1} = A_{l-1}^{-1} r_{l-1}$ is expensive and unnecessary. Since d_{l-1} is an approximation of the exact correction d_l there is no need to solve the coarse-grid equation $A_{l-1} d_{l-1} = r_{l-1}$ exactly. Since the coarse-grid equation is of the same form as the original equation $A_l Z_l = F_l$, we can apply the two-grid method to approximate d_{l-1}. Repeating this idea recursively until a gridlevel $l_{\min} \geq 0$ is reached leads to a *multi-grid method*.

On the coarsest grid, the smallest system of equations has to be solved, in the case of $l_{\min} = 0$ this would only be a scalar equation. Hence, the exact solution can be computed with low computational cost. The multi-grid method can easily be described as a recursive algorithm.

A single multi-grid iteration is commonly referred to as *cycle*. The sequence of operations during one multi-grid cycle is illustrated in Figure 8 and due to the form it is also called *V-cycle*.

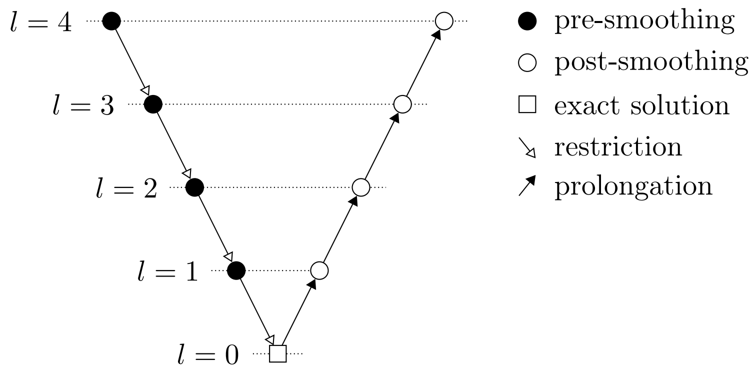

Figure 8: Graphical illustration of the recursive MG strategy.

An other common multi-grid cycle is the *W-cycle* illustrated in Figure 9. The W-cycle introduces additional computational cost compared to the V-cycle, but is more stable meaning that less smoothness requirements for the solution are needed [16]. The W-cycle can be seen as a generalization of the V-cycle shown in Algorithm 5. The V-cycle is obtained for $\eta = 0$ and the W-cycle for $\eta = 1$. For more details to MG methods see [12, 17, 18].

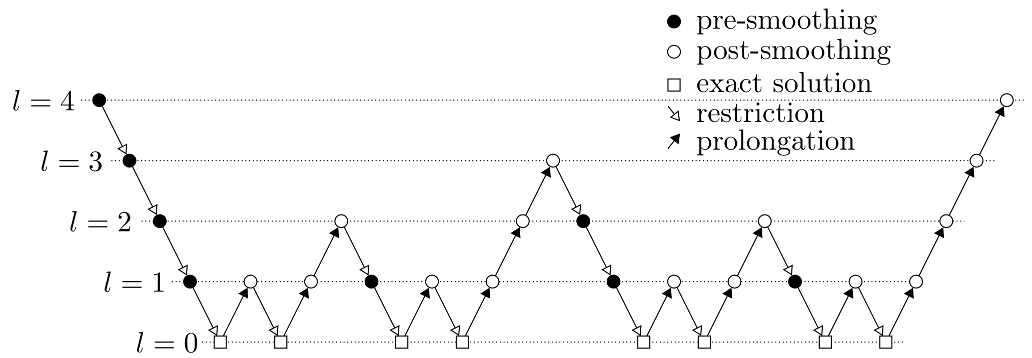

Figure 9: Graphical illustration of the operations during a single W-cycle.

Algorithmus 5 : `multi_grid`$(A_l, F_l, Z_l^0, \omega, \nu_{\text{pre}}, \nu_{\text{post}}, l_{\min}, l, \eta)$

```
1  if $l = l_{\min}$ then
2      $Z_l = A_l^{-1} F_l$
3  end
4  else
5      $Z_l =$ richardson$(A_l, F_l, Z_l^0, \omega, \nu_{\text{pre}})$          // pre-smoothing
6      $r_{l-1} = R_l(A_l Z_l - F_l)$                  // restriction of the residuum
7      $d_{l-1} = 0$                 // initial guess for coarse grid correction
       // recursion
8      for $0$ to $\eta$ do
9          $d_{l-1} =$ multi_grid$(A_{l-1}, r_{l-1}, d_{l-1}, \omega, \nu_{\text{pre}}, \nu_{\text{post}}, l_{\min}, l-1, \eta)$
10     end
11     $Z_l = Z_l - R_l^\top d_{l-1}$                              // correction step
12     $Z_l =$ richardson$(A_l, F_l, Z_l, \omega, \nu_{\text{post}})$          // post-smoothing
13 end
14 return $Z_l$
```

4.3 Linearity of the Multi-Grid Method

In order to combine the MG method with the INIS method and preserve the local convergence properties we have to prove that the MG method is linear in Z_l^0 and the right side F_l.
For a given gridlevel $l \in \mathbb{N}$ we want to show that there exist matrices $S_l^{\mathrm{MG}}, T_l^{\mathrm{MG}} \in \mathbb{R}^{J_l^2 \times J_l^2}$, such that

$$Z_l^{\mathrm{MG}} = S_l^{\mathrm{MG}} Z_l^0 + T_l^{\mathrm{MG}} F_l \tag{4.41}$$

holds. For simplicity we restrict ourselves to the V-cycle, i.e. $\eta = 1$, even though this estimate holds for $\eta \in \mathbb{N}$. We define the mapping

$$\begin{aligned} \varphi_l \colon \mathbb{R}^{J_l^2} \times \mathbb{R}^{J_l^2} &\to \mathbb{R}^{J_l^2} \\ (Z_l, F_l) &\mapsto \varphi_l(Z_l, F_l) \end{aligned} \tag{4.42}$$

as a single V-cycle with fixed number of pre- and post-smoothing steps ν_{pre} and ν_{post}.

Lemma 2 (Linearity of the V-cycle)**.** The mapping φ_l is linear in Z_l and F_l, i.e. for $l \geq 0$ there exist matrices $S_l^{\mathrm{MG}}, T_l^{\mathrm{MG}} \in \mathbb{R}^{J_l^2 \times J_l^2}$ such that

$$\varphi_l(Z_l, F_l) = S_l^{\mathrm{MG}} Z_l + T_l^{\mathrm{MG}} F_l \tag{4.43}$$

for all $Z_l, F_l \in \mathbb{R}^{J_l^2}$. For $l = 0$ these matrices are

$$\begin{aligned} S_l^{\mathrm{MG}} &= \mathbb{0}, \\ T_l^{\mathrm{MG}} &= A_l^{-1} \end{aligned} \tag{4.44}$$

and for $l > 0$ they are recursively defined as

$$\begin{aligned} S_l^{\mathrm{MG}} &= S_l^{\nu_{\mathrm{post}}} (S_l^{\nu_{\mathrm{pre}}} + R_l^T T_{l-1}^{\mathrm{MG}} R_l A_l S_l^{\nu_{\mathrm{pre}}}), \\ T_l^{\mathrm{MG}} &= S_l^{\nu_{\mathrm{post}}} (T_l^{\nu_{\mathrm{pre}}} + R_l^T (T_{l-1}^{\mathrm{MG}} R_l A_l T_l^{\nu_{\mathrm{pre}}} - T_{l-1}^{\mathrm{MG}} R_l)) + T_l^{\nu_{\mathrm{post}}}. \end{aligned} \tag{4.45}$$

