Decomposition Methods for Parameter Identification and Bilevel Programming

Dissertation

submitted to the University of Bremen
for the degree of Dr. rer. nat.

by

Kai Schäfer

August 2023

Reviewers:	Prof. Dr. Christof Büskens, University of Bremen
	Prof. Dr. Kathrin Flaßkamp, Saarland University
Date of defense:	December 21, 2023

Abstract

This thesis deals with optimization problems that involve complex subtasks. Treating these problems as nonlinear programs often leads to difficulties in the numerical solution, for example the convergence to undesired local solutions for poor initial guesses.

To circumvent these problems, the concept of decomposition is introduced. The complex subtask is modeled by artificial variables and constraints. On the one hand, this increases the dimensionality of the problem, which on the other hand allows a user to employ suitable initialization strategies. This idea is applied to the problem classes of nonlinear parameter identification for dynamical systems and bilevel optimization. For the latter, a reformulation method is developed, which is based on embedding a fixed number of Sequential Quadratic Programming iteration steps to solve the subtask given as a nonlinear program.

The focus of this work is on the numerical verification of different decomposition approaches. Using the examples of a pendulum and a robotic system, it is demonstrated that the choice of the problem formulation in combination with a suitable initialization strategy influences the region of attraction of the global solution of the associated parameter identification problem. Finally, an extension of a single shooting homotopy approach is presented. In the area of bilevel optimization, examples are used to show that the number of embedded iteration steps increases the region of attraction of the global solution. The newly developed method is also compared with other established methods. Using a collection of bilevel problems, its handling, flexibility, and efficiency are investigated. It is also demonstrated for both problem classes that a decomposition has only a minor impact on computation times due to the exploitation of sparsity. In addition, decomposition allows to keep the required number of iterations stable despite poor initial guesses.

The obtained results show that both problem classes can be considered under a common aspect. Important criteria, such as the robustness or the region of attraction of the global solution, can be influenced solely by reformulating the problem.

Zusammenfassung

Diese Arbeit befasst sich mit Optimierungsproblemen, die komplexe Unteraufgaben aufweisen. Werden diese Probleme als nichtlineare Programme behandelt, führt dies oft zu Schwierigkeiten bei der numerischen Lösung, zum Beispiel Konvergenz zu unerwünschten lokalen Lösungen bei schlechten Anfangsschätzungen.

Um diese Probleme zu umgehen wird das Konzept der Zerlegung eingeführt. Die komplexe Unteraufgabe wird durch künstliche Variablen und Nebenbedingungen modelliert. Dies erhöht einerseits die Dimensionalität des Problems, welche es andererseits erlaubt, geeignete Initialisierungsstrategien zu verwenden. Diese Idee wird auf die Problemklassen der nichtlinearen Parameteridentifikation für dynamische Systeme und Bileveloptimierung angewendet. Für Letztgenanntes wird eine Umformulierungsmethode entwickelt, welche auf der Einbindung einer festen Anzahl an Iterationsschritten eines Sequentiellen Quadratischen Programmierungsansatzes zur Lösung der Unteraufgabe basiert. Die Unteraufgabe ist in diesem Fall ein nichtlineares Optimierungsproblem.

Der Fokus dieser Arbeit liegt auf der numerischen Überprüfung von verschiedenen Zerlegungsansätzen. Anhand der Beispiele eines Pendels und eines Robotiksystems wird demonstriert, dass die Wahl der Problemformulierung in Kombination mit einer geeigneten Initialisierungsstrategie Einfluss auf den Einzugsbereich der globalen Lösung des Parameteridentifikationsproblems hat. Abschließend wird eine Erweiterung eines Single Shooting-Homotopieansatzes vorgestellt. Im Bereich der Bileveloptimierung wird anhand von Beispielen gezeigt, dass die Anzahl der eingebundenen Iterationsschritte den Einzugsbereich der globalen Lösung vergrößert. Das neu entwickelte Verfahren wird zudem mit anderen etablierten Methoden verglichen. Anhand einer Sammlung von Bilevelproblemen wird dessen Handhabung, Flexibilität und Effizienz untersucht. Es wird außerdem für beide Problemklassen demonstriert, dass eine Zerlegung nur geringfügigen Einfluss auf Rechenzeiten aufgrund der Ausnutzung von schwach-besetzten Matrizen hat. Darüber hinaus ermöglicht die Zerlegung, dass die erforderliche Anzahl von Iterationen auch bei schlechten Ausgangswerten stabil bleibt.

Die erzielten Ergebnisse verdeutlichen, dass beide Problemklassen unter einem gemeinsamen Aspekt betrachtet werden können. Wichtige Kriterien, wie die Robustheit oder der Attraktionsbereich der globalen Lösung, können allein über eine Reformulierung des Problems beeinflusst werden.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Dr. Christof Büskens for his lasting confidence in my work, for the encouraging discussions, and for his patience with me. Moreover, I am very grateful for the opportunities he has given me for my professional and personal development.

I am also deeply indebted to Prof. Dr. Kathrin Flaßkamp, whom I have considered an academic role model since my undergraduate studies. Throughout the years, she has been a constant source of support, knowledge, and advice. I am grateful for the numerous discussions and the valuable impulses for my research. Lastly, I would like to thank her for reviewing this dissertation.

For two exciting months, I relocated my research to the University of Southampton, where the investigations on bilevel optimization began. I am grateful to Prof. Dr. Jörg Fliege for his hospitality and for the many interesting and fruitful discussions on the decomposition approach.

Thanks should also go to all of my colleagues for contributing to such a pleasant working atmosphere. Moreover, thanks to Dr. Matthias Knauer, Ivan Mykhailiuk, Dr. Shruti Patel, and Matthias Rick for proofreading parts of this thesis, and special thanks to Marcel Jacobse for proofreading the entire document. Many thanks to Dr. Margareta Runge, who has been an outstanding companion over the years. I am fortunate to look back on countless moments when we motivated each other to get things done. A big thanks to Marek Wiesner for the excellent collaboration, including all the fun nonsense in and out of the office.

I acknowledge funding from the University of Bremen and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project number 281474342.

Without the moral support of friends and family, I could not have made this journey. I am particularly grateful to Johann Funk, Janina Goos, and Lisa Müller for constantly reminding me that there is life outside of writing a dissertation. Most of all, I am grateful to my parents Karin and Uwe and my brother Tore for their unconditional love, support, and understanding.

Contents

List of Acronyms xiii

List of Figures xv

List of Tables xvii

List of Algorithms xix

1 Introduction 1

2 Nonlinear Programming 9

2.1 Theory . . . 9

2.1.1 Problem Formulation . . . 9

2.1.2 Regularity Conditions . . . 11

2.1.3 Optimality Conditions . . . 12

2.2 Numerical Solution Methods . . . 13

2.2.1 Sequential Quadratic Programming . . . 14

2.2.2 Interior-Point Methods . . . 17

2.2.3 The NLP Solver `WORHP` . . . 21

2.2.4 Sparse Matrices . . . 22

3 Decomposition Methods 23

3.1 Concept . . . 23

3.1.1 Problem Formulations . . . 24

3.1.2 Examples . . . 27

3.1.3 Related Work . . . 31

3.2 Decomposition in Parameter Identification for Dynamical Systems . 36
3.2.1 Original Problem Formulation 38
3.2.2 Transcription 42
3.2.3 Reduced Formulation 45
3.2.4 Direct Multiple Shooting 50
3.2.5 Decomposed Formulation 56
3.3 Decomposition in Bilevel Programming 62
3.3.1 Original Problem Formulation 63
3.3.2 Single-Level Reformulation 65
3.3.3 Reduced Formulation 66
3.3.4 KKT Formulation 68
3.3.5 Decomposed Formulation 71
3.4 Connections 84
3.4.1 Relation to the Concept 85
3.4.2 Comparison 86

4 Applications and Numerical Results 89
4.1 Parameter Identification for a Pendulum 89
4.1.1 Comparison Setup 90
4.1.2 Initialization Strategies 91
4.1.3 Local Solutions 92
4.1.4 Computational Characteristics 96
4.2 Parameter Identification for a Robotic System 97
4.2.1 Idealized Example 98
4.2.2 Comparison of Transcription Methods 100
4.2.3 Sparsity 103
4.2.4 Real-World Scenario 104
4.3 A Non-Unique Lower-Level Problem 105
4.3.1 Problem Description 106
4.3.2 Reduced Objective Function 107
4.3.3 Regions of Attraction 108
4.4 Increasing the Region of Attraction 109
4.4.1 Problem Description 110

4.4.2 Reduced Objective Function 110
4.4.3 Regions of Attraction 111
4.4.4 Computational Characteristics 114
4.5 Bilevel Decomposition Applied to a Problem Library 115
4.5.1 Comparison Setup 116
4.5.2 Number of Provided SQP Steps 119
4.5.3 Solution Strategies 121
4.5.4 Fischer-Burmeister Smoothing 124
4.5.5 Computational Characteristics 127
4.6 A Combined Homotopy-Optimization Approach 129
4.6.1 Single Shooting Homotopy Method 131
4.6.2 Homotopy Parameter Embedding 135

5 Conclusions **139**
5.1 Summary 139
5.2 Outlook 141

Bibliography **145**

List of Acronyms

BFGS	Broyden–Fletcher–Goldfarb–Shanno	17
D-BP	Decomposed Bilevel Problem	73
D-DPIP	Decomposed Dynamical Parameter Identification Problem	56
D-KKT-BP	Decomposed-KKT Bilevel Problem	74
FB	Fischer-Burmeister	6
IP	Interior-Point	2
IQR	Interquartile Range	96
IVP	Initial Value Problem	43
KKT	Karush-Kuhn-Tucker	6
KKT-BP	KKT Bilevel Problem	68
LICQ	Linear Independence Constraint Qualification	11
MPCC	Mathematical Program With Complementarity Constraints	69
MS-DPIP	Multiple Shooting Dynamical Parameter Identification Problem	51
NLP	Nonlinear Program	2
O-BP	Original Bilevel Problem	64
ODE	Ordinary Differential Equation	3
O-DPIP	Original Dynamical Parameter Identification Problem	40
PDE	Partial Differential Equation	25
QP	Quadratic Program	14
R-BP	Reduced Bilevel Problem	66
R-DPIP	Reduced Dynamical Parameter Identification Problem	46
SQP	Sequential Quadratic Programming	2

List of Figures

1.1 Scheme for solving optimization tasks 1

3.1 Sketch of a mathematical pendulum. 41
3.2 Measurements of the pendulum system 42
3.3 Reduced objective function (pendulum, one-dimensional) 47
3.4 Single and multiple shooting 52
3.5 Converging to the global solution 59
3.6 Influence of N_{it} on the solution 80
3.7 Influence of N_{it} on the computation time 84

4.1 Reduced objective function (pendulum, two-dimensional) 93
4.2 Attraction regions (reduced formulation, pendulum) 93
4.3 Attraction regions (multiple shooting formulation, pendulum) 95
4.4 Computational characteristics (pendulum) 97
4.5 Robotic systems. 98
4.6 Optimized trajectories of the robot's movement (synthetic data) . . 101
4.7 Influence of N_{s} on the robustness and solution quality 102
4.8 Relative densities 104
4.9 Optimized trajectories of the robot's movement (real data) 105
4.10 Validation of the robot's parameter identification 105
4.11 Lower-level objective function (Problem 4.1) 107
4.12 Reduced objective function (Problem 4.1) 107
4.13 Attraction regions (all formulations, Problem 4.1) 109
4.14 Feasible reduced objective function (Problem 4.2) 111
4.15 Attraction regions (reduced and KKT formulation, Problem 4.2) . . 112

4.16 Convergence frequencies (Problem 4.2) 113
4.17 Computational characteristics (Problem 4.2) 115
4.18 Influence of N_{it} on solving `BOLib` 120
4.19 Relative errors (`BOLib`, non-adaptive strategy) 123
4.20 Relative errors (`BOLib`, Fischer-Burmeister smoothing) 126
4.21 Performance profiles (`BOLib`, reduced and KKT formulation) 128
4.22 Performance profiles (`BOLib`, decomposed formulations) 129
4.23 Homotopy synchronization 133
4.24 Reduced objective function in homotopy formulation (pendulum) . . 134
4.25 Measurements of a robotic system 136
4.26 Reduced homotopy-optimization objective function (robot) 137
4.27 Convergence behavior of homotopy formulations 138

List of Tables

2.1 WORHP configuration parameters 21
2.2 Software and hardware specifications 22

3.1 Kepler example in decomposition notation 31
3.2 Notation of state representations 45
3.3 Problem dimensions 85
3.4 Problem classes in decomposition notation 86

4.1 Initialization strategies (pendulum) 92
4.2 Convergence frequencies (pendulum) 94
4.3 Nominal model parameters (idealized robotic system) 100
4.4 Success rates of solution strategies (BOLib) 122
4.5 Success rates of complementarity constraint formulations (BOLib) . . 125

List of Algorithms

A Full-step exact-Hessian SQP 15
B Primal-dual interior-point method 20

C Fischer-Burmeister smoothing 125
D Homotopy continuation for single shooting 135

Chapter 1

Introduction

One of the key aspects in many technical fields is cost reduction. Although usually understood in a monetary sense, costs are arbitrary. It can be of interest to reduce the energy required, the effort, the time, or the deviation from a reference. Although these aims are manifold and diverse, a common tool is often used for its achievement: *optimization*. Its application requires a thorough problem formulation with decisions on design variables, optimization criteria, or restrictions. Once such a problem is formulated, modern implementations of optimization algorithms can be used for its solution.

The underlying process can often be broken down to a simple scheme, as illustrated in Figure 1.1. Consider the task to steer a car driving from one position to another. This can be achieved by solving an optimal control problem, which would be the starting point or the *original problem*. As solving this is usually not straight-forward, a common strategy is to reformulate it as a problem for which established solution methods exist. We call this the *formulation* of the original problem. In a subsequent step, it is solved by an *algorithm*. This reformulation step has the advantage that modern implementations can be used. In the end, a practitioner is of course interested

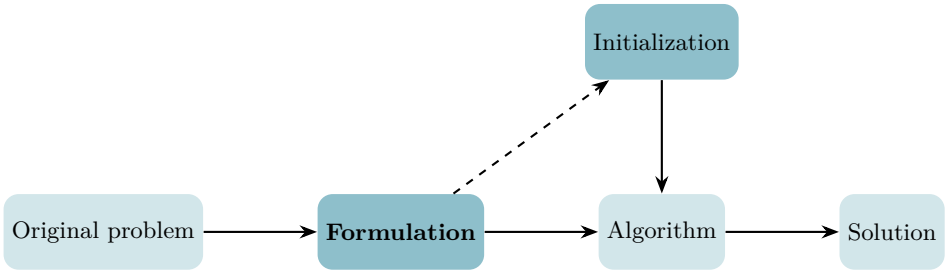


Figure 1.1: Schematic representation of a process for solving optimization tasks. The focus in this work is on the problem formulation.

in obtaining a *solution*, for example an optimal steering maneuver. Usually, the user has a limited view on the process, because both the formulation and the choice of the solution procedure require expert knowledge. However, these two aspects play an important role and can significantly influence finding a solution as well as its quality.

As general formulation, we are concerned with so-called *nonlinear programs (NLPs)*. Here, one aims to either minimize or maximize a function by finding those design variables that produce the smallest or largest possible output, respectively. Often, these variables are subject to restrictions of various complexity. The problems of interest are characterized by nonlinear functions and a finite number of variables and constraints. Formulations can differ by their dimensions (for example the number of optimization variables and constraints), their implementation effort, or their robustness when used by a specific algorithm. The latter aspect describes the ability of an algorithm in combination with the problem formulation to converge successfully. This criterion also gives information about the interplay between formulation and algorithm.