Proof. The proof will be performed by induction.
For $l = 0$ we have

$$\varphi_l(Z_l, F_l) = A_l^{-1} F_l \tag{4.46}$$

which proves the base case and Equation (4.44).
Lets assume Equation (4.43) holds for $l-1$. With Lemma 1 the restriction of the residuum after pre-smoothing can compactly be written as

$$r_{l-1} = R_l A_l S_l^{\nu_{\text{pre}}} Z_l + (R_l A_l T_l^{\nu_{\text{pre}}} - R_l) F_l. \tag{4.47}$$

With the induction for $l-1$ the recursion step is given as

$$\begin{aligned}
\varphi_{l-1}(\mathbb{0}, r_{l-1}) &\overset{(4.47)}{=} \varphi_{l-1}(\mathbb{0},\, R_l A_l S_l^{\nu_{\text{pre}}} Z_l + (R_l A_l T_l^{\nu_{\text{pre}}} - R_l) F_l) \\
&\overset{(4.43)}{=} T_{l-1}^{\text{MG}}(R_l A_l S_l^{\nu_{\text{pre}}} Z_l + (R_l A_l T_l^{\nu_{\text{pre}}} - R_l) F_l) \\
&= T_{l-1}^{\text{MG}} R_l A_l S_l^{\nu_{\text{pre}}} Z_l + (T_{l-1}^{\text{MG}} R_l A_l T_l^{\nu_{\text{pre}}} - T_{l-1}^{\text{MG}} R_l) F_l
\end{aligned} \tag{4.48}$$

Applying the correction and post-smoothing step yields

$$\begin{aligned}
\varphi_l(z_l, f_l) &= S_l^{\nu_{\text{post}}}(S_l^{\nu_{\text{pre}}} Z_l + T_l^{\nu_{\text{pre}}} F_l + R_l^T \varphi_{l-1}(d_{l-1}, r_{l-1})) + T_l^{\nu_{\text{post}}} F_l \\
&\overset{(4.48)}{=} S_l^{\nu_{\text{post}}}[S_l^{\nu_{\text{pre}}} Z_l + T_l^{\nu_{\text{pre}}} F_l + R_l^T (T_{l-1}^{\text{MG}} R_l A_l S_l^{\nu_{\text{pre}}} Z_l \\
&\quad + (T_{l-1}^{\text{MG}} R_l A_l T_l^{\nu_{\text{pre}}} - T_{l-1}^{\text{MG}} R_l) F_l)] + T_l^{\nu_{\text{post}}} F_l \\
&= S_l^{\nu_{\text{post}}}[(S_l^{\nu_{\text{pre}}} + R_l^T (T_{l-1}^{\text{MG}} R_l A_l S_l^{\nu_{\text{pre}}})) Z_l \\
&\quad + (T_l^{\nu_{\text{pre}}} + R_l^T (T_{l-1}^{\text{MG}} R_l A_l T_l^{\nu_{\text{pre}}} - T_{l-1}^{\text{MG}} R_l)) F_l] + T_l^{\nu_{\text{post}}} F_l \\
&= S_l^{\nu_{\text{post}}}(S_l^{\nu_{\text{pre}}} + R_l^T T_{l-1}^{\text{MG}} R_l A_l S_l^{\nu_{\text{pre}}}) Z_l \\
&\quad + (S_l^{\nu_{\text{post}}}(T_l^{\nu_{\text{pre}}} + R_l^T (T_{l-1}^{\text{MG}} R_l A_l T_l^{\nu_{\text{pre}}} - T_{l-1}^{\text{MG}} R_l)) + T_l^{\nu_{\text{post}}}) F_l
\end{aligned} \tag{4.49}$$

which proves the theorem. □

5 INIS-Multi-Grid (INIS-MG) for Optimization of PDE

In this chapter we explain how the INIS method can be combined with the MG method. Therefore, we state an PDE constrained optimal control test problem which we discretize, resulting in an NLP where we can apply the INIS method. The MG method can then be used to solve linear systems resulting from the discretization of the PDE.

5.1 PDE Constrained Optimal Control Test Problem

We start by stating the optimization problem

$$\begin{aligned}
&\underset{z(\cdot),\, u(\cdot)}{\text{minimize}} && \frac{1-\alpha}{2}\int_{\Omega} \left\| z - f_{\text{ref}}^{\gamma} \right\|^2 \mathrm{d}t + \frac{\alpha}{2}\int_{\partial\Omega} \|u\|^2 \,\mathrm{d}s, && \text{(5.1a)}\\
&\text{subject to} && -\Delta z = \beta z^3 \quad t \in \Omega = (0,1)^2, && \text{(5.1b)}\\
& && u \in \mathcal{C}(\partial\Omega), && \text{(5.1c)}\\
& && u|_{\partial\Omega_i} = u_i \quad i = 1,\dots,4\,, && \text{(5.1d)}\\
& && u_i \in \mathscr{P}_5(\partial\Omega_i) \quad i = 1,\dots,4\,, && \text{(5.1e)}\\
& && z|_{\partial\Omega_i} = u_i \quad i = 1,\dots,4\,, && \text{(5.1f)}
\end{aligned}$$

where the states $z(\cdot)$ have to satisfy the nonlinear Poisson Equation (5.1b) with $\beta \in \mathbb{R}$. The constraints in Equation (5.1c) - (5.1e) require the controls $u(\cdot)$ to be a polynomial of degree 5 on each edge and continuous on the entire boundary $\partial\Omega$. The numeration of the boundary is illustrated in Figure 10.

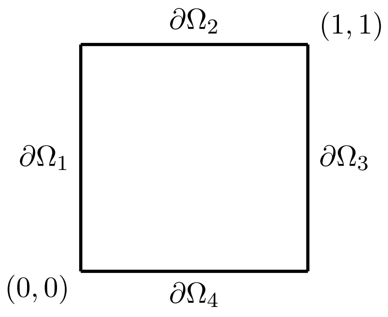

Figure 10: Domain Ω with numeration of its boundary.

The objective function penalizes the difference to the reference function

$$f_{\text{ref}}^{\gamma}(t) = \begin{cases} \gamma, & \text{for } t \in [0.2, 0.3]^2 \\ 0, & \text{otherwise.} \end{cases}$$

with $\gamma \in \mathbb{R}$, illustrated in Figure 11, and the absolute value of the controls. The contribution of the integrals to the objective function can be adjusted by the parameter $\alpha \in [0, 1]$.

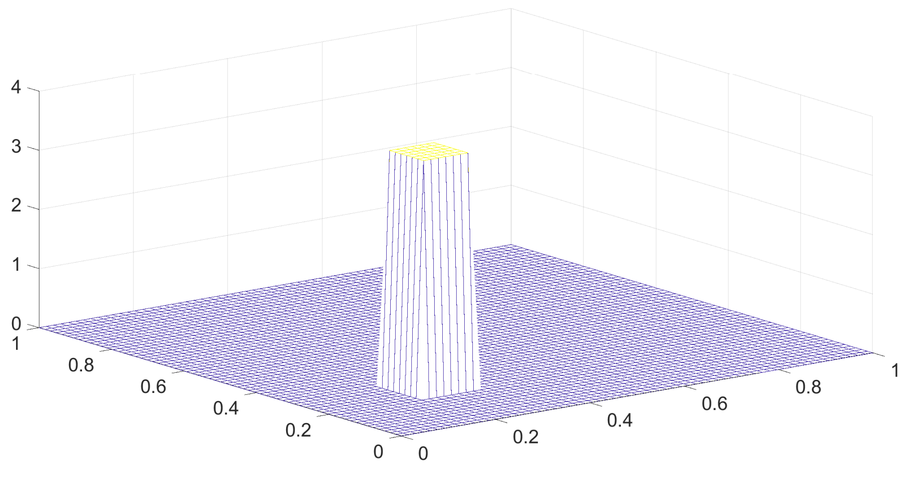

Figure 11: Reference function $f_{\text{ref}}^{\gamma}(\cdot)$ with $\gamma = 4$.