Only in rare cases, NLPs can be solved analytically. In many cases, numerical routines need to be applied. They can be divided into *global* and *local* algorithms. The former (see [87] for an overview) aim to find the best solution among all local ones. Many global optimization techniques have in common that a sequence of problems is solved using local optimization routines and typically, the overall computational efforts are much larger than in local optimization. Consequently, manageable problem dimensions are much smaller. For many types of optimization problem classes, a global solution is desired, but the computational effort should still be adequate. Mainly for this reason, we abstain from applying global optimization algorithms in the following and instead focus on local routines, which are designed to converge to the closest local solution for a given initialization of optimization variables. There exists a large variety of iterative local algorithms, for example Sequential Quadratic Programming (SQP) [16], trust-region methods [41], or Interior-Point (IP) approaches [44]. In this work, we assume that the *algorithm* (see Figure 1.1) is a fixed one and will use the SQP and IP methods provided by the NLP solver `WORHP` [20, 71] for solving the occurring problems. It might seem that fixing the algorithm is a hard restriction, but in fact, it is even desirable to keep solution processes as general as possible and thus, one strives to use the same algorithm for a variety of problems.

As mentioned before, it is usually of interest to compute a global solution of the given problem. However, the objective function value may not be the only criterion

worth considering. For real-world applications in particular, a reliable and robust solution process is required by practitioners. In other words, it is desired that the algorithm converges to the same solution even for poorly chosen initial guesses. In that regard, a solution's *region of attraction* can loosely be defined as the set of optimization variable initializations from within a pre-specified region that enable the solver to converge to that solution. A similar definition is for example given by Mykhailiuk et al. [86].

In many applications, an accurate representation of reality is required. Complex models, nonlinear terms, costly evaluations, or even embedded iterative schemes find their way into the system design. Consequently, these aspects occur in the optimization process, which is typically built around all the design decisions. Despite the advantages of an accurate model description, difficulties within the optimization process are likely to occur: costly evaluations add up within the solution process, the model may react sensitively to its input, or highly nonlinear terms appear. Due to the latter, the problem may become nonconvex, which leads to multiple local solutions although only a global solution is usually of interest. From a practical point of view, local algorithms may be trapped in the closest minimum, depending on the initialization. Since optimization is often only one part in automated processes, practitioners have to rely upon the outcome. In real-time environments, this may even be a crucial factor, as a poor solution quality may induce an undesired system behavior.

In this work, it is proposed to address the aforementioned challenges *only* by modifying the way the original problem is formulated (compare Figure 1.1). The technique in use is *decomposition*: replace complex terms within a problem by suitable artificial variables and constraints. Focusing on this part has the consequence that other aspects will find less attention, in particular the choice of the optimization algorithm. Two problem classes that exhibit the above-mentioned challenges find closer attention within this work: *parameter identification for dynamical systems* and *bilevel programming*.

Many time-dependent systems can be modeled by ordinary differential equations (ODEs). They shall be used for simulation or (optimal) control. In a preceding step, however, the model needs to be adapted to the given circumstances. While the general structure of the equations of motion is often known, parameters p in the ODE need to be calibrated for the specific system at hand. They can be identified by minimizing the distance between the ODE's solution q and measurements $\bar{q}$, recorded

at N_m time points t_i. This task can be written in the following form:

$$\begin{aligned} \underset{p,\,q}{\text{minimize}} \quad & \sum_{i=1}^{N_\mathrm{m}} (q(t_i;p) - \bar{q}_i)^2 \\ \text{subject to} \quad & q \text{ solves ODE}(p). \end{aligned}$$

A common way to solve these problems is the direct approach, which makes use of a time discretization. Instead of the original problem above, a finite-dimensional NLP involving a discretized version of q is solved. For its formulation, there exist several techniques that lead to equivalent problems. However, all of them may be suited differently well for being solved by standard solvers. It is, for example, well-known that the *single shooting* approach is less robust than a *direct multiple shooting* method [12]. What receives less attention in the literature is the problem of local solutions, although their occurrence is a common phenomenon. In [47], this is addressed with the help of global optimization and regularization. The latter is also used in [72] to reduce the number of local solutions. Still, what influences the convergence to specific minima has not yet been fully investigated.

The class of bilevel optimization deals with problems involving a subordinate problem. Hence, two levels exist. The upper-level problem depends on the solution of the lower-level problem, which can itself be parameterized in the upper-level variables. Such a problem may be written in the following form:

$$\begin{aligned} \underset{x,\,y}{\text{minimize}} \quad & F(x,y) \\ \text{subject to} \quad & y \text{ solves NLP}(x). \end{aligned}$$

This intricate structure makes this problem challenging in many facets. For its numerical solution, a reduction to a single-level NLP is common. There exist several possibilities, for example using value functions [127] or including lower-level optimality conditions into the upper-level problem (see [3], for example). The resulting problems are often nonconvex and the issue of local solutions inevitably arises. In addition, customized algorithms are often required for their solutions.

Although the two problem classes appear to stand alone, they share some common characteristics. For their practical solution, they can be formulated as an NLP, solvable by standard algorithms. Both problems require solving another parameterized task, which is why it is desirable to find a common approach. Parameter identification requires solving an ODE, while an NLP needs to be solved in bilevel programming. From a numerical point of view, this comes down to embedding iterative schemes

within a problem. Two questions directly emerge from this interaction:

- How can such embeddings be formulated?
- Is it possible to apply similar techniques for both classes?

Aims and Contributions

The goal of this work is to reliably find desired minima using decomposition techniques. The development of customized NLP formulations is in the center. Those reformulations should meet certain criteria when a standard local optimization algorithm is applied to solve them:

(i) The solution process shall be *robust*. Even for poor solver initializations, the algorithm shall converge.
(ii) A practitioner shall be able to *influence* the algorithmic behavior. By adequate solver initializations, convergence to desired minima shall be enabled.
(iii) For such a desired solution, the *region of attraction* shall be enlarged. Suitable initialization and formulation choices shall compensate for poor initial guesses.
(iv) The *computational effort* shall remain moderate. Even though the reformulations suffer from higher dimensions, their specific structures shall be exploited.

To this end, the idea of decomposition is illuminated. Starting from an original problem, a naive reformulation approach is examined and called into question. Subsequently, alternative formulations building on the idea of decomposition are developed and analyzed. This abstract concept of decomposition is applied to two problem classes.

For the area of parameter identification, an experimental comparison of different existing transcription methods is provided. For this purpose the methods are initially derived and examined on the basis of the above aspects. Using the example of a mathematical pendulum, it is shown that a parameter identification problem can exhibit several local solutions. It is demonstrated that the choice of shooting nodes in multiple shooting as well as their initialization influences the solution quality significantly. Similar studies using a more complex example of a robotic system provide comparable results. They add new insights to the existing, but expandable literature. In addition, an extension to an existing homotopy method is developed. The idea is to embed the homotopy continuation procedure into a single NLP. This approach is independent of the chosen transcription method. Initial experiments demonstrate its applicability.

For the area of bilevel programming, the contributions are two-fold. On the one hand, a novel reformulation technique is developed. It is based on embedding a finite number of SQP iteration steps into an optimization problem. Thus, it builds upon the well-known *Karush-Kuhn-Tucker (KKT) approach*. It is compatible with *Fischer-Burmeister (FB) smoothing* techniques and allows customized initialization strategies. On the other hand, this approach is compared against other existing reformulations with the help of extensive numerical experiments. Examinations of specific examples show how a solution's region of attraction can be increased by applying the novel method. The highlight is the application of the considered methods on a problem library, which is not yet the standard in the corresponding literature.

Overview

As the formulation and numerical solution of NLPs are key aspects of this work, the necessary preliminaries are given in **Chapter 2**. The basic definitions, theorems, and concepts are formally presented. Equal attention is given to numerical solution methods for NLPs. Two standard algorithms, that find application in the latter chapters, are introduced in their basic form. This chapter is a reproduction of existing theory.

The abstract problem formulation is developed in **Chapter 3**, in which the problem of interest is formally introduced. A categorization into the mathematical context as well as illustrative case studies follow. Next, the idea of decomposition is applied to the problem classes of parameter identification and bilevel programming. In both cases, the structure is identical: First, the original problem is formally introduced and the reformulation as NLP is discussed. Next, a *reduced* version is presented and possible drawbacks are discussed. Subsequently, an alternative *decomposed* formulation is introduced and its possible advantages are highlighted. In the case of parameter identification, existing transcription methods like single and multiple shooting as well as full discretization are introduced and analyzed. In bilevel programming, a reduced approach, the well-known KKT approach, as well as a newly developed decomposed approach are discussed. This approach is illustrated by examples that cover aspects like solution quality and sparsity considerations. Finally, variations of this approach are discussed, for example FB techniques.

Chapter 4 is a collection of examples to substantiate the hypotheses made in this work. It consists of comparing formulation techniques against each other with respect to several criteria. In particular, the effects to parameter identification are analyzed with respect to a mathematical pendulum and a robotic system. For

bilevel programming, two example problems are discussed in detail with respect to the region of attraction. The novel reformulation method is subsequently tested on a library of problems. Many examples within this chapter are accompanied by investigating computational aspects such as computation times or required numbers of iterations. This chapter closes with an idea to enhance a homotopy algorithm for parameter identification by combining it with optimization. It is illustrated by examples and initial experiments.

The dissertation closes with a short summary of its findings, concluding remarks, and an outlook on possible extensions.

Chapter 2

Nonlinear Programming

Optimization problems occur in almost every field of application, and their solution is often a difficult endeavor. They can be formulated in various forms, for example involving dynamical models, being unconstrained or constrained, or as optimization tasks that require solving a nested optimization problem. A common approach to solving different problems similarly is to reformulate them as standard finite-dimensional nonlinear programs. A formulation in a standardized manner has the advantage that one can make use of state-of-the-art solution methods that are designed to solve a broad range of these problems. Although this problem reformulation often involves approximations or reductions at some point, it is a well-established procedure.

This chapter has the purpose of giving a brief introduction into the topic of *nonlinear programming*. In the following, the standard problem formulation is presented and necessary theoretical concepts are introduced. Besides a formal introduction, numerical solution methods are presented. This chapter lays the foundation for all following investigations in this thesis.

2.1 Theory

The following introduction is a composition of contents from [40], [49], and [89]. Whenever necessary, a more precise reference is given.

2.1.1 Problem Formulation

In nonlinear programming, one is interested in finding the value of an *optimization variable* $z \in \mathbb{R}^{n_z}$ that optimizes a given *objective function* $F\colon \mathbb{R}^{n_z} \to \mathbb{R}$ with

respect to *constraints* $z \in S \subseteq \mathbb{R}^{n_z}$. Here, S is assumed to be represented by *inequality constraints* $G(z) \leq 0_{n_G}$[1] for a function $G\colon \mathbb{R}^{n_z} \to \mathbb{R}^{n_G}$ and *equality constraints* $H(z) = 0_{n_H}$ for a function $H\colon \mathbb{R}^{n_z} \to \mathbb{R}^{n_H}$:

$$S \coloneqq \{z \in \mathbb{R}^{n_z} \colon G(z) \leq 0_{n_G}, H(z) = 0_{n_H}\}.$$

An optimization problem is called an NLP if at least one of the involved functions is *nonlinear*.

Due to the following remark, we restrict ourselves to minimization tasks, in other words we aim at finding those variables $z \in S$ for which F attains a minimal value.

Remark 2.1 It holds that $\max F(z) = -\min -F(z)$.

The described problem formulation is summarized in the following notion:

Problem 2.2 (NLP)

Find variables $z \in \mathbb{R}^{n_z}$ that

$$\begin{aligned} \text{minimize} \quad & F(z) \\ \text{subject to} \quad & G(z) \leq 0_{n_G} \\ & H(z) = 0_{n_H}. \end{aligned}$$

A point z that fulfills the constraints is called a *feasible point*, while the set S containing all feasible points is called the *feasible region*. If it is empty, the problem is called *infeasible*. If $G_i(z) = 0$ for $z \in \mathbb{R}^{n_z}$ and an $i \in \{1, \dots, n_G\}$, this constraint is called *active at* z; the set

$$\mathcal{I}(z) \coloneqq \{i \in \{1, \dots, n_G\} \colon G_i(z) = 0)\}$$

is called the *set of active constraints at* z, or short the *active set*. For the characterization of optimal points, the notion of a neighborhood has to be introduced. For a point $\hat{z} \in \mathbb{R}^{n_z}$ and a radius $\delta > 0$, the set

$$\mathcal{U}_\delta(\hat{z}) \coloneqq \{z \in \mathbb{R}^{n_z} \colon \|z - \hat{z}\|_2 \leq \delta\}$$

is called *δ-neighborhood of $\hat{z}$*. With the collected ingredients, optimal points can be defined.

[1] The symbol 0_n represents an n-dimensional column vector containing zeros.

Definition 2.3 For (NLP), a feasible point $z^\star \in S$ is a

(i) *local minimum* if there is a $\delta > 0$ such that $F(z^\star) \leq F(z)$ for all $z \in \mathcal{U}_\delta(z^\star) \cap S$;
(ii) *strict local minimum* if there is a $\delta > 0$ such that $F(z^\star) < F(z)$ for all $z \in \mathcal{U}_\delta(z^\star) \cap S$, $z \neq z^\star$;
(iii) *global minimum* if $F(z^\star) \leq F(z)$ for all $z \in S$;
(iv) *strict global minimum* if $F(z^\star) < F(z)$ for all $z \in S$, $z \neq z^\star$.

This definition, however, is practically not applicable, since F would have to be evaluated at an infinite number of points to find out whether a given point is optimal. Therefore, conditions are introduced which provide a criterion to check whether a given point is optimal.

Definition 2.4 The *Lagrange function* $\mathcal{L}\colon \mathbb{R}^{n_z} \times \mathbb{R}^{n_G} \times \mathbb{R}^{n_H} \to \mathbb{R}$ is defined as

$$\mathcal{L}(z, \lambda, \mu) = F(z) + \lambda^\mathsf{T} G(z) + \mu^\mathsf{T} H(z)$$

and λ and μ (or synonymously their components) are called *Lagrange multipliers* or *dual variables*, while z is also referred to as being the *primal variable*.

To be prepared for the following considerations, we assume that F, G, and H are twice continuously differentiable from now on.

2.1.2 Regularity Conditions

To formulate conditions for a local minimum, so-called regularity conditions (or constraint qualifications) have to be introduced.

Definition 2.5 A feasible point $z \in S$ fulfills the *Mangasarian-Fromovitz Constraint Qualification* if $\nabla H_i(z)$, $i = 1, \ldots, n_H$, are linearly independent and there exists a vector $d \in \mathbb{R}^{n_z}$ such that $\nabla G_i(z)^\mathsf{T} d < 0$ for $i \in \mathcal{I}(z)$ and $\nabla H_i(z)^\mathsf{T} d = 0$ for $i = 1, \ldots, n_H$. In this case, z is called a *regular point*.

The following definition provides another constraint qualification:

Definition 2.6 A feasible point $z \in S$ fulfills the *Linear Independence Constraint Qualification (LICQ)* if $\nabla G_i(z)$, $i \in \mathcal{I}(z)$, and $\nabla H_i(z)$, $i = 1, \ldots, n_H$, are linearly independent. In this case, z is called a *normal point*.

Their relation is described in the following theorem:

Theorem 2.7 *Let $z \in S$ be a feasible point at which the LICQ is fulfilled. Then, the Mangasarian-Fromovitz Constraint Qualification is also fulfilled at this point.*

The proof can be found in [49, Theorem 2.41], for example.

2.1.3 Optimality Conditions

The definitions of regularity and the Lagrange function allow formulating optimality conditions. We begin with the necessary, so-called KKT conditions, which make use of the derivative of $\mathcal{L}$ with respect to z:

$$\nabla_z \mathcal{L}(z, \lambda, \mu) = \nabla F(z^\star) + \nabla G(z^\star)^\mathsf{T} \lambda + \nabla H(z^\star)^\mathsf{T} \mu.$$

Theorem 2.8 *Let $z^\star$ denote a local minimum of (NLP). If $z^\star$ is regular, then there exists a vector $\lambda \in \mathbb{R}^{n_G}$ and a vector $\mu \in \mathbb{R}^{n_H}$ such that the following conditions hold:*

$$\nabla_z \mathcal{L}(z^\star, \lambda, \mu) = 0_{n_z}, \tag{2.1}$$
$$G(z^\star) \leq 0_{n_G}, \tag{2.2}$$
$$H(z^\star) = 0_{n_H}, \tag{2.3}$$
$$\lambda^\mathsf{T} G(z^\star) = 0, \tag{2.4}$$
$$\lambda \geq 0_{n_G}. \tag{2.5}$$

A proof can be found in [49], for example.

Remark 2.9 Condition (2.4) is in so-called scalar-product form, while it is often equivalently stated as $\lambda_i G_i(z^\star) = 0$ for $i = 1, \ldots, n_G$.

Remark 2.10 If $z^\star$ is normal, then the Lagrange multipliers are unique.

A point $(z^\star, \lambda, \mu)$ satisfying the conditions (2.1) to (2.5) is called a *KKT point*. The presented conditions are also called *first-order conditions*, since only first-order derivatives of the Lagrange function are needed. To formulate *second-order conditions*, we have the following theorem:

Theorem 2.11 *Let $z^\star$ denote a local minimum of (NLP). If $z^\star$ is regular, then there exists $\lambda \in \mathbb{R}^{n_G}$ and $\mu \in \mathbb{R}^{n_H}$ such that* (2.1) *to* (2.5) *hold and*

$$d^\mathsf{T} \nabla^2_{zz} \mathcal{L}(z^\star, \lambda, \mu) d \geq 0$$

for all $d \in C(z^\star)$ with

$$\begin{aligned} C(z^\star) \coloneqq \{d \in \mathbb{R}^{n_z} : \; & d^\mathsf{T} \nabla G_i(z^\star) \leq 0 \text{ for } i \in \mathcal{I}(z^\star) \text{ and } \lambda_i = 0, \\ & d^\mathsf{T} \nabla G_i(z^\star) = 0 \text{ for } i \in \mathcal{I}(z^\star) \text{ and } \lambda_i > 0, \\ & d^\mathsf{T} \nabla H_j(z^\star) = 0 \text{ for } j = 1, \ldots, n_H\}. \end{aligned}$$

$C(z^\star)$ is called the critical cone *at $z^\star$.*

This is proven in [49], for example.

Theorem 2.8 does not only give necessary conditions, it serves in many numerical solution methods as a termination criterion. However, it is not ensured that $z^\star$ is a minimum point. Sufficient second-order conditions are needed:

Theorem 2.12 *If there exist vectors $\lambda \in \mathbb{R}^{n_G}$ and $\mu \in \mathbb{R}^{n_H}$ such that* (2.1) *to* (2.5) *hold and*

$$d^\mathsf{T} \nabla^2_{zz} \mathcal{L}(z^\star, \lambda, \mu) d > 0$$

for all $d \in C(z^\star) \backslash \{0_{n_z}\}$, then $z^\star$ is a strict local minimum of (NLP).

A proof of this theorem can also be found in [49].

2.2 Numerical Solution Methods

Only in rare cases, (NLP) can be solved analytically. For the majority of problems, especially in the context of real-world applications, one cannot expect to be able to calculate a solution by hand. In this section, therefore, we are concerned with the *numerical* solution of (NLP). Many algorithms are based on constructing a sequence of iterates. During the course of the algorithm, this sequence is supposed to converge to a local minimum. In practice, this amounts to iteratively improving an initial guess until first-order optimality conditions are fulfilled up to a given tolerance. Although each problem class benefits from tailored solution methods, there are two

approaches that have turned out to be efficient and applicable in the majority of cases: SQP and IP methods — introduced in the following.

2.2.1 Sequential Quadratic Programming

The idea of SQP is to improve an initial guess iteratively by solving a suitable, easier-to-solve subproblem. However, before starting to derive a basic SQP algorithm, some major ingredients have to be defined. SQP methods are based on solving quadratic programs (QPs), which are characterized by a quadratic objective function with linear equality and inequality constraints. To be precise, for vectors $c \in \mathbb{R}^{n_d}$, $d \in \mathbb{R}^{n_d}$, $v \in \mathbb{R}^{n_v}$, $w \in \mathbb{R}^{n_w}$, and matrices $A \in \mathbb{R}^{n_v \times n_d}$, $B \in \mathbb{R}^{n_w \times n_d}$, and $W \in \mathbb{R}^{n_d \times n_d}$, a standard QP is given as follows:

Problem 2.13
Find variables $d \in \mathbb{R}^{n_d}$ that

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} d^\mathsf{T} W d + c^\mathsf{T} d \\ \text{subject to} \quad & Ad \leq v \\ & Bd = w. \end{aligned}$$

Remark 2.14 If W is positive semi-definite, Problem 2.13 is a *convex QP*.

As in Theorem 2.8, we can formulate necessary optimality conditions for this problem[2]. Thus, there exist dual variables $\iota \in \mathbb{R}^{n_v}$ and $\kappa \in \mathbb{R}^{n_w}$ such that

$$\begin{aligned} Wd + c + A^\mathsf{T}\iota + B^\mathsf{T}\kappa &= 0_{n_d}, \\ Ad - v &\leq 0_{n_v}, \\ Bd - w &= 0_{n_w}, \\ \iota^\mathsf{T}(Ad - v) &= 0, \\ \iota &\geq 0_{n_v} \end{aligned}$$

holds. In this case, constraint qualifications are not needed to formulate necessary optimality conditions due to the linearity of the constraints. There exist many

[2]Necessary optimality conditions for QPs will be used in Subsection 3.3.5, where they will appear as artificial problem constraints.

Algorithm A Full-step exact-Hessian SQP

1: Choose $z^{[0]}$, $\lambda^{[0]}$, $\mu^{[0]}$, set $k = 0$
2: **while** Termination criterion is not fulfilled **do**
3: Solve (QP) and obtain $d^{[k]}$ together with $\lambda^{[k+1]}$, $\mu^{[k+1]}$
4: Update $z^{[k+1]} = z^{[k]} + d^{[k]}$
5: Update $k \leftarrow k + 1$
6: **end while**

advanced solvers for the numerical solution of Problem 2.13, such as `qpOASES` [36], `HPIPM` [46], or `WORHP`'s internal solvers [62].

With an initialization $z_{\text{ini}} \coloneqq z^{[0]}$, the general idea of SQP methods is to iteratively improve a current iterate $z^{[k]}$ by the solution $d^{[k]}$ of a suitable quadratic subproblem. This is formulated in such a way that its objective function serves to approximate the Lagrange function $\mathcal{L}$ and the constraints linearize the original ones in (NLP) with the current iterate as evaluation point. Herein, the dual variables λ and μ have to be updated as well, which is why they have to be initialized, too. For the exact Hessian $\mathcal{H} \coloneqq \nabla^2_{zz}\mathcal{L}(z, \lambda, \mu)$, the problem reads as follows:

Problem 2.15 (QP)

Find variables $d \in \mathbb{R}^{n_z}$ that

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} d^\mathsf{T} \mathcal{H} d + \nabla F(z)^\mathsf{T} d \\ \text{subject to} \quad & G(z) + \nabla G(z)^\mathsf{T} d \leq 0_{n_G} \\ & H(z) + \nabla H(z)^\mathsf{T} d = 0_{n_H}. \end{aligned}$$

The basic procedure is now given by Algorithm A. Usually, the termination criterion is fulfilled when a KKT point is approximated. Since $\mathcal{H}$ might not be positive definite everywhere, (QP) can exhibit multiple KKT points. In order to be able to formulate the following convergence theorem, one chooses that point that has a minimal distance to the one determined in the previous step (compare [49]).

Theorem 2.16 *Let $(z^\star, \lambda, \mu)$ be a normal KKT point of problem formulation (NLP) for which $G(z^\star) + \lambda \neq 0_{n_G}$ and sufficient second-order conditions 2.12 hold[3]. Then there exists an $\epsilon > 0$ such that for each initial guess $\left(z^{[0]}, \lambda^{[0]}, \mu^{[0]}\right) \in \mathcal{U}_\epsilon(z^\star, \lambda, \mu)$ and each sequence $\left\{\left(z^{[k]}, \lambda^{[k]}, \mu^{[k]}\right)\right\}$ generated by Algorithm A, the following statements are true:*

[3]The condition $G(z^\star) + \lambda \neq 0_{n_G}$ is also called *strict complementarity*.

(i) Algorithm A is well-defined and $\left\{\left(z^{[k]}, \lambda^{[k]}, \mu^{[k]}\right)\right\}$ *converges to* $(z^\star, \lambda, \mu)$.
(ii) The convergence rate is superlinear.
(iii) If $\nabla^2 F$, $\nabla^2 G_i$ $(i = 1, \ldots, n_G)$, *and* $\nabla^2 H_j$ $(j = 1, \ldots, n_H)$ *are locally Lipschitz continuous, the convergence rate is quadratic.*

A proof of this theorem can be found in [49].

Up to now, it is not intuitively clear how the specific choice of (QP) can be derived. To get an understanding, we consider the following exclusively equality-constrained NLP:

Problem 2.17
Find variables $z \in \mathbb{R}^{n_z}$ that

$$\begin{aligned} \text{minimize} \quad & F(z) \\ \text{subject to} \quad & H(z) = 0_{n_H}. \end{aligned}$$

The corresponding Lagrange function is given by $\bar{\mathcal{L}}(z, \mu) = F(z) + \mu^\mathsf{T} H(z)$ with multipliers $\mu \in \mathbb{R}^{n_H}$. For a regular optimal point $z^\star$, the KKT conditions read

$$K(z^\star, \mu) \coloneqq \begin{pmatrix} \nabla_z \bar{\mathcal{L}}(z^\star, \mu) \\ H(z^\star) \end{pmatrix} = 0_{n_K}.$$

Applying Newton's method to solve this system of equations leads to the iteration scheme

$$\nabla K\left(z^{[k]}, \mu^{[k]}\right) \begin{pmatrix} z^{[k+1]} - z^{[k]} \\ \mu^{[k+1]} - \mu^{[k]} \end{pmatrix} = -K\left(z^{[k]}, \mu^{[k]}\right) \tag{2.6}$$

wherein (omitting function inputs for readability)

$$\nabla K = \begin{pmatrix} \nabla^2_{zz} \bar{\mathcal{L}} & \nabla H^\mathsf{T} \\ \nabla H & 0_{n_H \times n_H} \end{pmatrix}.$$

For $d \coloneqq z^{[k+1]} - z^{[k]}$ and $\eta \coloneqq \mu^{[k+1]}$, (2.6) becomes

$$\begin{pmatrix} \nabla^2_{zz} \bar{\mathcal{L}} & \nabla H^\mathsf{T} \\ \nabla H & 0_{n_H \times n_H} \end{pmatrix} \begin{pmatrix} d \\ \eta - \mu^{[k]} \end{pmatrix} = - \begin{pmatrix} \nabla_z \bar{\mathcal{L}} \\ H \end{pmatrix},$$

which can be equivalently transformed to

$$\begin{pmatrix} \nabla^2_{zz}\bar{\mathcal{L}} & \nabla H^\mathsf{T} \\ \nabla H & 0_{n_H \times n_H} \end{pmatrix} \begin{pmatrix} d \\ \eta \end{pmatrix} = - \begin{pmatrix} \nabla F \\ H \end{pmatrix}$$

due to the definition of $\bar{\mathcal{L}}$. This is also known as *Lagrange-Newton method*. Finally, this equation is identical to the KKT conditions of (QP) if inequality constraints are omitted, which justifies the usage of this specific QP. With inequality constraints, this approach of applying Newton's method to the KKT conditions is not applicable. Practically, they are simply added to the respective QPs within SQP.

Line Search Methods

Algorithm A is the basis for many extensions and improvements. In its current form, the algorithm converges to a local minimum only if the initial guess lies within a specific region around the minimum. For globalization, a step size $\alpha \in (0, 1]$ is introduced in each iteration via

$$z^{[k+1]} = z^{[k]} + \alpha^{[k]} d^{[k]}.$$

There are several methods for its computation, for example via merit functions or filter methods. For an overview, we refer to Nocedal's and Wright's textbook [89].

Quasi-Newton Methods

The algorithm requires the computation of the Hessian of the Lagrange function $\mathcal{H} = \nabla^2_{zz}\mathcal{L}$, which can be costly on the one hand and might not always be positive definite, on the other hand. This can affect the convergence behavior. Instead, it is common to approximate the Hessian matrix in each iteration, for example using Broyden–Fletcher–Goldfarb–Shanno (BFGS) (see [89]) or Davidon-Fletcher-Powell [42] approximations.

2.2.2 Interior-Point Methods

Another widely applied class of algorithmic approaches are IP methods, to which a brief introduction is given in the following. The considerations are based on a review article by Forsgren et al. [44], while the structure is inspired by the overview in [125].

We consider the following exclusively inequality-constrained NLP:

Problem 2.18
Find variables $z \in \mathbb{R}^{n_z}$ that

$$\begin{aligned} &\text{minimize} && F(z) \\ &\text{subject to} && G(z) \geq 0_{n_G}. \end{aligned}$$

As the name already suggests, the main idea in IP methods is to iteratively approach a local optimal solution from the interior of the feasible set. This can be achieved by using so-called *barrier functions* B, which consist of the original objective function F and a *barrier term* $\mathcal{B}$, which only depends on the inequality constraints G. A famous example for such a function is the commonly used *logarithmic barrier term*[4]

$$\mathcal{B}_{\log}(z) \coloneqq -\sum_{k=1}^{n_G} \log(G_k(z)),$$

which fulfills the desired condition that for

$$\lim_{j\to\infty} z^{[j]} \in \partial S$$

with S being the feasible set, it holds for $z^{[j]} \in S^{\mathrm{o}}$ that

$$\lim_{j\to\infty} \mathcal{B}\left(z^{[j]}\right) = \infty \tag{2.7}$$

for $j \geq 0$. Herein, the interior of a set A is denoted as A^{o}, while its boundary reads ∂A. Hence, for a sequence of interior feasible points that converge to the boundary of the feasible set, the barrier term converges to infinity. The corresponding barrier function is then given as

$$\begin{aligned} B(z,\nu) &\coloneqq F(z) + \nu\mathcal{B}_{\log}(z) \\ &= F(z) - \nu\sum_{k=1}^{n_G} \log(G_k(z)) \end{aligned}$$

with $\nu > 0$ being the so-called *barrier parameter*. Next we consider the following parametric optimization problem:

[4] This is only defined for inactive points ($G_k(z) \neq 0$).

Problem 2.19
Find variables $z \in \mathbb{R}^{n_z}$ that

$$\text{minimize} \quad B(z, \nu).$$

Instead of solving the original problem, a sequence of these parametric problems is solved for a fixed value of ν, respectively. By construction, solutions that correspond to $\nu > 0$ are feasible interior points due to (2.7). The connection to the original problem from Section 2.1 can now be established by comparing the necessary optimality conditions. Since Problem 2.19 is unconstrained, they read as follows:

$$\begin{aligned} \nabla_z B(z, \nu) &= \nabla F(z) - \sum_{k=1}^{n_G} \frac{\nu}{G_k(z)} \nabla G_i(z) \\ &= 0_{n_z}. \end{aligned}$$

In a next step, we introduce the variable

$$\lambda_{\nu,i} \coloneqq \frac{\nu}{G_i(z)}$$

for an $i \in \{1, \ldots, n_G\}$, which can be transformed to

$$\lambda_{\nu,i} G_i(z) = \nu. \tag{2.8}$$

It can easily be seen that $\lambda_{\nu,i} \geq 0$ and thus, it can be interpreted as a perturbed complementarity condition, similar to (2.4). For decreasing values of ν, the original condition is finally approached.