5.2 Discretizing the PDE Constrained Test Problem

In this chapter, we want to discretize the optimization problem (5.1) such that we approximate it by a finite-dimensional NLP. We will do so by using different techniques like Riemann sums for the objective function and finite differences for the PDE. Additionally, we can reduce the number of controls which is beneficial for the INIS algorithm as explained in Remark 1.

First, we partition the domain Ω with a uniform grid by choosing a stepsize $h = 1/J$ with $J = 2^{L+1}$ where $L \in \mathbb{N}$ corresponds to the finest gridlevel introduced in Subsection 4.2.2. We then define the gridpoints $t_{i.j} = (ih, jh)$ for $0 \leq i, j \leq J$ and the number of interior grid points in a single row by $I := J - 1$.
The discretization of the objective function with Riemann sums leads to the double sum

$$\frac{1-\alpha}{2} h^2 \sum_{i=1}^{I} \sum_{j=1}^{I} (z_{i,j} - f_{\text{ref}}^{\gamma}(t_{i,j}))^2 + \frac{\alpha}{2} h \sum_{i=1}^{4} \sum_{j=1}^{J} u_i^2((j-1)h) \tag{5.2}$$

where h^2 is the area of a single square of the uni-form grid for discretization of the integral defined on Ω and h the step size for the integral defined on $\partial\Omega$. Note that $u_i \in \mathscr{P}_5(\partial\Omega_i)$, hence they are determined by their coefficients $w_i^j \in \mathbb{R}$ for $j = 0, \ldots, 5$, i.e.

$$u_i(t) = \sum_{j=0}^{5} w_i^j t^j \quad \text{for } i = 1, 2 \tag{5.3a}$$

$$u_i(t) = \sum_{j=0}^{5} w_i^j (1-t)^j \quad \text{for } i = 3, 4 \tag{5.3b}$$

with $t \in [0, 1]$ illustrated in Figure 12. The controls are these 24 coefficients, but they can be reduced using Equation (5.1c), which requires continuity in the corners of Ω.

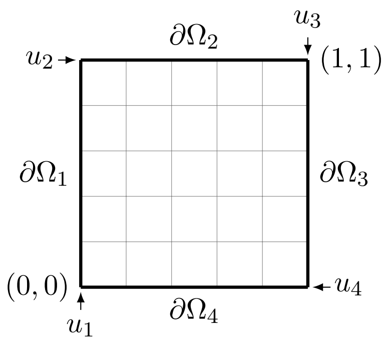

Figure 12: Discretization of Ω with uniform grid and boundary polynomials u_i for $i = 1, \ldots, 4$.

We can eliminate 4 coefficients as:

$$\begin{aligned} w_1^0 &= \sum_{i=0}^{5} w_4^i & \qquad w_3^0 &= \sum_{i=0}^{5} w_2^i \\ w_2^0 &= \sum_{i=0}^{5} w_1^i & \qquad w_4^0 &= \sum_{i=0}^{5} w_3^i \end{aligned} \tag{5.4}$$

Continuity then follows by definition and the controls are diminished to 20. The boundary states can be easily be eliminated using Equation (5.1f),

$$\begin{aligned} z_{i,0} &:= u_1(ih) & \qquad z_{i,J} &:= u_3(ih) \\ z_{J,j} &:= u_2(jh) & \qquad z_{0,j} &:= u_4(jh) \end{aligned} \tag{5.5}$$

for $i, j = 0, \ldots, J$. As in Chapter 4 we discretize the Laplace operator with finite differences. The discretization of the nonlinear PDE (5.1b) at an interior point $t_{i,j}$, i.e. $1 \leq j, m \leq I$, reads similar to Chapter 4 as

$$-h^{-2}(z_{i,j-1} + z_{i-1,j} - 4z_{i,j} + z_{i+1,j} + z_{i,j+1}) = \beta z_{i,j}^3. \tag{5.6}$$

As for Equation (4.10) we can eliminate the states associated with the boundary grid points from the left hand side. Substituting these states with Equation (5.5) and moving them to the right side leads e.g. for $t_{1,j}$ with $2 \leq j \leq I-1$ to

$$\begin{aligned} & -h^{-2}(z_{1,j-1} + z_{0,j} - 4z_{1,j} + z_{2,j} + z_{1,j+1}) && = \beta z_{1,j}^3 \\ \Leftrightarrow & -(z_{1,j-1} + u_4(jh) - 4z_{1,j} + z_{2,j} + z_{1,j+1}) && = h^2 \beta z_{1,j}^3 \\ \Leftrightarrow & -(z_{1,j-1} - (4 - h^2\beta z_{1,j}^2) z_{1,j} + z_{2,j} + z_{1,j+1}) && = u_4(jh) \end{aligned} \tag{5.7}$$

Recalling the lexicographic enumeration for the interior points introduced through Equation (4.9) we define the matrices $X^i_{L,\beta}[Z_L] \in \mathbb{R}^{I\times I}$ by

$$X^i_{L,\beta}[Z_L] = \begin{bmatrix} 4 - h^2\beta z^2_{(i-1)I+1} & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 - h^2\beta z^2_{iI} \end{bmatrix} \tag{5.8}$$

for $i = 1, \dots, I$ and the coefficient vector $Z_L = [z_1, \dots, z_N]^\top$ with $N = I^2$.
The nonlinear system of equations can be written as

$$A_{L,\beta}[Z_L] Z_L = b_L[w], \tag{5.9}$$

with the matrix

$$A_{L,\beta}[Z_L] = \begin{bmatrix} X^1_{L,\beta}[Z_L] & -\mathbb{1} & & \\ -\mathbb{1} & \ddots & \ddots & \\ & \ddots & \ddots & \mathbb{1} \\ & & -\mathbb{1} & X^I_{L,\beta}[Z_L] \end{bmatrix} \tag{5.10}$$

and the vector

$$b_L[w] = \begin{bmatrix} d_1[w] \\ d_2[w] \\ \vdots \\ d_{I-1}[w] \\ d_I[w] \end{bmatrix},$$

where the vectors $d_i[w]$ are defined as

$$d_1[w] = \begin{bmatrix} u_1(h) + u_4(Ih) \\ u_1(2h) \\ \vdots \\ u_1((I-1)h) \\ u_1(Ih) + u_2(h) \end{bmatrix}, \quad d_I[w] = \begin{bmatrix} u_3(Ih) + u_4(h) \\ u_3((I-1)h) \\ \vdots \\ u_3(2h) \\ u_3(h) + u_2(Ih) \end{bmatrix}$$

and

$$d_i[w] = \begin{bmatrix} u_4((I-i)h) \\ 0 \\ \vdots \\ 0 \\ u_2(ih) \end{bmatrix} \quad \text{for } i = 2, \dots, I-1.$$

This leads us to the reduced NLP

$$\underset{Z_L \in \mathbb{R}^N,\, w \in \mathbb{R}^{n_w}}{\text{minimize}} \quad \frac{1-\alpha}{2} h^2 \sum_{i=1}^{N} (Z_L^i - f_{\text{ref}}^{\gamma}(t_i))^2 + \frac{\alpha}{2} h \sum_{i=1}^{4} \sum_{j=0}^{J} u_i(jh)^2 \tag{5.11a}$$

$$\text{subject to} \quad A_{L,\beta}[Z_L] Z_L = b_L[w] \tag{5.11b}$$

where Equations (5.1c) - (5.1f) hold per definition.

5.3 INIS-MG Method

In this section, we want to derive an algorithm that combines INIS and MG to efficiently solve NLP (5.11). Therefore, we first define the objective function f_L and the constraints g_L as

$$f^L(Z_L, w) := \frac{1-\alpha}{2} h^2 \sum_{i=1}^{N} (Z_L^i - f_{\text{ref}}^{\gamma}(t_i))^2 + \frac{\alpha}{2} h \sum_{i=1}^{4} \sum_{j=0}^{J} u_i(jh)^2 \tag{5.12a}$$

$$g^L(Z_L, w) := A_{L,\beta}[Z_L] Z_L - b_L[w] \tag{5.12b}$$

The exact Jacobian g_{Z_L} reads as

$$g_{Z_L}^L(Z_L, w) = \begin{bmatrix} \tilde{X}_{L,\beta}^1[Z_L] & -\mathbb{1} & & \\ -\mathbb{1} & \ddots & \ddots & \\ & \ddots & \ddots & -\mathbb{1} \\ & & -\mathbb{1} & \tilde{X}_{L,\beta}^I[Z_L] \end{bmatrix} \in \mathbb{R}^{I^2 \times I^2} \tag{5.13}$$

with

$$\tilde{X}_{L,\beta}^i[Z_L] = \begin{bmatrix} 4 - 3h^2\beta z_{i(I)+1}^2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 - 3h^2\beta z_{iI+I}^2 \end{bmatrix} \tag{5.14}$$

Following the idea of an inexact method we want to use an approximate of $g_{Z_L}^L(Z_L, w)^{-1}$. A natural choice would be

$$A_{L,\beta}[\mathbb{0}]^{-1} = g_{Z_L}^L(\mathbb{0}, w)^{-1} \approx g_{Z_L}^L(Z_L, w)^{-1}, \tag{5.15}$$

where $A_{L,\beta}[\mathbb{0}]$ corresponds to the discretization matrix of the Poisson problem (4.5). We go one step further an use the linear operator that corresponds to on V-cycle of the MG method solving the equation

$$A_{L,\beta}[\mathbb{0}]x = b, \tag{5.16}$$

which we refer to as M^{-1}. One iteration of this combination of the INIS and MG method, called INIS-MG, is illustrated in Algorithm 6. Note that in step 6 the transposed $A_{L,\beta}[\mathbb{0}]^\top$ is used, but because $A_{L,\beta}[\mathbb{0}]$ is a symmetric matrix this results in the same linear operator M^{-1}.

Algorithmus 6 : `INIS_MG`$(A_{L,\beta}[\mathbb{0}], D, y_L^k, \delta z^0, \delta\lambda^0, \delta D^0, \omega, \nu_{\text{pre}}, \nu_{\text{post}}, l_{\min}, L, \eta)$

```
   // Get feasible direction
1  $\delta\bar{z} = -\texttt{multi\_grid}(A_{L,\beta}[\mathbb{0}], g(y_L^k), \delta z^0, \omega, \nu_{\text{pre}}, \nu_{\text{post}}, l_{\min}, L, \eta)$
   // Solve condensed QP
2  $b = Z^\top \nabla_y \mathcal{L}(y_L^k, \lambda_L^k) - Z^\top \tilde{H} \begin{bmatrix} \delta\bar{z} \\ \mathbb{0} \end{bmatrix}$
3  $\delta w = -(Z^\top \tilde{H} Z)^{-1} b$
   // Expand to full variable space
4  $\delta z = \delta\bar{z} - D^k \delta w$
5  $b = [\mathbb{1}_N \quad \mathbb{0}] \left(\nabla_y \mathcal{L}(y_L^k, \lambda_L^k) + \tilde{H}\delta y\right)$
6  $\delta\lambda = -\texttt{multi\_grid}(A_{L,\beta}[\mathbb{0}]^\top, b, \delta\lambda^0, \omega, \nu_{\text{pre}}, \nu_{\text{post}}, l_{\min}, L, \eta)$
   // Update iterates
7  $y_L^{k+1} = y_L^k + (\delta z^\top, \delta w^\top)^\top$
8  $\lambda^{k+1} = \lambda^k + \delta\lambda$
   // Update sensitivities
9  $B = g_z(y_L^k) D^k - g_w(y_L^k)$
10 $\delta D = -\texttt{multi\_grid}(M, B, \delta D^0, \omega, \nu_{\text{pre}}, \nu_{\text{post}}, l_{\min}, L, \eta)$
11 $D^{k+1} = D^k + \delta D$
```

6 Numerical Experiments with INIS-MG

In this chapter we perform numerical experiments on a PDE constrained optimal control test problem. We compare the MG method on solving the corresponding forward problem with the built-in `MATLAB` function `mldivide`. After we solved the test problem with INIS-MG and compared the result to a reference solution provided by `ipopt`, we have a look at the performance of the INIS-MG method regarding the CPU time for solving test problem. Last we provide a experimental verification of Corollary 1.

6.1 Software and Implementation Framework

This section aims on giving an overview of the software used within this thesis. All implementations were made using `MATLAB` on a laptop running Windows 10 equipped with an Intel i7.8565U and 16GB of RAM.

6.1.1 `CasADi`

For the computation of derivatives, we used the open-source tool `CasADi`, which calculates gradients, Jacobians, and Hessians with forward and reverse mode of algorithmic differentiation (AD) [19, 20]. It is written in self-contained C++ and was used via a full-featured interface to `MATLAB`.

6.1.2 `ipopt`

`ipopt` is an open-source software package for solving large-scale nonlinear optimization problems. `ipopt` implements an interior point line search filter method, hence the name Interior Point Optimizer. `ipopt` is the default solver for NLPs in CasADi and was usesd in this framework.
A comprehensive description of the algorithm can be found in [21], for more mathematical details see [22] and a guide to using `ipopt` can be found in [23].

Termination criterion

For the class of NLP problems (3.1) introduced in Chapter 3, `ipopt` terminates if an approximate solution (z^*, w^*, λ^*) satisfies the condition

$$E_0(z^*, w^*, \lambda^*) \leq \varepsilon, \tag{6.1}$$

where the optimality error $E_0(\cdot)$ is defined as

$$E_0(z^*, w^*, \lambda^*) := \max \left\{ \frac{\|\nabla f(z^*, w^*) + \nabla g(z^*, w^*)\lambda^*\|_\infty}{s_d(\lambda^*)}, \|g(z^*, w^*)\|_\infty \right\} \tag{6.2}$$

with scaling factor

$$s_d(\lambda^*) = \max \left\{ s_{\max}, \frac{\|\lambda^*\|_1}{(2n_{\mathrm{z}} + n_{\mathrm{w}})} \right\} / s_{\max} \tag{6.3}$$

and $s_{\max} = 100$.
In order to compare the INIS-MG method to `ipopt`, we implemented our algorithm with the same termination criterion.