In *primal-dual* IP methods, the newly introduced variables $\tilde{\lambda} \coloneqq \lambda_\nu \in \mathbb{R}^{n_G}$ and the original ones are treated independently: condition (2.8) is also taken into account within the problem. This leads to the necessary optimality conditions

$$\begin{pmatrix} \nabla F(z) - \nabla G(z)^\mathsf{T} \tilde{\lambda} \\ \operatorname{diag} G(z) \tilde{\lambda} - \nu_{n_G} \end{pmatrix} = 0_{n_z + n_G}$$

($\operatorname{diag} G(z)$ is the diagonal matrix of the components of $G(z)$), which can be solved by Newton's method. For additionally equality-constrained problems, Problem 2.19 is extended:

Algorithm B Primal-dual interior-point method

1: Choose $z^{[0]}$, $\tilde{\lambda}^{[0]}$, $\tilde{\mu}^{[0]}$ with $z^{[0]} \in S$, $\nu^{[0]} > 0$, set $k = 0$
2: **while** Termination criterion is not fulfilled **do**
3: Solve Problem 2.20 and obtain $z^{[k+1]}$, $\tilde{\lambda}^{[k+1]}$, $\tilde{\mu}^{[k+1]}$
4: Choose $\nu^{[k+1]}$ with $0 < \nu^{[k+1]} < \nu^{[k]}$
5: Update $k \leftarrow k + 1$
6: **end while**

Problem 2.20
Find variables $z \in \mathbb{R}^{n_z}$ that

$$\begin{aligned} &\text{minimize} && B(z, \nu) \\ &\text{subject to} && H(z) = 0_{n_H}. \end{aligned}$$

Necessary optimality conditions of this problem now read

$$\Sigma\Big(z, \tilde{\lambda}, \tilde{\mu}, \nu\Big) \coloneqq \begin{pmatrix} \nabla F(z) - \nabla G(z)^\mathsf{T} \tilde{\lambda} - \nabla H(z)^\mathsf{T} \tilde{\mu} \\ \operatorname{diag} G(z) \tilde{\lambda} - \nu_{n_G} \\ H(z) \end{pmatrix} = 0_{n_z + n_G + n_H} \tag{2.9}$$

with Lagrange multipliers $\tilde{\mu} \in \mathbb{R}^{n_H}$. The application of Newton's method leads to the following linear system of equations for one step:

$$\begin{pmatrix} \mathcal{H}\Big(z, \tilde{\lambda}, \tilde{\mu}\Big) & -\nabla G(z)^\mathsf{T} & -\nabla H(z)^\mathsf{T} \\ \Lambda_G \nabla G(z)^\mathsf{T} & G(z) & 0_{n_G \times n_z} \\ \nabla H(z) & 0_{n_H \times n_G} & 0_{n_H \times n_z} \end{pmatrix} \begin{pmatrix} \Delta z \\ \Delta \tilde{\lambda} \\ \Delta \tilde{\mu} \end{pmatrix} = -\Sigma\Big(z, \tilde{\lambda}, \tilde{\mu}, \nu\Big).$$

Herein,

$$\mathcal{H}\Big(z, \tilde{\lambda}, \tilde{\mu}\Big) = \nabla^2 F(z) - \sum_{i=1}^{n_G} \tilde{\lambda}_i \nabla^2 G_i(z) - \sum_{i=1}^{n_H} \tilde{\mu}_i \nabla^2 H_i(z)$$

denotes the corresponding Hessian matrix. Iteratively solving the system of equations (2.9) for successively decreasing values of ν yields the basic primal-dual interior-point method, summarized in Algorithm B. The algorithm iterates until a termination criterion is fulfilled, typically the check for necessary optimality conditions of Problem 2.18.

Parameter	Default value	Used value
KeepAcceptableSol	True	False
LowPassFilter	True	False
InitialLMest	True	False
ScaledKKT	True	False
Ares	[42, 41, 42, 43, 44, 41, 50]	[50]

Table 2.1: Modified WORHP configuration parameters for all presented computations.

The given algorithm only describes the basic procedure. Numerous extensions and modifications exist. In combination with an efficient implementation, IP methods have become the state-of-the-art for solving a large variety of NLP problems. State-of-the-art software tools are IPOPT (see [120]) or WORHP's IP solver, see [71].

2.2.3 The NLP Solver WORHP

In this thesis, several types of optimization problems are formulated as standard NLPs. They all have to be solved numerically with the requirement that even large problems can be solved in a reasonable time. Therefore, the NLP solver WORHP[5] [20] is used in this thesis. WORHP provides both a solver based on SQP and an IP algorithm [71]. It is designed for the numerical solution of large-scale problems, although small to medium-sized problems can also be solved efficiently.

WORHP is highly adjustable to the use-case at hand and therefore, several configuration parameters can be changed from their default values. In all the computations throughout this thesis, the following options are used if not stated otherwise: First and second-order derivatives are computed via finite differences. Features like termination heuristics or recovery strategies are disabled to maintain a consistent iteration behavior and to make numerical analyses more reliable. The changed parameters are collected in Table 2.1. For all the computations, a tolerance of 10^{-6} is used (optimality, feasibility, complementarity). These computations are executed on different computers, their specifications are listed in Table 2.2.

There exist several libraries that build upon WORHP, for example the optimal control software TransWORHP (mentioned in [69]), which is also used within this work for the numerical solution of dynamical parameter identification problems.

[5]The abbreviation "WORHP" stands for "We Optimize Really Huge Problems".

	Parameter identification	Bilevel programming
WORHP	1.11	1.15
Central processing unit	Intel® Core™ i7-4790 (4 cores, 8 threads)	AMD Ryzen™ 7 PRO 4750u (8 cores, 16 threads)
Base frequency	3.6 GHz	1.7 GHz
Random-access memory	32 GB	32 GB
Operating system	Debian 8	Ubuntu 20.04

Table 2.2: Software and hardware specifications.

2.2.4 Sparse Matrices

When applying second-order methods to solve an NLP, first and second-order derivatives (gradient of objective function, Jacobian matrix of constraints, and Hessian matrix of Lagrange function) need to be computed. For large-scale problems, this can lead to high computational efforts. However, specific problem formulations may exhibit sparsity in these derivatives. In the following, the concept of *sparse matrices* is defined. The definition we choose goes back to [51] and is, for example, also used in [48].

Definition 2.21 A matrix $A \in \mathbb{R}^{m \times n}$ with entries a_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$ is called *sparse* if $a_{ij} = 0$ for many tuples (i, j). With the number of non-zero entries denoted as n_{nz}, a sparsity measure for a matrix $A \in \mathbb{R}^{m \times n}$ is given by

$$\delta_{\mathrm{rel}}(A) \coloneqq \frac{n_{\mathrm{nz}}}{nm} \in [0, 1].$$

If $\delta_{\mathrm{rel}}(A) = 1$, the matrix is called *dense*. For values near zero, we call the matrix *sparse*.

Remark 2.22 In the special case of a Jacobian or Hessian matrix, entries that are zero independent of the evaluation point are *structural zeros*. A non-zero entry can still contain a numerical zero.

Chapter 3

Decomposition Methods

This chapter discusses the idea of decomposition: replacing complex terms in a problem formulation with more tractable ones. This involves the introduction of artificial optimization variables and constraints. To this end, the general concept is formally introduced in Section 3.1, including a literature review and initial examples. We then present two specific problem classes in which the idea of decomposition finds application. We give introductions to the field of *parameter identification in dynamical systems* and to the problem class of *bilevel programming*. Although both areas are usually distinct, we apply the concepts of Section 3.1 to them in a similar way and demonstrate their applicability. Parameter identification (Section 3.2) belongs to the field of optimization with constraints involving time-dependent, dynamical equations. In this case, decomposition strategies are well-known, but their impact on finding desired local solutions has not been fully investigated. Therefore, we introduce this field with a focus on decomposition properties. In bilevel programming (Section 3.3), evaluating the problem functions involves the solution of another parameterized optimization problem. In addition to a formal introduction to the problem class, a novel concept for decomposition is presented and analyzed. Both mathematical areas have in common that the original problem contains a complex term in the problem functions, namely the solution of an ODE or an NLP.

3.1 Concept

The focus within this work is on finding suitable formulations for a given optimization problem (compare Figure 1.1). The interplay between such a formulation and the given algorithm is important for being successful. Since we fix the algorithm choice and aim to use state-of-the-art NLP solvers, we concretize the idea of finding

suitable formulations for a given problem in this section. The general idea that will be presented in the following is *decomposition*: split up difficult terms in the problem formulation and embed those parts into the overall problem by adding artificial variables and constraints. This idea will be described more formally in the following.

3.1.1 Problem Formulations

We consider optimization problems of the standard type (as in Problem 2.2) with optimization variables $z \in \mathbb{R}^{n_z}$. Evaluating the problem-defining functions requires the "evaluation" of a subtask ψ parameterized in z. This "evaluation" can correspond to a simulation of a model, the execution of an algorithm, or the evaluation of a generic function, for example. The outcome — a motion, a final iterate, or a solution — is denoted as $w \in \mathbb{R}^{n_w}$. To emphasize that ψ represents something arbitrary, yet non-trivial or costly to evaluate, we denote it as $\psi[z]$.

In many applications, one can think of z being the variables of interest, while w corresponds to a model simulation. The model here would be parameterized in z and represents the practitioner's system of interest. When it comes to formulating an optimization problem, a naive approach would be to optimize only with respect to z, while the model is simulated in the background in each iteration the algorithm performs.

The Reduced Problem

Although the output w of ψ is required to evaluate problem-defining functions, the above-described optimization problem can be formulated only in terms of z. We call this a *reduced problem*, since the number of variables and constraints is reduced to a minimum.

Problem 3.1 (R-NLP)

Find variables $z \in \mathbb{R}^{n_z}$ that

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}(z, \psi[z]) \\ \text{subject to} \quad & \mathcal{G}(z, \psi[z]) \leq 0_{n_\mathcal{G}} \\ & \mathcal{H}(z, \psi[z]) = 0_{n_\mathcal{H}}. \end{aligned}$$

This is a problem in standard NLP form. However, every evaluation of $\mathcal{F}$, $\mathcal{G}$, or $\mathcal{H}$ requires evaluating ψ at z. Hence, the second component of each function is said to have an implicit nature and the underlying expression is called an *implicit constraint* [58], because it is hidden in such a way that the algorithm only operates on z. This is why we call z the *primary variable* and w the *secondary variable*. At first sight, this formulation offers advantages:

Reduced problem dimensions can keep the computational effort and memory requirements low.

Simplified implementation is given due to the separation of both tasks. An NLP solver can be combined with existing tools to evaluate ψ.

Anytime availability (as it is called in [15]) means that a solution of the underlying task is available at any stage of the overlaying algorithmic process. It may not appear to be necessary to solve the underlying task up to a high precision as long as the algorithm for the reduced problem has not converged. However, this can be advantageous in online frameworks, where an optimization process can be interrupted due to external factors. If this happens, one still has access to a solution of the underlying task.

However, this problem formulation exhibits disadvantages, which may not be visible in first place.

Computation time can depend heavily on ψ, although problem dimensions are reduced to a minimum. Consider the case that $\psi[z]$ represents solving a partial differential equation (PDE) that is parameterized in z. In typical scenarios, n_w (the dimension of the outcome given as discretized state variables in time and space) is much larger than n_z and solving the PDE becomes a time-consuming process. When an algorithm is run to solve this problem, the PDE would have to be solved every time the algorithm needs to evaluate $\mathcal{F}$, $\mathcal{G}$, or $\mathcal{H}$. Furthermore, in gradient-based methods, derivatives that depend on ψ have to be computed. In case finite differences are used, the computational burden can quickly become large due to numerous function evaluations.

Highly nonlinear terms may arise due to the application of ψ. Often, nonlinear problems are nonconvex, which allows them to have multiple local solutions. Converging to the global solution is a well-known challenge in local methods.

Consistency in evaluating ψ is important for the algorithm that is used to solve the problem. Consider the case that evaluating ψ corresponds to applying an iterative scheme (parameterized in z) until a given tolerance is reached. In

unfortunate cases, the scheme terminates close to the tolerance for a certain parameterization, while for another value it falls far below the tolerance. This can lead to overall differentiability issues.

Reachability restrictions can occur due to reduced image spaces. The evaluation of $\mathcal{F}$, for instance, is restricted by the image space of ψ. Of course, this is intended for the problem's solution, but it is not necessary during the solution process. In the corresponding iteration steps, an exact solution of the subtask is not required, which may allow the solver to find better values for $\mathcal{F}$. In particular, this could enable a solver to overcome local solutions which would be attained in unlucky cases of reachability restrictions[1].

Despite the many disadvantages, the reduced problem formulation can be found in many applications.

Decomposition Strategies

The concept followed in this work is *decomposition*. The idea is to introduce a set of artificial variables and constraints which represent the evaluation of ψ. Hence, it will be optimized with respect to primary and secondary variables *simultaneously*. To this end, the variable space is extended by w. An intuitive approach is to add the artificial constraint $w = \psi[z]$, which allows formulating the following problem:

Problem 3.2
Find variables $z \in \mathbb{R}^{n_z}$ and $w \in \mathbb{R}^{n_w}$ that

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}(z, w) \\ \text{subject to} \quad & \mathcal{G}(z, w) \leq 0_{n_\mathcal{G}} \\ & \mathcal{H}(z, w) = 0_{n_\mathcal{H}} \\ & \mathcal{H}_0(z, w) \coloneqq w - \psi[z] = 0_{n_w}. \end{aligned}$$

This formulation already offers advantages due to the splitting of the variables. The problem functions can be evaluated at a larger set of possible values, namely at (z, w) instead of $(z, \psi[z])$. This is possible since w is no longer dependent and will be optimized simultaneously to z. However, this formulation still has a drawback: evaluating the constraints still requires evaluating the expression ψ, which happens in every iteration of an overlaying optimization algorithm. As we assume that this is a costly procedure, these costs add up within the optimization process. Thus, it

[1] This phenomenon finds closer attention in Example 3.22.

is desirable to find a formulation which completely avoids evaluating ψ. However, the connection between z and w may not get lost. Generally, it suffices that this connection is given at the end of the optimization process — not at each problem function evaluation. In particular, we require it to be given implicitly once the optimization algorithms terminates successfully. The aim is to avoid evaluating ψ. This leads to the following task:

Find conditions $\mathcal{G}_\psi(z, w) \leq 0_{n_{\mathcal{G}_\psi}}$ and $\mathcal{H}_\psi(z, w) = 0_{n_{\mathcal{H}_\psi}}$ that make evaluating ψ redundant.

Once these conditions are found, we can formulate a *decomposed problem* in standard NLP form.

Problem 3.3 (D-NLP)

Find variables $z \in \mathbb{R}^{n_z}$ and $w \in \mathbb{R}^{n_w}$ that

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}(z, w) \\ \text{subject to} \quad & \mathcal{G}(z, w) \leq 0_{n_\mathcal{G}} \\ & \mathcal{G}_\psi(z, w) \leq 0_{n_{\mathcal{G}_\psi}} \\ & \mathcal{H}(z, w) = 0_{n_\mathcal{H}} \\ & \mathcal{H}_\psi(z, w) = 0_{n_{\mathcal{H}_\psi}}. \end{aligned}$$

With this reformulation, repeated evaluations of $\psi[z]$ are avoided and even become obsolete. The question remains under which circumstances such a reformulation can be found. Finding a general answer to this question is not trivial. For insights, we refer to the following sections.

3.1.2 Examples

The following examples shall illustrate potential fields of application for the concept of decomposition.

Symbolic Expressions

We start the list of examples on a very general level. Symbolic expressions or arbitrary terms could be decomposed into their individual parts, which are then reassembled as artificial constraints via slack variables. To realize this in an automated way, techniques from the "AI Feynman" algorithm [112] could be used. There, among other

simple mathematical operations, so-called "basic functions" are used to represent arbitrary functions via a tree graph. In addition, other properties of these functions are exploited, such as symmetries or separability (additive and multiplicative). A similar procedure is also used for automatic differentiation [53]. Here, the chain rule is applied recursively to elementary functions and operations to determine partial derivatives. This could also be used to decompose an arbitrary term to obtain a reformulation in the sense discussed here. Whether this approach is profitable, however, remains to be investigated.

Iterative Procedures

In the case where ψ represents the application of an iterative procedure, parameterized in z, the decomposition could be applied as follows: Since the number of steps required to apply this procedure depends on the input z, a consistent underlying process can be guaranteed by fixing the number of steps to N. Then, one would treat each iterated variable as an artificial optimization variable. They are then connected by artificial constraints representing the iterative scheme. Thus, the iterates are no longer determined by applying the underlying algorithm, but by the algorithmic steps of the overlying algorithm. Thus, finding a feasible solution represents the application of the underlying algorithm. Formally, we introduce the optimization variables

$$w_1, w_2, \ldots, w_N \in \mathbb{R}^{n_w}$$

which represent the sequence of iterates generated by the respective algorithm. They are linked by the applied iteration scheme and embedded into the problem as artificial constraints. For a sufficiently large N, w is (hopefully) approximated by w_N, and we obtain an almost equivalent problem formulation. This idea is pursued in Section 3.3 and analyzed using the example of bilevel programming.