6.2 Test Problem

As test problem we use the reduced NLP (5.11) introduced in Section 5.2, i.e.

$$\underset{Z_L \in \mathbb{R}^N,\, w \in \mathbb{R}^{n_\mathrm{w}}}{\text{minimize}} \quad \frac{1-\alpha}{2} h^2 \sum_{i=1}^{N} (Z_L^i - f_{\mathrm{ref}}^{\gamma}(t_i))^2 + \frac{\alpha}{2} h \sum_{i=1}^{4} \sum_{j=0}^{J} u_i(jh)^2 \tag{6.4a}$$

$$\text{,subject to} \quad A_{L,\beta}[Z_L] Z_L - b_L[w] = 0, \tag{6.4b}$$

with parameters

$$\alpha = 0.5, \quad \beta = 80, \quad \gamma = 4. \tag{6.5}$$

For a given $w^* \in \mathbb{R}^{20}$ the equation

$$\delta Z_L^k = -A_{L,\beta}^{-1}[\mathbb{0}](A_{L,\beta}[Z_L] Z_L - b_L[w^*]) \tag{6.6}$$

is solved in each iteration for the forward problem.
For a fine discretization of Ω we have $N \gg n_\mathrm{w} = 20$ which makes the condensing procedure very beneficial. Furthermore, the MG method is a very efficient solver for the Poisson equation, which has to be solved approximately several times in each INIS-MG iteration. Thus, the INIS-MG method is well suited for this test problem.

6.3 Linear System Solve: Multi-Grid vs. `mldivide`

In each INIS iteration, the equation

$$A_{L,\beta}[\mathbb{0}] x = b \tag{6.7}$$

has to be solved $n_w + 2$ times for varying right hand sides $b \in \mathbb{R}^N$. Instead of using the MG method to compute the solution x we could also use the `MATLAB` built-in

function `mldivide` commonly know as the "\" operator to solve such equations. The `mldivide` function takes the structure of matrix M into account to choose an appropriate solver and therefore solves the problem very efficiently, for details see [24]. In our case, we have a symmetric matrix where `mldivide` applies a Cholesky factorization using CHOLMOD [25], a set of routines for solving linear equations with sparse and symmetric positive definite matrices. So we are not able to solve the system with a user-provided tolerance.
For that reason, we want to compare the MG method to `mldivide` in solving a linear system that occurs when applying SQP and condensing to our test problem (6.4). We calculated the solution for different tolerances on the residual, i.e. for a user-provided $\varepsilon > 0$ the MG method terminates if the equation

$$\|A_{L,\beta}[\mathbb{0}]x - b\|_\infty < \varepsilon \tag{6.8}$$

is fulfilled. We also timed the computation time of a single V-cycle.
The MG method was called with the parameters

$$\nu_{\text{pre}} = 2, \quad \nu_{\text{post}} = 2, \quad l_{\min} = 0, \quad \eta = 1 \tag{6.9}$$

and the Richardson method with the parameter

$$\omega = 0.2. \tag{6.10}$$

We used the initial guess $x_0 = [0.005, \dots, 0.005]^\top$ and the right hand side $b = g(x_0)$. The results are illustrated in Figure 13.

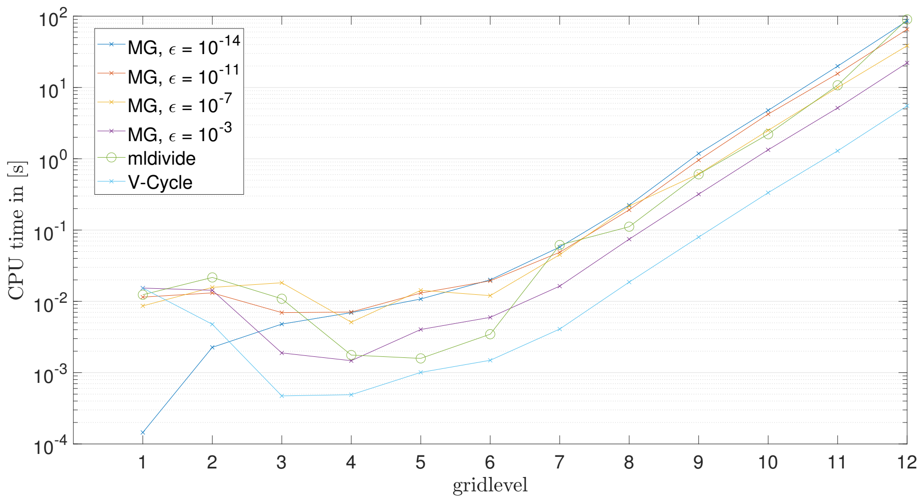

Figure 13: CPU time to compute solution of $A_{L,\beta}[0]x = g(x_0)$ for girdlevels $L = 1, \dots, 12$ using `mldivide`, the MG method with different tolerances and a single V-cycle.

The highest accuracy for the MG method was chosen as $\varepsilon = 10^{-14}$, which is the accuracy of `mldivide`.

We were not able to outperform `mldivide` significantly in calculating the exact solution on gridlevel $L = 12$, i.e. on a linear system with $(2^L - 1)^2 = 16,769,025$ unknowns. Because of lack of memory `MATLAB` was not able to perform `mldivide` on even higher gridlevels. Using only one V-cycle for solving Equation (6.7) approximately is about 10 times faster than using `mldivide` to compute the exact solution and leads to residuals in the order of magnitude 10^{-2}.

6.4 Solving the Test Problem

The purpose of this section is to illustrate the solution obtained by `ipopt` and verify that the INIS-MG method approximates the same solution.

The test problem was solved on gridlevel $L = 5$ with the initial guess

$$y_L^0 = [0.005, \dots, 0.005]^\top \tag{6.11}$$

and tolerance $\varepsilon = 10^{-6}$. `ipopt` and INIS-MG provided the approximate solutions

$$(Z^*_{\texttt{ipopt}}, w^*_{\texttt{ipopt}}),\ (Z^*_{\text{INIS}}, w^*_{\text{INIS}}) \in \mathbb{R}^{981}, \tag{6.12}$$

of the reduced NLP. These solutions have to be expanded back on to the boundary $\partial\Omega$ by evaluating the polynomials $u_1(\cdot), \dots, u_4(\cdot)$ which are defined by the coefficients w as explained in Section 5.2. The evaluation of these polynomials with coefficients $w^*_{\texttt{ipopt}}$ is illustrated in Figure 14 where we also marked the corners of Ω.

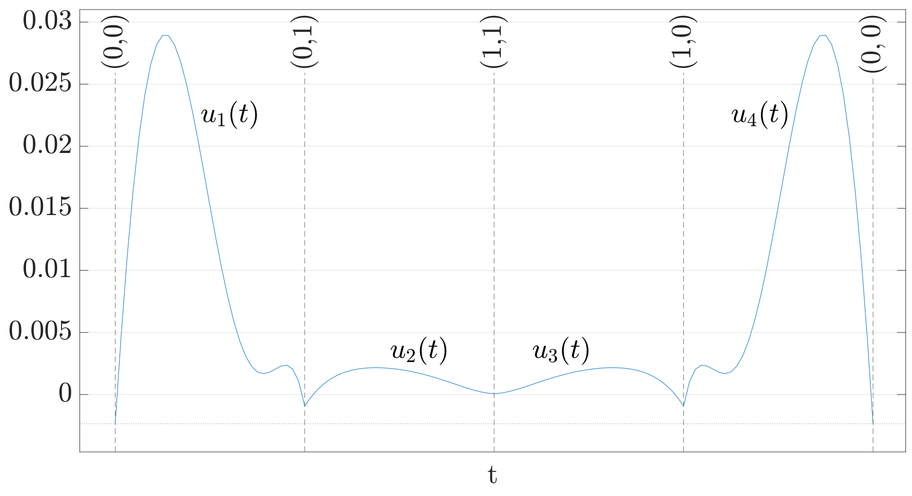

Figure 14: Polynomials $u_1(\cdot), \dots, u_4(\cdot)$ with coefficients $w^*_{\texttt{ipopt}}$.

Adding these evaluated polynomials to the solutions $Z^*_{\texttt{ipopt}}$, Z^*_{INIS} leads to Figure 15, which illustrates the solution $Z^*_{\texttt{ipopt}}$ of the PDE constraint optimal control problem (5.1) and the pointwise relative error $e^{\text{rel}}(Z^*_{\texttt{ipopt}}, Z^*_{\text{INIS}})$, which is defined by its entries

$$e_i^{\text{rel}}(Z^*_{\texttt{ipopt}}, Z^*_{\text{INIS}}) := \frac{|z^*_{\texttt{ipopt},i} - z^*_{\text{INIS},i}|}{|z^*_{\texttt{ipopt},i}|}, \tag{6.13}$$

for $i = 1, \dots, N$.

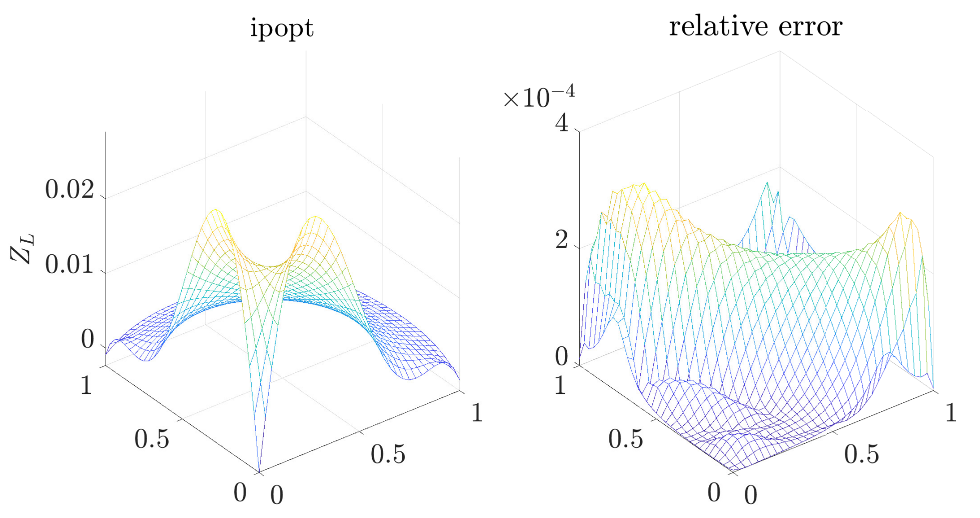

Figure 15: Plot of the expanded solution $Z^*_{\texttt{ipopt}}$ for gridlevel $L = 5$ and the relative error $e^{\text{rel}}(Z^*_{\texttt{ipopt}}, Z^*_{\text{INIS}})$.

As explained in Chapter 3, the controls w^* have a special role since they implicitly define $Z^*_L(w^*)$. Therefore we also want to have a closer look on $w^*_{\texttt{ipopt}}$ and w^*_{INIS}. Since we reduced the coefficients via Equation (5.4) we have to add the missing 4 coefficients. The coefficients $w^*_{\texttt{ipopt}}$ are illustrated in Figure 16 together with the relative error $e^{\text{rel}}(w^*_{\texttt{ipopt}}, w^*_{\text{INIS}})$.

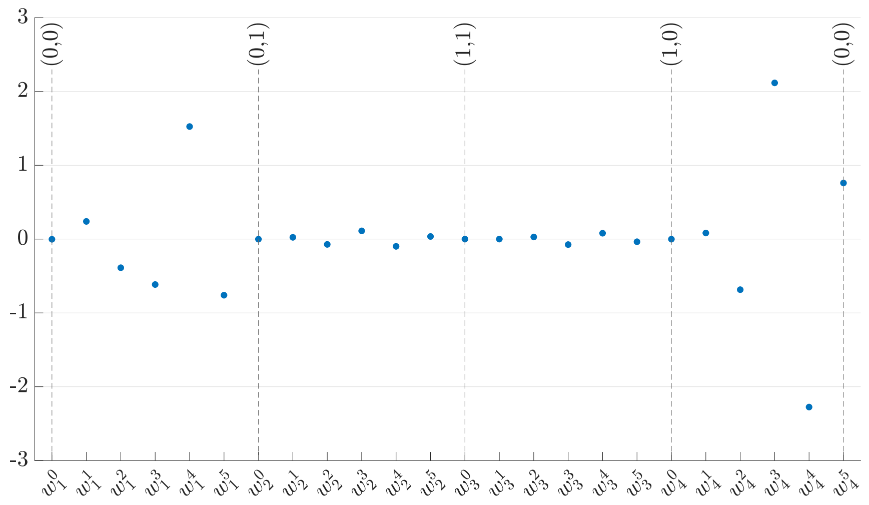

Figure 16: Plot of the expanded coefficients $w^*_{\texttt{ipopt}}$.

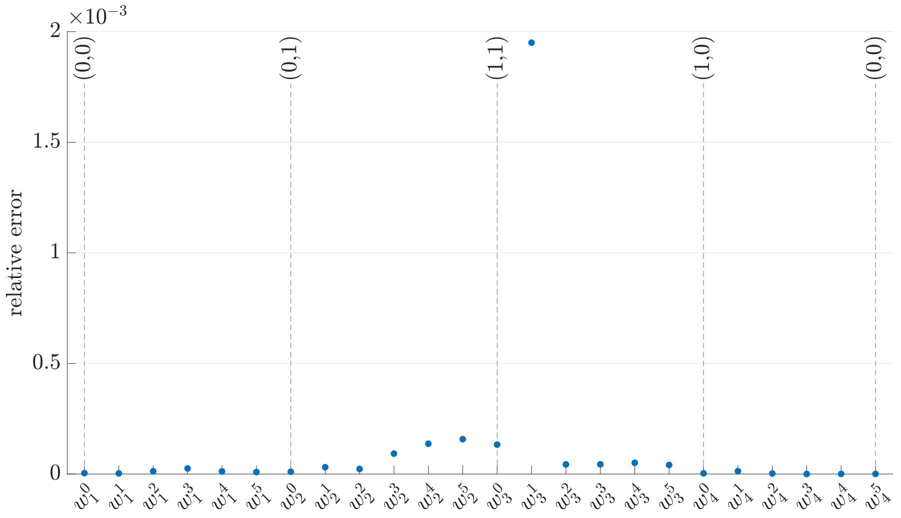

Figure 17: Relative error $e^{\mathrm{rel}}(w^*_{\texttt{ipopt}}, w^*_{\mathrm{INIS}})$ of expanded coefficients w^*_{INIS}.

6.5 Computation Times of INIS-MG and ipopt

In this section, we want to compare the INIS-MG method to `ipopt` regarding the CPU time for solving the test problem (6.4). As in the previous section we use the initial guess

$$y_L^0 = [0.005, \ldots, 0.005]^\top \tag{6.14}$$

and solve with accuracy

$$\varepsilon = 10^{-6}$$

We solved the test problem for the gridlevels $L = 1, \ldots, 11$ and summarized the CPU times in Figure 18. With these settings the INIS-MG method converged only for gridlevels $L \geq 4$.