Implicit Equations

Let $\psi[z]$ represent solving the implicit equation

$$\hat{H}(z, w) = 0_{n_{\hat{H}}} \tag{3.1}$$

for w. The function $\hat{H} \colon \mathbb{R}^{n_z} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_{\hat{H}}}$ is at least once continuously differentiable. Since we cannot assume that w has an explicit form, the system in (3.1) has to be solved numerically, for example using Newton's method. For an initial starting point

w_0, we have the iteration

$$w_{k+1} = w_k + \Delta w_k, \quad k = 0, 1, 2, \ldots$$

in which Δw_k is the solution of the linear system

$$\nabla_w \hat{H}(z, w_k) \Delta w_k = -\hat{H}(z, w_k).$$

This allows formulating a reduced version as in (R-NLP). For each evaluation of the problem functions, (3.1) has to be solved numerically, which can lead to several problems. First, the algorithm needs to be initialized by choosing w_0 and — since z changes its value during the optimization process — this initial guess needs to be updated adequately. One strategy is to update the initialization by the previously obtained solution. However, for large changes in z, this can be risky. Second, appropriate termination tolerances have to be chosen, and third, this inner iterative scheme has to be applied numerous times, which can be time-consuming.

The decomposed counterpart of the reduced problem uses (3.1) as an artificial constraint while treating $w \in \mathbb{R}^{n_w}$ as an optimization variable. Although problem dimensions increase, the previously-mentioned challenges become obsolete, because the explicit application of Newton's method is no longer necessary. Since the equation is treated as a problem constraint, its handling falls into the responsibility of the optimization algorithm. In the notation of (D-NLP), we have $\mathcal{H}_\psi = \hat{H}$, while $\mathcal{G}_\psi$ is not needed. One disadvantage of this decoupling is to lose the possibility to set a specific tolerance for (3.1) (the termination tolerance for Newton's method). In decomposed form, the constraint's tolerance of the optimization algorithm would also be employed for this equation. However, this issue can be addressed by adding scaling tolerances in the overlaying problem. For illustration, we consider the following example[2]:

Example 3.4: Kepler's Equation

We consider the problem of finding the date d at which Earth (in cartesian coordinates (c_x, c_y, c_z)) is closest to the given coordinate $(1, 0, 0)$. Herein, (c_x, c_y, c_z) is determined by the so-called true anomaly ϕ, which represents an angle. Its calculation, on the other hand, requires to compute the so-called eccentric anomaly E

[2] At this point the author would like to thank Dr. Matthias Knauer for providing this illustrative example. Details on the application can be found in [26].

by solving the nonlinear equation

$$E - e\sin(E) - M(d) = 0. \tag{3.2}$$

It depends on the eccentricity e and the mean anomaly M, which is a linear transformation of d. All in all, the following dependencies can be observed:

$$d \xrightarrow{\text{linear}} M \xrightarrow{\text{implicit nonlinear equation}} E \xrightarrow{\text{nonlinear}} \phi \xrightarrow{\text{nonlinear}} (c_x, c_y, c_z)$$

Thus, we can follow that

$$(c_x, c_y, c_z) = (c_x(d), c_y(d), c_z(d)).$$

Since (3.2) cannot be solved analytically, it has to be solved numerically — for example using Newton's method — and obtaining the variables (c_x, c_y, c_z) becomes even more complex: Nonlinear functions and their concatenations occur and an iteration scheme (Newton's method) has to be applied. Nevertheless, an optimization problem for the task of finding a specific date can now be formulated. It is sufficient to only optimize with respect to d, the remaining calculations are hidden to the solver:

$$\underset{d}{\text{minimize}} \quad (c_x(d) - 1)^2 + c_y(d)^2 + c_z(d)^2.$$

This is a problem in reduced, but standard NLP form and can be solved by standard techniques. However, the inner iteration scheme to solve (3.2) needs to be executed for each iterated value of the variable d.
Many of the earlier-mentioned difficulties of reduced problem formulations can be overcome by an equivalent reformulation of the problem in decomposed form (compare Problem 3.3). We introduce the optimization variable E in combination with its defining equation as artificial constraint. This has the effect that the cartesian coordinates become "less nonlinear", since the numerical solution of the nonlinear equation is no longer needed. A slightly higher-dimensional problem can be formulated:

$$\begin{aligned} \underset{d,\,E}{\text{minimize}} \quad & (c_x(E) - 1)^2 + c_y(E)^2 + c_z(E)^2 \\ \text{subject to} \quad & E - e\sin(E) - M(d) = 0. \end{aligned}$$

In this formulation, both d and E can be varied. In particular, there is no need to solve the nonlinear equation explicitly, since it is given as a constraint. To

Section notation	Kepler example
$z \in \mathbb{R}^{n_z}$	$d \in \mathbb{R}$
$w \in \mathbb{R}^{n_w}$	$E \in \mathbb{R}$
evaluate ψ	solve $E - e\sin(E) - M(d) = 0$ for E using Newton's method
$\mathcal{G}_\psi$	—
$\mathcal{H}_\psi$	$E - e\sin(E) - M(d)$

Table 3.1: Connections between Example 3.4 and the concept of decomposition.

translate this example into the notation of this section, we refer to Table 3.1. This procedure of eliminating dependencies can now be further extended, for example by considering M as an optimization variable together with its defining equation.

3.1.3 Related Work

The introduced setting can be found in very different ways in several other research works. Within this subsection, we provide some existing directions, highlight several aspects, and list interesting references.

(Partially) Reduced SQP Methods

The dependencies between w and z can be exploited algorithmically, especially in the context of Newton-based optimization methods. The starting point are so-called *reduced SQP* methods, which were initially developed for general equality-constrained NLPs [10, 21, 88], but also for inequality-constrained cases [74]. By a suitable factorization of the constraints' Jacobian matrix, the QP subproblems can be solved only on a subspace of variables, which helps to improve the overall efficiency.

Schulz and Bock [100, 101] develop so-called *Partially Reduced SQP Methods*, which exploit the circumstance that w can be seen as a by-product of z. They consider the problem

$$\begin{aligned} \underset{z,\,w}{\text{minimize}} \quad & \mathcal{F}(z, w) \\ \text{subject to} \quad & \mathcal{H}(z, w) = 0_{n_\mathcal{H}} \end{aligned}$$

and similarly to (R-NLP), they formulate a reduced version of it. Therein, it is not necessary to solve $\mathcal{H}(z, w) = 0_{n_\mathcal{H}}$ in each iteration. Instead, they suggest performing only one Newton-step and develop a modified SQP algorithm based on this. The Hessian's size within the QP subproblems is reduced to treating only the primary variables z, which can be used efficiently in optimal control problems, for example. The algorithm is extended to include inequality constraints and a convergence analysis is given.

Optimization Involving Dynamical Systems

Famous examples with reduced and decomposed problem formulations are optimization problems involving dynamical systems, for example *optimal control* or *parameter identification*, also known as *inverse problems*. Using the "first-discretize-then-optimize" approach, these problems can be formulated in standard NLP form (see Subsection 3.2.2), either by including the numerical scheme to solve the underlying dynamical system via artificial variables and constraints (decomposed formulation) or by repeatedly solving the dynamical system, for example using a *parameter-* or *control-to-state-mapping* (reduced formulation).

Biegler [9] studies optimal control problems involving algebraic-differential equations and divides the approaches into being *sequential* and *simultaneous*, while the focus is on the latter. For a direct transcription with collocation on finite elements, he points out that there is a connection between optimality conditions of the NLP and the discretized variational problem[3] and consequently, NLP and optimal control solutions can be related. Besides, he mentions that the simultaneous approach has further advantages, for example the capability to deal with instabilities or chaotic systems.

Heinkenschloss and Vicente [59] use a similar procedure to generalize optimal control problems and provide an algorithm interface. They focus on scaling and derivative computation via sensitivity and adjoint equation approaches to overcome problems arising from time discretization. Based on this, Heinkenschloss [58] focuses on the reduced problem formulation and provides algorithmic approaches such as a Conjugate Gradient method and a Gauss–Newton method. He closes with an example of optimal PDE control. Both works claim that while the decomposed formulation has its advantages, it is often impractical to use because of the increased dimensions.

[3]The "first-optimize-then-discretize" approach leads to a discretized variational problem.

Wernsing [125] examines the concept of decomposition in an application involving PDE-constrained optimization. Following a "first-discretize-then-optimize" approach, the optimization problem can either be formulated in a reduced version[4], which hides solving the PDE from the solver, or in a decomposed way[5], which embeds the discretized PDE into the problem. His numerical investigations clearly show that the decomposed problem formulation can be solved more efficiently than its reduced counterpart, while the solution quality remains similar.

Echim [35] formulates both problem formulations in the context of parameter identification involving ODEs, also using a "first-discretize-then-optimize" approach. Instead of focusing on convergence rates and efficiency comparisons, the solution quality is of interest. Robustness studies for an academic example as well as a real-world application indicate that the decomposed formulation is to be preferred, since it often leads to finding better solutions. A similar study is presented in Section 4.1 in this work.

Kaltenbacher [63] compares *all-at-once*[6] with reduced approaches for time-dependent inverse problems. She investigates the impact of these different formulations on numerical algorithms such as Landweber-Kacmarz, Landweber iteration, and the iteratively regularized Gauss–Newton method (see references in [63]). A convergence analysis in a function space setting is also presented. The numerical results on an inverse PDE problem indicate computation time benefits for the all-at-once approach.

Since this problem class is also treated within this thesis, a more sophisticated literature review is given in Section 3.2. The focus is on parameter identification for dynamical systems with an emphasis on numerical analyses.

Nonlinear Least-Squares

Schittkowski [99] considers the general nonlinear least-squares objective function

$$J(u) \coloneqq \frac{1}{2}\phi(u)^{\mathsf{T}}\phi(u)$$

[4] This is called *Nested Analysis and Design (NAND)*.
[5] *Simultaneous Analysis and Design (SAND)* is the counterpart to NAND.
[6] The term "all-at-once" refers to simultaneously optimizing with respect to both primary and secondary variables and can be understood synonymously to "decomposed".

with $\phi(u) \coloneqq (\phi_1(u), \ldots, \phi_{n_l}(u))^\mathsf{T}$ and proposes to introduce the artificial variables $v \in \mathbb{R}^{n_l}$ in combination with the artificial constraints

$$\phi(u) - v = 0_{n_l}.$$

Hence, J can be written as $J(u) = \frac{1}{2} v^\mathsf{T} v$. In case an SQP method is used, algorithmic advantages can be observed on a set of test problems. The benefit of this is a reduction in the number of function evaluations and algorithm steps.

Lifting

The idea of introducing artificial variables and constraints to formulate an equivalent problem is also called *lifting*: optimization variables are lifted into a higher-dimensional space, potentially leading to improved convergence rates as reported in several works. This is used in many related areas, among them are numerical dynamical optimization (see Section 3.2 for references), global mixed-integer nonlinear programming within the solver `BARON` [94], or Newton-type optimization methods including Newton's method itself. This is described briefly in the following.

Albersmeyer and Diehl [1] introduce *Lifted Newton Methods*. They start their study on root-finding problems $J(u) = 0_{n_u}$ with J representing the application of an algorithm with intermediate variable values

$$v_i \coloneqq \phi_i(u, v_1, v_2, \ldots, v_{i-1}) \text{ for } i = 1, 2, \ldots, m.$$

Hence, $J(u)$ can be written as $\phi_J(u, v_1, v_2, \ldots, v_m)$. Instead of solving the original problem, they consider the lifted system of equations

$$K(u, v) = \begin{pmatrix} \phi_1(u) - v_1 \\ \phi_2(u, v_1) - v_2 \\ \vdots \\ \phi_m(u, v_1, \ldots, v_{m-1}) - v_m \\ \phi_J(u, v_1, \ldots, v_m) \end{pmatrix}$$

representing the algorithm steps. Newton's method is analyzed for this case and an approach for an efficient derivative computation is given. The lifted system can be solved efficiently using structure exploitation, which is called "condensing" therein. They show that under specific assumptions, one can expect faster local convergence to the optimal solution while having fewer additional costs per iteration. This idea

is extended to the class of Newton-type optimization via Gauss–Newton methods and SQP, for which they provide lifted versions and demonstrate their benefits on large-scale example problems. Lifted Newton Methods can be interpreted as a generalization of direct multiple shooting methods known from optimal control or parameter identification. The theoretical focus within their work lies in determining the convergence rate for a simplified setting, while they observe impressive improvements on complex examples.

The approach proposed in this work differs from Lifted Newton Methods in such a way that the algorithm to solve the lifted problem remains as it is. On the one hand, this reduces the implementation effort. The design of adapted solvers is avoided, existing solvers can be used. On the other hand, strategies for improving the efficiency, such as condensing, are not considered. However, this aspect is addressed from the perspective of sparsity exploitation. They also mention that lifted schemes benefit from more initialization freedom, but the possible consequences find less attention. In problems with multiple local solutions, an adequate initialization can strongly influence which solution is found. This aspect is given more attention in this work, especially in Section 4.1.

Zach and Bourmaud [129] study lifting approaches within robust cost optimization, in which one aims to minimize functions $\Gamma(\theta) = \sum_{i=1}^{N} \gamma(\|\phi_i(\theta)\|)$ with respect to $\theta \in \mathbb{R}^{n_p}$, while the functions ϕ_i represent residual functions; γ is a so-called robust kernel function. These types of objective functions usually possess many local minima due to their high nonconvexity. To escape poor local solutions, they introduce lifting variables v_i, that can be seen as confidence weights, via $\sum_{i=1}^{N} \left(a(v_i) \frac{\|\phi_i(\theta)\|^2}{2} + \delta(a(v_i)) \right)$ with $a(v) \coloneqq v^2$ (for example) and δ being convex and increasing. By initializing them with large values, the objective function is smoothed. In their lifting approach, they optimize simultaneously with respect to θ and v_i. Hence, they embed the original objective function in a higher-dimensional search space. A similar approach is also investigated in this work. In Section 4.6, this idea is applied to homotopy continuation for parameter identification in dynamical systems.

Alternating Direction Method of Multipliers

This method is originally designed for convex and separable objective functions $\mathcal{F}(z, w) = \mathcal{F}_1(z) + \mathcal{F}_2(w)$ and linear constraints $Az + Bw = c$ (with matrices A and B and a vector c). The idea is to minimize the (augmented) Lagrange function with respect to z and w in an alternating way while updating the dual variables. Under some mild assumptions, this procedure is guaranteed to converge to the optimal solution (compare [17]). It was also successfully applied to nonconvex problems (see

for example [121] and references therein). The challenge is to reformulate a given problem in the above way taking the separability of $\mathcal{F}$ into account. The approach used in this work differs from the method due to the simultaneous optimization with respect to z and w.

All in all, the idea of adding artificial variables and constraints finds application in many optimization-related research areas.

3.2 Decomposition in Parameter Identification for Dynamical Systems

For many technical processes from a wide variety of fields, general models exist that describe the dynamical behavior. The structure of these models is usually known, as they are often based on physical laws. To ensure an accurate simulation or control of the underlying system, the model must be adapted to the specific application. This is usually done by calibrating the model parameters using measurements from the system.

In this section, the dynamical systems of consideration are given as ODEs. These models cover a wide range of applications from many fields, for example biology [109], chemistry [92], or mechanics [75]. The aim is to identify scalar parameters within the model equations that influence the model behavior. Roughly speaking, this is achieved by adjusting these parameters until the corresponding model output matches given system measurements best.

This field is challenging in several disciplines. Without appropriate measurements, the parameter identification will not succeed. *Optimal experimental design* can be used to generate experiments that can lead to identifying relevant model parameters, see [7] for instance.

Another crucial decision is whether to use the so-called direct or the indirect approach. In the direct one, the choice of the optimization algorithm is an important factor and subject of many research articles. Possible algorithms are Levenberg-Marquardt [76] or Gauss–Newton algorithms [13] for nonlinear least-squares problems, SQP or IP methods for general purpose NLPs (as introduced in Section 2.2 and used within Chapter 4).