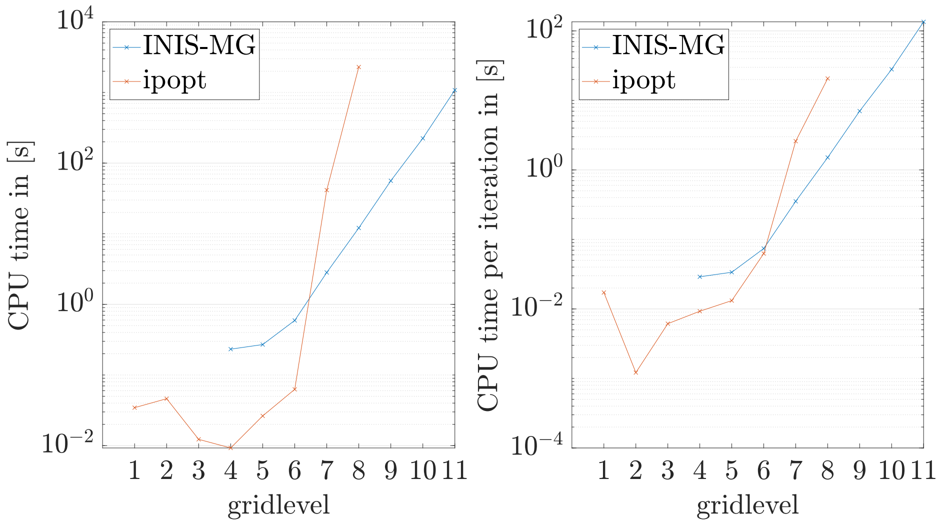

Figure 18: CPU time to compute NLP solution with INIS-MG and `ipopt` on different gridlevels.

Only `ipopt` was able to find an optimal solution on these low gridlevels, but it took `ipopt` too much time to compute solutions for gridlevels $L \geq 9$.

On gridlevels 2, 7 and 8 `ipopt` needed much more iterations to converge than on all the other gridlevels which explains the large jumps in the CPU time plot. The number of iterations needed on each gridlevel in order to converge are illustrated in Figure 19.

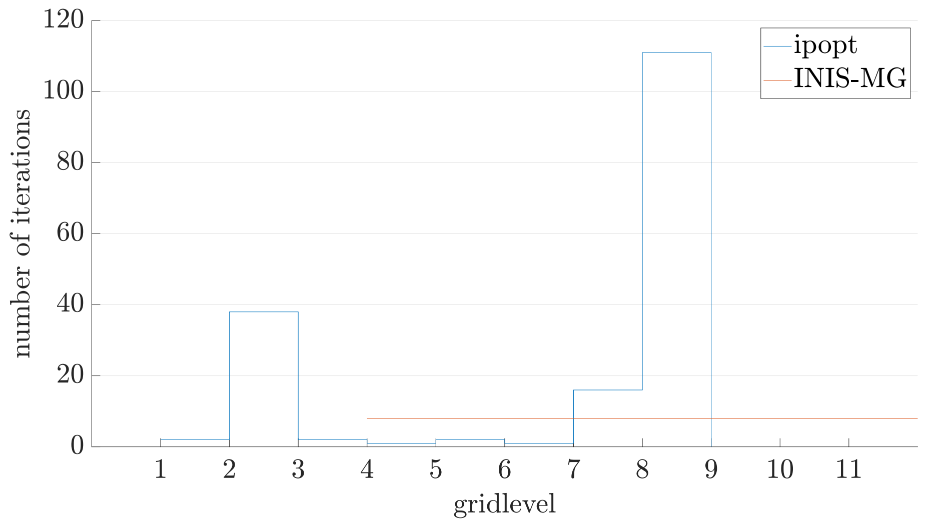

Figure 19: Number of iterations needed for convergence of the algorithm `ipopt` and INIS-MG solving the test problem on gridlevels $L = 1, \dots, 11$.

In order to get a better picture of the advantages of the INIS-MG method over `ipopt`, let us regard the factor θ_L by which the computation times $t_L^{\texttt{ipopt}}, t_L^{\text{INIS-MG}}$ differ, i.e.

$$\theta_L = \frac{t_L^{\texttt{ipopt}}}{t_L^{\text{INIS-MG}}}, \tag{6.15}$$

which is shown for a complete solve and a single iteration in Figure 20.

The NLP is solved faster by `ipopt` than with the INIS-MG method for gridlevels $L < 7$. Since it takes `ipopt` less iterations to converge than INIS-MG and a single iteration is also performed faster by `ipopt`. For gridlevels $L \geq 7$ the number of iterations of `ipopt` increases significantly while the number of iterations of the INIS-MG stays constant. Additionally, a single `ipopt` iteration is more expensive than a INIS-MG iteration. Hence, the INIS-MG method outperforms `ipopt` on gridlevels $L = 7, 8$ by the factors $\theta_7 = 17.55$ and $\theta_8 = 214.3$ for the complete solve.

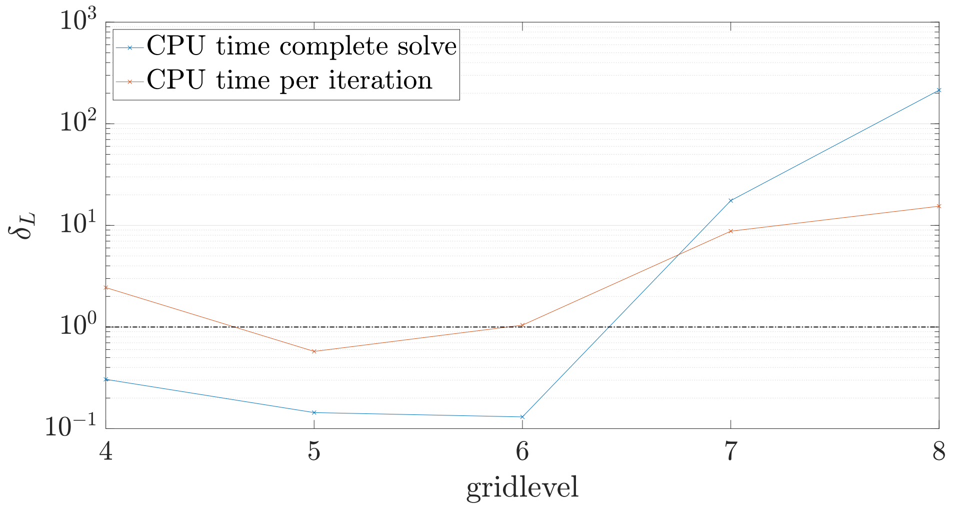

Figure 20: Factor $\theta_L = t_L^{\texttt{ipopt}} / t_L^{\text{INIS-MG}}$ on gridlevels $L = 4, \dots, 8$

6.6 Local Contraction of Forward Problem and INIS-MG

In this last section, we want to give an experimental verification of Corollary 1, which states the connection between the contraction rate of the forward problem and the INIS algorithm. In the case of the INIS-MG method, we use an exact Hessian. Corollary 1 states

$$\kappa^*_{\text{INIS}} = \kappa^*_F.$$

To verify this result experimentally we regard the error

$$e^k = |y^k - y^*| \tag{6.16}$$

for the iterates at iteration k of the INIS-MG method applied to the test problem (6.4) and the iterates of the MG method applied to the corresponding forward problem. For the forward problem we applied a single V-cycle to obtain the steps. The results are illustrated in Figure 21