With regard to Section 3.1, evaluating the underlying complex term means numerically solving the ODE. Hence, the choice of the integration scheme is also an

important decision, which often depends on the specific problem at hand. While state-of-the-art integration schemes like Runge-Kutta methods often perform sufficiently well, other types like variational integrators[7] (see [39] for example) also have their advantages.

Furthermore, the way the objective function is formulated can have a strong influence on the computational result. Typically, least-squares formulations (ℓ_2 functions) are used. There are also other possibilities, such as the ℓ_1 objective function studied by Kostina [70]. She shows that this type of objective function is very suitable for measurements with outliers. Furthermore, it is shown that quasi-Newton algorithms (such as BFGS approximations in SQP) are not suitable for parameter estimation and introduce "stable solutions", which are characterized by continuous and differentiable dependence on perturbations, for example measurement perturbations. So-called "unstable solutions" are recognized by a slow convergence behavior of the Gauss–Newton method they use.

This section focuses on NLP formulations of a parameter identification problem and their impacts on finding solutions.

The NLP formulation can have a decisive influence on the result of the parameter identification task. Although different problem formulations may equivalently represent the same original problem, they do show a different algorithmic behavior, and thus some formulations are more suitable than others. In the following, we will investigate some above-mentioned aspects in more detail. In particular, we will focus on the latter aspect and aim to answer the following question: how can the original parameter identification problem be reformulated so that an algorithm reliably converges to the desired solution?

Remark 3.5 The studied problem belongs to the class of *inverse problems*: given an observed output of a system, one searches for the "cause". Herein, the output is given in terms of measurements. The unknown parameters in the ODE model are the cause. Typically, inverse problems are *ill-posed*, which means that one of the properties

- "existence",
- "uniqueness",
- or "continuous dependency on the measurements"

[7] Variational integrators can also be of Runge-Kutta type.

is violated[8]. To obtain a *well-posed* problem, *regularization* is a common step, which means that a term is added to the objective function to *smooth* it. By this, an early convergence to local minima is aimed to be avoided. However, regularizing a problem requires at least some vague knowledge of the desired parameter values, and also introduces some bias into the problem.

In the following, the problem of interest is formally introduced.

3.2.1 Original Problem Formulation

For a function $s\colon \mathbb{R} \times \mathbb{R}^{n_q} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_q}$, we consider the dynamical system

$$\dot{q}(t) = s(t, q(t), p) \tag{3.3}$$

with t representing the time variable ranging in the time interval $[t_a, t_b]$ with $t_a < t_b$. In the field of parameter identification, $t_a = 0$ is a common choice. A *state* at a specific time point is denoted by $q(t)$. We assume s to be Lipschitz continuous in q, such that there exist unique solutions $q(t;\, p, q_a) \in \mathbb{R}^{n_q}$ for all initial values $q_a \in \mathbb{R}^{n_q}$ and all parameter values $p \in \mathbb{R}^{n_p}$ (see [56], among others).

Remark 3.6 The term $q(t;\, \omega)$ denotes the explicit dependency on the time variable t and an implicit dependency on an arbitrary $\omega \in \mathbb{R}^{n_w}$, for example due to a parameterization of the corresponding dynamics.

Remark 3.7 It is also common to consider systems of the form

$$\dot{q}(t) = s(t, q(t), u(t), p)$$

with an additional input function $u\colon \mathbb{R} \to \mathbb{R}^{n_u}$. Usually, it represents a piecewise continuous control term. In parameter identification tasks, measurements of u would influence the dynamics. For simplicity, however, we omit this dependency in the following.

We assume that this model is able to accurately represent a real-world time-dependent system. This system operates with a nominal parameter vector $p^\star \in \mathbb{R}^{n_p}$, which is unknown to a practitioner. To simulate the system or for control purposes, this parameter needs to be identified. For this, one needs to obtain measurements of

[8] This definition goes back to Hadamard [55].

system states that correspond to this parameter choice. From an application point of view, one can only expect to have a finite number of measurement points at specific time points. One can think of sensors that measure only at a given frequency. We explicitly allow that not all components of the states have to be measured. Indeed, this is a typical circumstance. It can be caused by missing sensors, for example, and can restrict the practitioner to using certain algorithms[9]. In case of incomplete state information, we collect the indices of the measured state variables in the set

$$\mathcal{J} \coloneqq \{j \in \{1, \dots, n_q\} \colon \text{there exist measurements for state variable } q_j\}$$

and $q|_{\mathcal{J}}$ describes those components of q for which measurements exist. Its complement — the index set of unmeasured state variables — is denoted as $\mathcal{J}^{\mathsf{c}}$ and analogously, $q|_{\mathcal{J}^{\mathsf{c}}}$ are the unmeasured states. We assume that the measurements are given at N_{m} discrete time points

$$t_a = \bar{t}_1 < \dots < \bar{t}_{N_{\mathrm{m}}} = t_b$$

for each measured state variable, and denote them as

$$\bar{q}_k = \left(\bar{q}_k^{[\bar{t}_1]}, \dots, \bar{q}_k^{[\bar{t}_{N_{\mathrm{m}}}]}\right)^{\mathsf{T}} \in \mathbb{R}^{N_{\mathrm{m}}}$$

for $k \in \mathcal{J}$. It is inevitable that measurements are corrupted by noise and therefore, one cannot assume that the model is able to reproduce the measurements in an exact way. Thus, we cannot expect to find model parameters p that fulfill

$$q|_{\mathcal{J}}(t_i; p) = \bar{q}^{[\bar{t}_i]}$$

for $i \in \{\bar{t}_1, \dots, \bar{t}_{N_{\mathrm{m}}}\}$. This is why one wants to find those parameter values that lead to a solution of the ODE (3.3) on $[t_a, t_b]$ which approximates the measurements in the *best* way. The phrase "best" implies the search for those parameters that optimize a given criterion. This task can be formulated as minimizing the term[10]

$$\frac{1}{2} \sum_{i=1}^{N_{\mathrm{m}}} \left\| q|_{\mathcal{J}}(\bar{t}_i; p) - \bar{q}^{[\bar{t}_i]} \right\|_2^2,$$

while it is required that the state q is a solution of the dynamical system (3.3) for $t \in [t_a, t_b]$.

[9] It is also common to consider indirectly measured states, for example measuring only the product of two states instead of the corresponding factors.

[10] Scaling by $\frac{1}{2}$ is a common choice in the literature.

Remark 3.8 One can expect that a single measurement point for a specific state $k \in \mathcal{J}$ at a given time point $\bar{q}_k^{[\bar{t}_i]}$ is affected by an additive error ϵ that is independent and normally distributed with zero mean and variance σ^2 (usually written as $\epsilon \sim \mathcal{N}(0, \sigma^2)$). Hence, the objective function can be formulated as

$$\frac{1}{2} \sum_{i=1}^{N_\mathrm{m}} \frac{1}{\sigma_i^2} \left\| q|_{\mathcal{J}}(\bar{t}_i;\, p) - \bar{q}^{[\bar{t}_i]} \right\|_2^2.$$

This choice is motivated by the fact that its global minimizer corresponds to a global maximizer of the *likelihood function*, hence to a parameter set which is the *most likely one* to lead to the given measurements. A detailed derivation is given in [105], for instance. However, we do not follow this approach and assume that the variances are constant.

Now, we have collected all ingredients to formulate the original problem, which we call *Original Dynamical Parameter Identification Problem (O-DPIP)*:

Problem 3.9 (O-DPIP)

Find model parameters $p \in \mathbb{R}^{n_p}$ and piecewise continuously differentiable continuous functions q that

$$\begin{aligned} &\text{minimize} && \frac{1}{2} \sum_{i=1}^{N_\mathrm{m}} \left\| q|_{\mathcal{J}}(\bar{t}_i;\, p) - \bar{q}^{[\bar{t}_i]} \right\|_2^2 \\ &\text{subject to} && \dot{q}(t) = s(t, q(t), p) \quad \text{for } t \in [t_a, t_b]. \end{aligned}$$

Due to the optimization with respect to q, (O-DPIP) belongs to the class of *infinite-dimensional optimization problems*.

Before continuing, let us take a look at an academic example.

Example 3.10

We consider the technological system of a pendulum. A point mass is connected to a rod of length $L = 1\,\mathrm{m}$, and we assume that the standard gravity is $g = 9.81\,\mathrm{m/s^2}$. The angle between the rod position at rest and the observed rod position is denoted as θ (see Figure 3.1). Equations of motion can be derived by applying Newton's

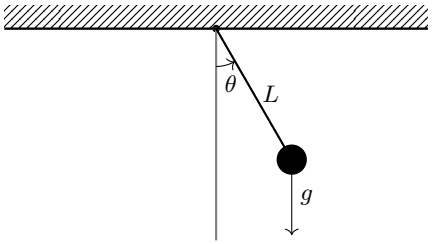


Figure 3.1: Sketch of a mathematical pendulum.

Second Law and are given by

$$\ddot{\theta}(t) + \frac{g}{L}\sin(\theta(t)) = 0.$$

This can be transformed to the first-order system of coupled ODEs

$$\begin{aligned}\dot{q}_1(t) &= q_2(t),\\ \dot{q}_2(t) &= -\frac{g}{L}\sin(q_1(t))\end{aligned}$$

for $t \in [t_a, t_b]$, $q_1 = \theta$, and $q_2 = \dot{\theta}$. We assume that the rod length L is the only unknown and that measurements $\bar{q}_1 \in \mathbb{R}^{N_\mathrm{m}}$ (see Figure 3.2) are given at $N_\mathrm{m} = 101$ equidistant points in the time interval $[0, 10]$. This corresponds to a measurement step size of $\bar{h} = 0.1$. These measurements are created artificially by integrating the ODE using the classical Runge-Kutta scheme with initial values $(q_{1a}, q_{2a}) = (\frac{\pi}{6}, 0)$ and the nominal parameter $L = L^\star = 1$. They are perturbed as described above, using a variance value of $\sigma^2 = 0.01$. The corresponding original problem has the following form:

$$\begin{aligned}\underset{L,\,q_1,\,q_2}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N_\mathrm{m}}\Big(q_1(\bar{t}_i;\,z) - \bar{q}_1^{[\bar{t}_i]}\Big)^2 \\ \text{subject to} \quad & \dot{q}_1(t) = q_2(t), && t \in [0, 10]\\ & \dot{q}_2(t) = -\frac{g}{L}\sin(q_1(t)), && t \in [0, 10].\end{aligned}$$

Although the pendulum belongs to one of the easier nonlinear systems, it reveals interesting and representative phenomena in the field of numerical parameter identification. It is referred to repeatedly in the remainder of this section, while it is examined in more detail in Section 4.1.

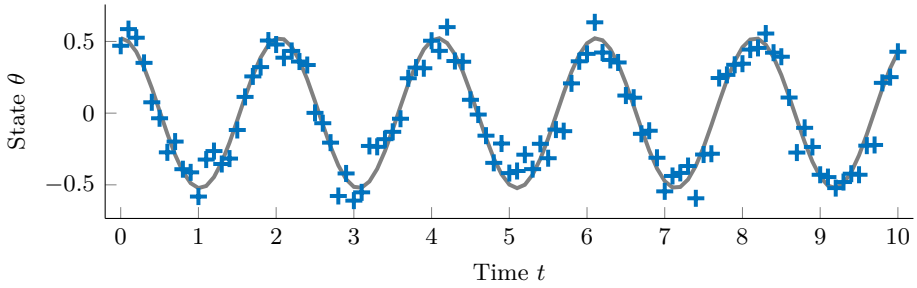


Figure 3.2: Perturbed measurement points of the pendulum system (Example 3.10) and the unperturbed trajectory.

For the solution of optimization problems involving ODEs, two approaches are established: *indirect* and *direct* methods. In the indirect approach, necessary optimality conditions are derived for (O-DPIP). This results in a boundary value problem, which is then solved numerically, for example with shooting or collocation methods. The direct approach uses a discretized, finite-dimensional version of (O-DPIP), from which KKT points are computed numerically. This approach follows the strategy to *first discretize, then optimize*: In a first step, a time-based discretization is applied and the time interval is represented by a set of discrete time points $\{t_i\}_{i=1}^{N_\mathrm{d}}$. Accordingly, the state variable $q(\cdot)$ is approximated by a set of discrete values $\tilde{q}^{[t_i]}$, $i = 1, \ldots, N_\mathrm{d}$. We call such a transformation of an infinite-dimensional problem into the standard NLP form *transcription*.

This is the point where our investigations begin. Problem 3.9 can be reformulated (or transcribed) in several, possibly equivalent ways, which can all be passed to an NLP solver. From the perspective of a practitioner, the aim should be to find a formulation which has beneficial properties, for example stable iteration processes, moderate computation times, or robustness with respect to solver initializations. The latter aspect addresses the ability to finding desired local solutions reliably. In the following, we present several formulations of the original problem and discuss their advantages and drawbacks, accompanied by examples.

3.2.2 Transcription

In this work, the direct approach is used to solve parameter identification problems involving dynamical systems. In regard to the concept of decomposition (as introduced in Subsection 3.1.1), we are interested in the numerical behavior of different

transcription techniques, which are categorized as being *sequential* or *simultaneous*, see [9] for an overview.

The former approach is characterized by only optimizing with respect to the primary variables p and q_a. The initial value q_a is here also considered to be a primary variable, since the solution of the ODE uniquely depends on p and q_a. The secondary state variables are then obtained by simulating the model using the primary variables, which is numerically solving the ODE. In regard to the definitions in Subsection 3.1.1, this ansatz corresponds to the *reduced formulation* which leads to an NLP of small dimensions.

In the latter approach, the secondary and primary variables are decoupled and optimized at the same time. Their dependency can now be found in additional constraints, which makes an external integration redundant. However, problem dimensions increase. In the wording of Subsection 3.1.1, this corresponds to a *decomposed formulation.*

A third option is given by the well-established direct multiple shooting approach, which combines ideas from both mentioned methods.

In the following, we investigate the single shooting approach (sequential, Subsection 3.2.3), a direct multiple shooting method (mixed sequential and simultaneous, Subsection 3.2.4), and the idea of full discretization (simultaneous, Subsection 3.2.5). What all of these approaches have in common is that a solution of the underlying ODE has to be approximated by numerical integration, since this solution will be used in the objective function. A brief introduction to the subject is given in the following:

Remark 3.11 The approximate solution of initial value problems (IVPs) of the form

$$\begin{aligned} \dot{q}(t) &= \tilde{s}(t, q(t)), \\ q(t_a) &= q_a \end{aligned}$$

is called *numerical integration.* For the majority of IVPs, an analytical expression for a solution is not available. Therefore, numerical routines must be applied. The general idea is to approximate the solution $q(t)$ on a time interval $[t_a, t_b]$ by a set of points $\tilde{q}^{[t_n]}$, $n = 1, \ldots, N_\mathrm{d}$. For a given step size $h > 0$, *explicit Runge-Kutta methods* are defined by the iteration rule

$$\tilde{q}^{[t_{n+1}]} = \tilde{q}^{[t_n]} + h \sum_{i=1}^{s} b_i k_i,$$

in which b_i are real coefficients and the integer s denotes the number of stages. These are given by

$$\begin{aligned}
k_1 &= \tilde{s}(t_n, \tilde{q}^{[t_n]}),\\
k_2 &= \tilde{s}(t_n + c_2 h, \tilde{q}^{[t_n]} + h(a_{21} k_1)),\\
k_3 &= \tilde{s}(t_n + c_3 h, \tilde{q}^{[t_n]} + h(a_{31} k_1 + a_{32} k_2)),\\
&\vdots\\
k_s &= \tilde{s}(t_n + c_s h, \tilde{q}^{[t_n]} + h(a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{s,s-1} k_{s-1}))
\end{aligned}$$

with nodes c_i, $i = 2, 3, \ldots, s$, and coefficients a_{ij} for $1 \leq j < i \leq s$. If the coefficient matrix a_{ij} is not in lower-triangular form, the method is called implicit and at each step, a system of algebraic equations has to be solved. Famous schemes are *Explicit Euler*, *Classical Runge-Kutta Method*, or *Trapezoidal Rule*. We summarize both explicit and implicit methods in the iteration rule

$$\tilde{q}^{[t_{n+1}]} = \tilde{q}^{[t_n]} + h\phi\Big(\tilde{q}^{[t_n]}, \tilde{q}^{[t_{n+1}]}\Big)$$

with $\phi(\cdot, \cdot)$ being the corresponding *incremental function*.