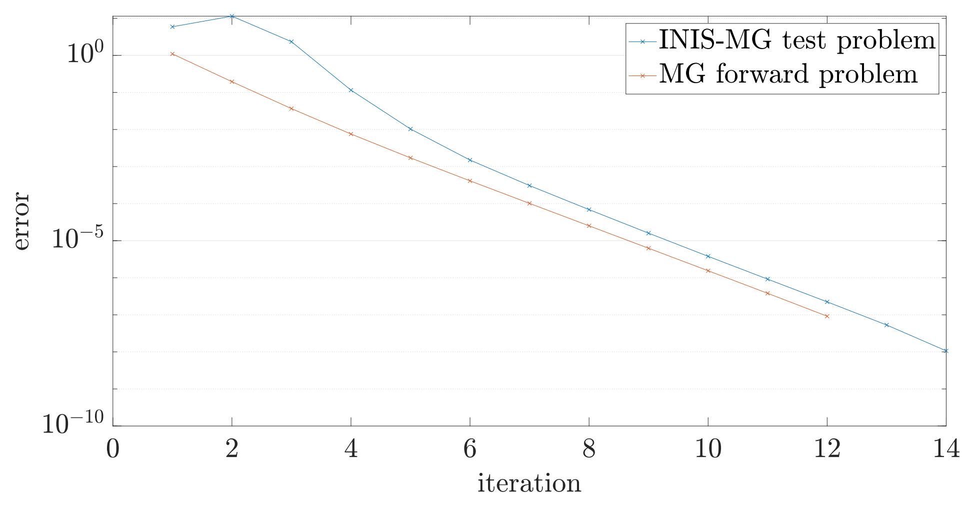

Figure 21: Plot of the error $|y^k - y^*|$ for the iterates of the INIS-MG method and the forward problem performed on gridlevel $L = 7$.

The Figure verifies that the methods convergence linearly with similar contraction rate which can be obtained as exponential of the slope, i.e.

$$\begin{aligned} \kappa^*_{\text{INIS-MG}} &\approx \exp(-1.4446) = 0.2358 \\ \kappa^*_F &\approx \exp(-1.4019) = 0.2461 \end{aligned} \tag{6.17}$$

Calculating κ^*_F through its definition yields

$$\kappa^*_F = \rho(M^{-1}g_z - \mathbb{1}_{n_z}) = 0.28232 \tag{6.18}$$

which is about $14 - 19\%$ larger. Hence, both algorithms convergence better than the theoretical result states.

7 Conclusion and Outlook

This thesis aimed at combining the MG method with the INIS method to obtain an efficient solver for PDE constraint optimal control problems.
Therefore, both the INIS and the MG method and its respective outstanding properties have been presented and discussed. For both methods, it was emphasized how they can be used to efficiently solve problems they are intended for. It was shown that the MG method is an appropriate method to be used within the INIS algorithm preserving the INIS specific convergence properties. In this thesis, only a special class of NLPs was regarded but the INIS-MG algorithm can easily be generalized for NLPs with inequality constraints and additional equality constraints. To test the INIS-MG algorithm a test problem was stated for which the algorithm is well suited.
Because all implementations were made using `MATLAB`, the MG method was compared against the built-in function `mldivide` on solving linear systems. Verifying that it is beneficial to use the MG method as a solver for the forward problem instead of `mldivide`. After verification that the INIS-MG method yields the same solution as the NLP solver `ipopt`, both solvers were compared with respect to the CPU time. Regarding the time needed for solving the 2-dimensional test problem, the INIS-MG method outperformed `ipopt` by a factor up to 200. Furthermore, an experimental verification of the connection between the contractions rates of the forward problem and the INIS-MG method was verified experimentally.
Future work includes the extension of the presented INIS-MG method with respect to more general PDEs, inequality constraints and 3-dimensional problems. Additionally, different versions, such as a version with an inexact hessian can be investigated. Moreover, a combination of the MG method with the IN method could be implemented and benchmarked against the INIS-MG method and `ipopt`. Furthermore, in order to improve the robustness of the INIS-MG method globalization strategies should

be integrated and it would be interesting to investigate if the INIS-MG method maintains its efficiency for other PDE constrainted optimal control problems.

Bibliography

[1] C. Meyer, “Optimizing the temperature profile during sublimation growth of sic single crystals: Control of heating power, frequency, and coil position,” 2004.

[2] I. Neitzel and F. Tröltzsch, “Numerical analysis of state-constrained optimal control problems for pdes,” 2000.

[3] H. Neunzert, *Progress in Industrial Mathematics at ECMI 94*. Vieweg+Teubner Verlag, 1996.

[4] P. Deuflhard, M. Seebass, D. Stalling, R. Beck, and H.-C. Hege, “Hyperthermia treatment planning in clinical cancer therapy: Modelling, simulation, and visualization,” *Computational Physics, Chemistry and Biology*, vol. 3, 1997.

[5] G. Leugering, S. Engell, A. Griewank, M. Hinze, R. Rannacher, V. Schulz, M. Ulbrich, and S. Ulbrich, *Constrained Optimization and Optimal Control for Partial Differential Equations*. 2012.

[6] M. Diehl, *Lecture Notes on Numerical Optimization*. 2016. (Available online: `http://cdn.syscop.de/publications/Diehl2016.pdf`).

[7] M. Diehl and S. Gros, *Numerical Optimal Control - script draft*. 2017. (Available online: `https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf`).

[8] R. H. Hoppe, “Chapter 4 sequential quadratic programming,” 2006. (Available online: `https://www.math.uh.edu/~rohop/fall_06/Chapter4.pdf`).

[9] R. Quirynen, S. Gros, and M. Diehl, "Inexact Newton-type optimization with iterated sensitivities," *SIAM Journal on Optimization*, vol. 28, no. 1, pp. 74–95, 2018.

[10] C. Marquardt, "Inexact newton-type optimization with iterated sensitivities for nonlinear model predictive control of cyclic processes," Master's thesis, 2019.

[11] A. Potschka, *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints.* PhD thesis, University of Heidelberg, 2011.

[12] W. Hackbusch, *Multi-Grid Methods and Applications*, vol. 4. Springer-Verlag Berlin Heidelberg GmbH, 1985.

[13] S. Bartels, *Numerical Approximation of Partial Differential Equations.* Springer International Publishing, 2016.

[14] S. Bartels, *Numerik* 3×9. *Drei Themengebiete in jeweils neun kurzen Kapiteln.* Heidelberg: Springer Spektrum, 2016.

[15] T. Lyche, "The poisson matrix and kronecker products." University of Oslo Norway, 2007.

[16] P. C. St John and F. J. Doyle III, "Estimating confidence intervals in predicted responses for oscillatory biological models," *BMC systems biology*, vol. 7, p. 71, 2013.

[17] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial, 2nd Edition.* 2000.

[18] P. Wessling, *An Introduction to Multigrid Methods.* John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, corrected reprint by R.T. Edwards, Inc. 2004, 1992.

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[20] "CasADi." `http://web.casadi.org`, 2020.

[21] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[22] A. Wächter, *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering.* PhD thesis, Carnegie Mellon University, 2002.

[23] S. Vigerske and A. Wächter, "Introduction to ipopt: A tutorial for downloading, installing, and using ipopt," 2015.

[24] Mathworks, "mldivide, \ ." `https://de.mathworks.com/help/matlab/ref/mldivide.html`, 2020.

[25] T. A. Davis, "User guide for cholmod: a sparse cholesky factorization and modication package." `https://fossies.org/linux/SuiteSparse/CHOLMOD/Doc/CHOLMOD_UserGuide.pdf` 2018.