In the following, several transcription methods are presented. Since all of them involve numerical integration, a common notation is used. The number of discretization points is N_d and the equidistant grid

$$t_a = t_1 < t_2 < \cdots < t_{N_\mathrm{d}} = t_b$$

is used. The corresponding step size between two consecutive discrete time points amounts to

$$h \coloneqq \frac{t_b - t_a}{N_\mathrm{d} - 1}.$$

The integration scheme applied to numerically solve the ODE can be represented by solving the system of equations

$$H_\mathrm{ODE}(\tilde{q}, p) \coloneqq \begin{pmatrix} \tilde{q}^{[t_2]} - \tilde{q}^{[t_1]} - h\phi\Big(\tilde{q}^{[t_1]}, \tilde{q}^{[t_2]}, p\Big) \\ \vdots \\ \tilde{q}^{[t_{N_\mathrm{d}}]} - \tilde{q}^{[t_{N_\mathrm{d}-1}]} - h\phi\Big(\tilde{q}^{[t_{N_\mathrm{d}-1}]}, \tilde{q}^{[t_{N_\mathrm{d}}]}, p\Big) \end{pmatrix} = 0_{(N_\mathrm{d}-1)n_q} \tag{3.4}$$

	Continuous	Discretized	Measurements
Time interval	$[t_a, t_b]$	$\{t_i\}_{i=1}^{N_\mathrm{d}}$	$\{\bar{t}_i\}_{i=1}^{N_\mathrm{m}}$
State	$q(t)$	$\tilde{q}^{[t]}$	$\bar{q}^{[\bar{t}]}$
Step size	—	$\tilde{h} \coloneqq \frac{t_b - t_a}{N_\mathrm{d} - 1}$	$\bar{h} \coloneqq \frac{t_b - t_a}{N_\mathrm{m} - 1}$

Table 3.2: Notation for different state representations.

with $\tilde{q}^{[t_1]}$ $(= q_a)$ being the initial value, which needs to be given. Its solution (in case there exists one) represents a discretized trajectory at different time points t. Usually, it is uniquely determined by the initial state and parameter values. To refer to it, we denote it by $\tilde{q}(t;\, q_a, p)$ or $\tilde{q}\Big(t;\, \tilde{q}^{[t_1]}, p\Big)$ whenever appropriate.

Remark 3.12 In general, (3.4) can be solved by applying Newton's method. If the integration scheme is explicit and $\phi\Big(\tilde{q}^{[t_k]}, \tilde{q}^{[t_{k+1}]}, p\Big)$ does not depend on $\tilde{q}^{[t_{k+1}]}$ for each $k = 1, \ldots, N_\mathrm{d} - 1$, the system can be solved efficiently by sequential evaluations. This is the case, for example, with the Explicit Euler method.

We assume that measurements are given at N_m equidistant time points $\bar{t}_i$ using a constant step size $\bar{h}$. To simplify the setting, we restrict ourselves to cases with $N_\mathrm{m} = N_\mathrm{d}$. An overview of the used notation is given in Table 3.2.

Remark 3.13 Often, there are a lot more measurements than discretized state values $(N_\mathrm{m} \gg N_\mathrm{d})$ and one cannot expect that the corresponding time points overlap each other. In that case, one would still sum over all measurement points within the objective function and the discretized states would have to be interpolated. Alternatively, one can design the numerical integration in such a way that the discrete trajectory is given at the specific measurements time points.

3.2.3 Reduced Formulation

Since a solution of the dynamical system (3.3) is uniquely determined by its initial value q_a and the parameter value p, a reduced version of Problem 3.9 can be formulated that corresponds to (R-NLP). In regard to this, we exclusively optimize for the above-mentioned primary variables. Since the discrete trajectory needs to be accessed within the objective function, the main idea is, roughly speaking, to solve the ODE whenever needed.

The above description leads to an unconstrained optimization problem in NLP form. The *Reduced Dynamical Parameter Identification Problem (R-DPIP)* can be formulated as follows:

Problem 3.14 (R-DPIP)

Find initial state values $q_a \in \mathbb{R}^{n_q}$ and model parameters $p \in \mathbb{R}^{n_p}$ that

$$\text{minimize} \quad F_{\mathrm{R}}(q_a, p) \coloneqq \frac{1}{N_{\mathrm{d}}} \sum_{i=1}^{N_{\mathrm{d}}} \left\| \tilde{q}|_{\mathcal{J}}(t_i;\, q_a, p) - \bar{q}^{[t_i]} \right\|_2^2.$$

At first sight, the presented formulation offers practical advantages (compare Subsection 3.1.1). The number of optimization variables amounts to $n_q + n_p$, which is usually reasonably low. Further, there do not exist any constraints, the problem belongs to the class of unconstrained optimization. Dependent on the problem and especially the ODE at hand, the computation time can, however, still be large due to an inappropriate choice of the ODE solver. Especially for high-precision (non-adaptive) integration schemes (small step sizes, for example), the main computation time is taken over by repeatedly solving the ODE. This hands over to the fact that inner and outer iterations are separated: the implementation effort is minimal, since existing methods can be used. In each call of the objective function, (3.3) needs to be solved numerically. A further note is that the objective function is usually highly nonlinear due to the inner numerical integration. Nonconvexity is usually the consequence, which causes the problem to have multiple local minima. In the following, a typical example of this phenomenon is illustrated:

Example 3.15

We return to Example 3.10 and investigate the shape of the objective function. For this we use an Explicit Euler scheme with step size $h = 0.005$, which corresponds to $N_{\mathrm{d}} = 2001$ discretization points. To obtain an idea of the objective function's shape, we evaluate it at many values of L, ranging from $L_{\mathrm{low}} = 0.2$ to $L_{\mathrm{upp}} = 2$. Note that it is therefore necessary to fix the initial values to their nominal values (see Example 3.10). The shape is shown in Figure 3.3. As expected, the objective function attains its global minimum at $L^\star \approx 1$ (we cannot expect to find an accurate value due to the error in the numerical integration and the measurement noise). However, several other local minima occur with similar objective function values. In particular, we can observe a very "flat local minimum" in the region $(1.6, 1.8)$. Here, small perturbations in the initial parameters would not cause the solver to compute a different solution, which is in contrast to

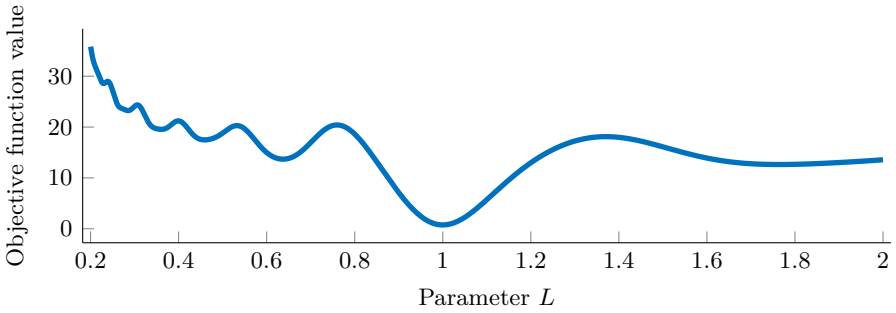


Figure 3.3: Shape of the objective function of the reduced formulation for the pendulum example 3.15 with fixed initial values.

the minima in the region $(0.2, 0.4)$. This aspect is considered more closely in Section 4.1. The shape of the objective function makes the parameter identification a challenging task for local methods. Depending on an initial guess for L, the nearest local minimum is assumed to be found by a solver.

Remark 3.16 In the jargon of inverse problems, the generation of data is called the *forward problem*, while the following parameter identification with this data is called *inverse problem*. If in both problems the same configurations are used, it is called an *inverse crime*[11] and should be avoided. For a better imitation of real-world problems, we circumvent such an inverse crime by using different integration schemes in the measurement generation in combination with varying step sizes and artificial measurement perturbations.

The previous example has revealed that the integration of the dynamics for some parameter values may lead to undesired values of the objective function. In fact, it is even possible that an ODE cannot be solved for certain parameter values and a numerical solution by integration techniques is no longer possible. Consequently, this would lead to a failure of an NLP solver since the objective function could not be evaluated. This behavior is presented in the following example (compare [11]):

Example 3.17

We consider a *Lotka-Volterra* model, which describes the behavior of an ecological system consisting of predator q_1 and prey q_2. For time t, the dynamical behavior

[11] Rainer Kress [24] is usually associated with this term.

is given by

$$\dot{q}_1(t) = -k_1 q_1(t) + k_2 q_1(t) q_2(t),$$
$$\dot{q}_2(t) = k_3 q_2(t) - k_4 q_1(t) q_2(t).$$

For parameter values $(k_1, k_2, k_3, k_4) = (0.5, 0.5, 0.5, -0.2)$ in combination with initial values $(q_1(0), q_2(0)) = (0.4, 1)$, the solution exhibits a singularity at $\hat{t} \approx 3.3$ and thus, the system cannot be integrated, which lets the reduced formulation fail for times going beyond $\hat{t}$.

Initialization of (R-DPIP)

If there exist measurements for all state variables (and especially at $t = t_a$), it is reasonable to use $q_{a\text{ini}} = \bar{q}^{[\bar{t}_1]}$. For unmeasured states, one needs to "guess" suitable values, which can be based on expert knowledge, for instance. It is much more difficult to guess suitable parameter initializations. If a parameter has a physical interpretation, one can focus on a specific area in which the optimal parameter should be located. However, one always needs to include inaccuracies of the integrator or in the measurements which can alter the expected optimal value. For non-physical parameters, there is often no other choice than to guess.

Computation of Derivatives

A natural and often used idea to solve (R-DPIP) is to use gradient-based approaches, such as Newton-type methods. They require computing the derivative of the objective function $F_{\mathrm{R}}(z)$ with respect to the optimization variables $z \coloneqq (q_a, p)$. Since F_{R} contains the ODE's numerical solution $\tilde{q}$, which depends on q_a and p, the derivatives

$$\frac{\partial \tilde{q}}{\partial q_a} \text{ and } \frac{\partial \tilde{q}}{\partial p}$$

have to be computed as well. According to [91], there are mainly three approaches:

(i) *External differentiation* is the application of finite difference methods to compute the desired derivatives, for example via

$$\frac{\partial \tilde{q}(t;\, q_a, p)}{\partial p_i} \approx \frac{1}{\tilde{h}} \Big(\tilde{q}\Big(t;\, q_a, p + \tilde{h} e_i\Big) - \tilde{q}(t;\, q_a, p) \Big).$$

Herein, $\tilde{h}$ is a small step size and e_i denotes the i-th unit vector.

(ii) *Internal differentiation* is a differentiation of the applied integration scheme. We refer to [11] or [12] for details.

(iii) Solving *sensitivity equations* simultaneously to the original system is the third option. The sensitivity $\frac{\partial q}{\partial z}(t) =: S(t;\, z)$ can be obtained by solving the IVP

$$\begin{aligned} \dot{S}(t;\, z) &= \frac{\partial s}{\partial q}(t, q(t), p) S(t;\, z) + \frac{\partial s}{\partial z}(t, q(t), p) \\ S(t_a;\, z) &= [1_{n_q \times n_q}, 0_{n_q \times n_p}]. \end{aligned}$$

Recall that s is the right-hand side of the original ODE from (3.3).

(R-DPIP) is in standard NLP form (as in Problem 2.2) and can therefore be solved by state-of-the-art iterative gradient-based solvers. The derivatives $\nabla_z F_{\mathrm{R}}(z) \in \mathbb{R}^{n_q+n_p}$ and $\nabla^2_{zz} F_{\mathrm{R}}(z) \in \mathbb{R}^{(n_q+n_p)\times(n_q+n_p)}$ are usually dense.

Literature Review

The reduced formulation is known under several names. The term *initial value approach* emphasizes that a solution of the ODE is solely dependent on its initial values (and the unknown parameters, of course). The approach is often referred to as the *single shooting* method. This terminology stresses that for an evaluation of the objective function, it is necessary to integrate the system of ODEs once for the current parameters and initial values. Shooting here refers to integrating the model equations. Since the integration is *hidden* to the optimization solver, it is also often called being *black-box*.

Despite its many well-known disadvantages, the reduced formulation is still widely used in many works on parameter identification problems. However, it is often recognized that it has numerical difficulties. Bock [12] lists and demonstrates many pitfalls of the single shooting method in parameter identification (and suggests the use of a *multiple shooting* method, see Subsection 3.2.4). Peifer and Timmer [91] compare the initial value approach with a multiple shooting method. Their numerical results on examples from biochemical processes demonstrate that the problems are rarely solvable, and when they are, undesirable solutions are frequently computed. Ribeiro and Aguirre [93] also compare single and multiple shooting with a focus on the objective function. However, they test their methods on the logistic map instead of using an ODE, but again the results show that many local minima occur and that more advanced problem formulations (such as multiple shooting) tend to produce more desirable solutions. Michalik et al. [77] develop an extension, the so-called "Incremental single shooting", which solves the parameter identification problem on a

small time and data interval and increases it successively. Vyasarayani et al. [117–119] investigate a homotopy continuation method for the single shooting approach[12].

3.2.4 Direct Multiple Shooting

The idea of direct multiple shooting is to combine the external numerical integration and a higher-dimensional search space. This is achieved by splitting the time interval into smaller parts with the introduction of so-called *shooting nodes*. These nodes serve as initial values from which the numerical integration is restarted. Thus, the ODE needs to be solved multiple times, but on shorter time intervals. In the optimization problem, the values at these nodes are treated as additional variables that need to be optimized. To make sure that one solves the original problem, so-called *continuity constraints* are introduced which ensure that initial values from a sub-interval match with final values from the prior sub-interval at a solution point.

Formally, we introduce N_s equidistantly distributed shooting nodes τ on the interval $[t_a, t_b]$ that fulfill

$$t_a = \tau_1 < \tau_2 < \cdots < \tau_{N_\mathrm{s}} < t_b$$

and — for simplicity — we demand each of them to correspond to one of the time points used in the discretization for the numerical integration. Hence, it shall hold that

$$\tau_i \in \{t_1, \ldots, t_{N_\mathrm{d}-1}\}$$

for $i = 1, \ldots, N_\mathrm{s}$ (in analogy to the choice of the measurement time points). Note that we can avoid setting the last time point as a shooting node, which is, for instance, not the case in optimal control problems due to the existence of control terms that may also have an effect there. At each of these nodes, the ODE is integrated up to the next one. Thus, the length of each time interval depends on the number of shooting nodes: one node ($\tau_1 = t_a$) corresponds to the reduced formulation (as in Subsection 3.2.3), two nodes correspond to two intervals of halved length, and so on. In the resulting problem formulation, the shooting node values

$$\tilde{q}^{[\tau_1]}, \tilde{q}^{[\tau_2]}, \ldots, \tilde{q}^{[\tau_{N_\mathrm{s}}]}$$

[12] This approach is extended by the author in Section 4.6.

(which we collect as $\tilde{q}^{[\cdot]}$) are then treated as optimization variables. To obtain a continuous state trajectory (a solution of the original ODE) at the end of the optimization process, constraints need to be introduced which guarantee that final values from one sub-interval coincide with initial values from the following sub-interval. Hence, it should hold that

$$\tilde{q}\Big(\tau_{j+1};\, \tilde{q}^{[\tau_j]}, p\Big) = \tilde{q}^{[\tau_{j+1}]}$$

for $j = 1, \ldots, N_\mathrm{s} - 1$. These are usually highly nonlinear constraints due to the (hidden) numerical integration which is necessary to obtain $\tilde{q}\Big(\tau_{j+1};\, \tilde{q}^{[\tau_j]}, p\Big)$. The corresponding optimization problem now takes on the following form of a *Multiple Shooting Dynamical Parameter Identification Problem (MS-DPIP)*:

Problem 3.18 (MS-DPIP)

Find variables $\tilde{q}^{[\tau_1]}, \ldots, \tilde{q}^{[\tau_{N_\mathrm{s}}]} \in \mathbb{R}^{n_q}$ and $p \in \mathbb{R}^{n_p}$ that

$$\begin{aligned} \text{minimize} \quad & \frac{1}{N_\mathrm{d}} \sum_{i=1}^{N_\mathrm{d}} \Big\| \tilde{q}|_{\mathcal{J}}\Big(t_i;\, \tilde{q}^{[\tau_{l(i)}]}, p\Big) - \bar{q}^{[t_i]} \Big\|_2^2 \\ \text{subject to} \quad & \tilde{q}\Big(\tau_{j+1};\, \tilde{q}^{[\tau_j]}, p\Big) - \tilde{q}^{[\tau_{j+1}]} = 0_{n_q} \quad \text{for } j = 1, \ldots, N_\mathrm{s} - 1 \end{aligned}$$

with $l(i) \coloneqq \max\limits_{1 \le j \le N_\mathrm{s}} \{j \colon \tau_j \le t_i\}$ representing a shooting node index.

Remark 3.19 It is not necessary that the shooting nodes are distributed on the time interval in an equidistant manner. In [50], for instance, it is shown how shooting nodes can be selected adaptively. It is also common that the ODE is solved on each sub-interval accurately using either an adaptive integration method or a small integration step size. This is, in fact, one of the many advantages of multiple shooting methods: a more accurate approximation of the solution of the ODE does not necessarily imply higher problem dimensions. However, in this work, we focus less on an accurate ODE solution and more on the effects of higher problem dimensions.

The multiple shooting approach can be seen as both a sequential and simultaneous transcription method and is often characterized as being in between both ideas. On the one hand, integration still takes place that is invisible to the solver. However, this is only done on a smaller time interval and is therefore less likely to cause problems. On the other hand, the search space on NLP level is increased and finding better solutions may be easier due to the enhanced flexibility.

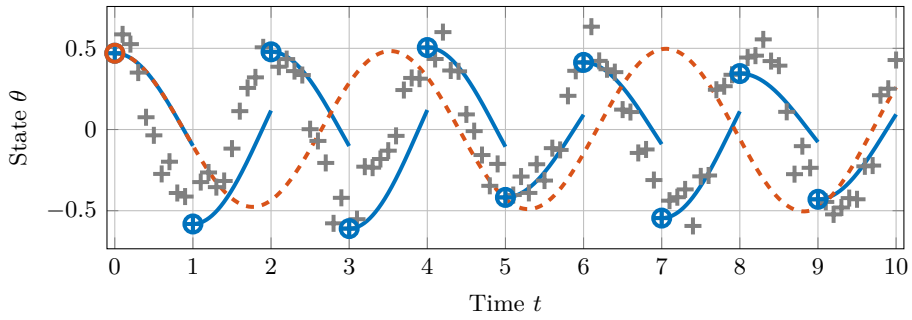


Figure 3.4: Illustration of direct multiple shooting and comparison against single shooting for the pendulum example. Gray crosses: measurement points; red circle: initial value for single shooting; red dotted line: single shooting trajectory; blue circles: measurement values at shooting nodes; blue lines: multiple shooting trajectories.

Example 3.20

For illustration, we consider the pendulum example with all ingredients from Example 3.10 and Example 3.15. We introduce 10 equidistantly distributed shooting nodes on the time interval. Thus, the problem dimensions increase from $1 + 2 = 3$ to $1 + 2 \cdot 10 = 21$ variables and $2 \cdot 9 = 18$ constraints are added. Still, the problem has a low size, but one can already have a premonition of potential advantages, as illustrated in Figure 3.4. Here, the model is evaluated with a poor initial parameter guess for the single shooting method. The corresponding trajectory is far off the measurements. If the integration is restarted at the shooting nodes, initialized by the measurements, the model cannot deviate too much due to the shortened time horizon. In this example, the number of shooting nodes is selected manually. With fewer nodes, the effects would not be visually obvious.

Initialization of (MS-DPIP)

In addition to the primary variables as described in the respective paragraph in Subsection 3.2.3, the shooting node values $\tilde{q}^{[\tau_i]}$ $(i = 1, \ldots, N_\mathrm{s})$ as secondary variables need to be initialized. A natural way is to choose the corresponding measurement values (as far as they exist). This leads to an excellent initial guess, which can help a solver overcoming undesired local minima[13].

[13]This aspect is investigated in Section 4.1.

Computation of Derivatives

The introduction of shooting nodes increases the problem dimensions: the number of optimization variables rises from $n_p + n_q$ to $n_p + N_\mathrm{s} n_q$, the number of constraints amounts to $n_q(N_\mathrm{s} - 1)$. The resulting derivatives exhibit sparse structures that originate from the specific problem formulation. Starting with the objective function, we need to distinguish different cases:

(i) If each measurement point corresponds to the location of a shooting node ($\bar{t}_i \in \{\tau_1, \dots, \tau_{N_\mathrm{s}}\}$, $i = 1, \dots, N_\mathrm{m}$), the objective function is independent of p.
(ii) If there are less shooting nodes than measurement points ($N_\mathrm{s} < N_\mathrm{m}$), the objective function's gradient is dense. This is the typical case.
(iii) In case there are at least two shooting nodes between two measurement points, the objective function would only depend on the last one that is located directly in front of the next measurement point (see the definition of l in (MS-DPIP)). Thus, the gradient can be sparse — in practice, however, this case can be neglected.

In the following, we assume to be in case (ii).

The constraints, consisting of continuity conditions, are more important for sparsity considerations. Since a shooting node only depends on the previous one (as initial value) and the parameters, a block structure arises. In the following, we collect the optimization variables in $z \coloneqq \left(\tilde{q}^{[\cdot]}, p\right)$ and the constraints in

$$H_\mathrm{MS}(z) \coloneqq \begin{pmatrix} \tilde{q}\left(\tau_2;\, \tilde{q}^{[\tau_1]}, p\right) - \tilde{q}^{[\tau_2]} \\ \vdots \\ \tilde{q}\left(\tau_{N_\mathrm{s}-1};\, \tilde{q}^{[\tau_{N_\mathrm{s}-2}]}, p\right) - \tilde{q}^{[\tau_{N_\mathrm{s}-1}]} \\ \tilde{q}\left(\tau_{N_\mathrm{s}};\, \tilde{q}^{[\tau_{N_\mathrm{s}-1}]}, p\right) - \tilde{q}^{[\tau_{N_\mathrm{s}}]} \end{pmatrix} \in \mathbb{R}^{(N_\mathrm{s}-1)n_q}.$$

Its derivative with respect to $\tilde{q}^{[\cdot]}$ has a sparse block structure:

$$\nabla_{\tilde{q}^{[\cdot]}} H_\mathrm{MS}(z) = \begin{pmatrix} r_1 & -I & & & & \\ & r_2 & -I & \ddots & & \\ & & \ddots & \ddots & & \\ & & & r_{N_\mathrm{s}-2} & -I & \\ & & & & r_{N_\mathrm{s}-1} & -I \end{pmatrix} \in \mathbb{R}^{(N_\mathrm{s}-1)n_q \times N_\mathrm{s} n_q}$$

with partial derivatives[14]

$$r_i \coloneqq \frac{\partial \tilde{q}}{\partial \tilde{q}^{[\tau_i]}}\Big(\tau_{i+1}; \tilde{q}^{[\tau_i]}, p\Big) \in \mathbb{R}^{n_q \times n_q}$$

and $I \in \mathbb{R}^{n_q \times n_q}$ being the identity matrix. The partial derivative with respect to the model parameters is generally dense:

$$\nabla_p H_{\mathrm{MS}}(z) = \begin{pmatrix} \frac{\partial \tilde{q}}{\partial p}\Big(\tau_2; \tilde{q}^{[\tau_1]}, p\Big) \\ \vdots \\ \frac{\partial \tilde{q}}{\partial p}\Big(\tau_{N_\mathrm{s}}; \tilde{q}^{[\tau_{N_\mathrm{s}-1}]}, p\Big) \end{pmatrix} \in \mathbb{R}^{(N_\mathrm{s}-1)n_q \times n_p}.$$

Next, we denote the objective function in (MS-DPIP) as $F_{\mathrm{MS}}(z)$. Each component of its gradient only depends on the respective shooting node in combination with the model parameters. For the Lagrange function

$$\mathcal{L}_{\mathrm{MS}}(z, \mu) \coloneqq F_{\mathrm{MS}}(z) + \sum_{i=1}^{N_\mathrm{s}-1} \mu_i^\mathsf{T} H_{\mathrm{MS}i}(z),$$

we therefore obtain the following partial derivatives:

$$\nabla_{\tilde{q}^{[\cdot]}} \mathcal{L}_{\mathrm{MS}}(z, \mu) = \begin{pmatrix} \nabla_{\tilde{q}^{[\tau_1]}} F_{\mathrm{MS}}(z) \\ \nabla_{\tilde{q}^{[\tau_2]}} F_{\mathrm{MS}}(z) \\ \vdots \\ \nabla_{\tilde{q}^{[\tau_{N_\mathrm{s}-1}]}} F_{\mathrm{MS}}(z) \\ \nabla_{\tilde{q}^{[\tau_{N_\mathrm{s}}]}} F_{\mathrm{MS}}(z) \end{pmatrix} + \begin{pmatrix} s_1 \mu_1 \\ -\mu_1 + s_2 \mu_2 \\ \vdots \\ -\mu_{N_\mathrm{s}-2} + s_{N_\mathrm{s}-1} \mu_{N_\mathrm{s}-1} \\ -\mu_{N_\mathrm{s}-1} \end{pmatrix} \in \mathbb{R}^{N_\mathrm{s} n_q},$$

$$\nabla_p \mathcal{L}_{\mathrm{MS}}(z) = \nabla_p F_{\mathrm{MS}}(z) + \sum_{i=1}^{N_\mathrm{s}-1} \nabla_p H_{\mathrm{MS}i}(z)^\mathsf{T} \mu_i \in \mathbb{R}^{n_p},$$

which are dense. The Lagrange multipliers are denoted as $\mu \in \mathbb{R}^{(N_\mathrm{s}-1)n_q}$. Following, the mixed second-order derivatives

$$\nabla^2_{\tilde{q}^{[\cdot]}p} \mathcal{L}_{\mathrm{MS}} \in \mathbb{R}^{N_\mathrm{s} n_q \times n_p} \text{ and } \nabla^2_{p\tilde{q}^{[\cdot]}} \mathcal{L}_{\mathrm{MS}} \in \mathbb{R}^{n_p \times N_\mathrm{s} n_q}$$

describe dense matrices, as well. Obviously, this is also the case for the deriva-

[14] These derivatives can be computed via sensitivity equations, for example.

tive $\nabla^2_{pp}\mathcal{L}_{\mathrm{MS}} \in \mathbb{R}^{n_p \times n_p}$. The second-order derivative with respect to the shooting nodes $\nabla^2_{\tilde{q}^{[\cdot]}\tilde{q}^{[\cdot]}}\mathcal{L}_{\mathrm{MS}}$ is a block-diagonal matrix, and hence, the Hessian matrix of the Lagrange function has the following form:

$$\nabla^2\mathcal{L}_{\mathrm{MS}} = \begin{bmatrix} \times & & & \times \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ \times & \cdots & \cdots & \times \end{bmatrix} \in \mathbb{R}^{n_z \times n_z}.$$

Herein, the symbol $\times$ denotes a non-zero element of appropriate size. Empty spaces correspond to structural zeros. It can be seen that each additional shooting node increases the dimensions, adds another diagonal block, and increases the absolute number of non-zero elements.

For optimal control problems[15], which are similar to parameter identification problems, Büskens [18] shows that there exists an optimal number of shooting nodes with respect to an efficient derivative computation, in which the number of non-zero elements is the basis for his calculations. Due to the absence of control terms (as optimizable quantities) in parameter identification problems, this statement is not valid in our specific setting. Here, the computational demand rises. Roughly speaking, each additional shooting node leads to another block in the respective matrices.

The question, how many shooting nodes one should introduce, is still not answered. On the one hand, less shooting nodes lead to a smaller problem with more complex "hidden" numerical simulations of the ODE. On the other hand, more shooting nodes increase the problem dimensions, but reduce the amount of inner iterations due to the shorter time intervals. In total, however, this amount stays the same. Additionally, it depends on the tolerance for the constraint violation up to which accuracy the ODE is solved.

The required number of shooting nodes to successfully solve a problem depends strongly on the dynamics and the given measurements, as it can be seen in Figure 3.4. Too few shooting nodes allow longer time integrations, which may lead to large deviations from the measurements using poor initial parameter guesses. Conversely, even with poor initial parameters, many shooting nodes reduce the possibility of deviations from the measurements due to the shorter time horizons. This phenomenon is investigated on an example in Section 4.1, where the number of shooting nodes is in the focus.

[15] Instead of model parameters, a time-dependent control term is considered.

Literature Review

The multiple shooting method has its origins in the solution of boundary value problems and found early mention in [65] and [107]. The resulting system of equations is usually solved numerically using Newton's method. This technique can also be used in optimization problems in which the system of equations is treated as artificial constraints, as explained above. In the case of dynamical parameter identification problems, Bock is one of the main contributors, for early works we refer to [11, 12]. Since then, multiple shooting has become the subject of countless research papers. Although many contributions exist in which it is applied to solve optimal control problems, parameter identification has also gained interest in the past. Bock et al. [14] investigate multiple shooting for differential-algebraic equations, while Müller and Timmer [81] apply this approach to PDEs. Zimmer et al. [131] focus on the landscape of the objective function within stochastic systems. They show that multiple shooting reduces the number of local minima on the one hand and has a smoothing effect on the objective function, on the other hand.

To demonstrate its advantages, multiple shooting is often compared to single shooting, see for example the references in Subsection 3.2.3. In this context, several characteristics are also pointed out by means of an academic example given in Section 4.1.

3.2.5 Decomposed Formulation

A disadvantage of the reduced formulation is that the solver cannot directly optimize for the discretized trajectories, although explicitly used in the objective function. This can be partly overcome by direct multiple shooting methods, in which the problem dimensions are increased to attain more flexibility in the solution process. Still, numerical integration takes place that is hidden to a solver.

The idea of decomposition is to optimize for both the trajectories and the parameters *simultaneously* by including each step of the integration procedure in the problem formulation. Consequently, the ODE does not have to be solved in each iteration of an NLP solver — the solution trajectories are allowed to be infeasible during the course of optimization. Thus, the trajectories can be optimized directly to minimize the objective function while, at the same time, being optimized to fulfill the equations representing the integration scheme. This results in a so-called *Decomposed Dynamical Parameter Identification Problem (D-DPIP)*.

Problem 3.21 (D-DPIP)

Find variables $\tilde{q}^{[t_1]}, \tilde{q}^{[t_2]}, \ldots, \tilde{q}^{\left[t_{N_\mathrm{d}}\right]} \in \mathbb{R}^{n_q}$ and $p \in \mathbb{R}^{n_p}$ that

$$\begin{aligned} &\text{minimize} && \frac{1}{N_\mathrm{d}} \sum_{i=1}^{N_\mathrm{d}} \left\| \tilde{q}|_{\mathcal{J}}^{[t_i]} - \bar{q}^{[t_i]} \right\|_2^2 \\ &\text{subject to} && H_{\mathrm{ODE}}(\tilde{q}, p) = 0_{(N_\mathrm{d}-1)n_q}. \end{aligned}$$

In this problem, it is no longer necessary to simulate the system in each iteration of the used solution algorithm, since this is implicitly achieved once a solution of the optimization problem is found. In fact, this is already achieved once a feasible point is found — a feasible point of (D-DPIP) corresponds to a trajectory generated by a numerical integration scheme applied to the dynamics. All dependencies are now explicit.

In contrast to (R-DPIP), it does no longer matter whether the incremental function is explicit or implicit. To also address a drawback, adaptive integration methods cannot be used in this formulation, since a fixed time grid needs to be given to formulate the optimization problem.

Initialization of (D-DPIP)

In analogy to the description for multiple shooting (see Subsection 3.2.4), the discretized trajectory can be initialized using the measurements. If measurement and discretization grid do not coincide, intermediate values can be interpolated. This initialization can have a strong effect on the algorithmic behavior. For demonstration, we illustrate the advantages of added dimensions in combination with excellent initial guesses in the following example:

Example 3.22

Again, we return to Example 3.10 with the specifications made in Example 3.15. For optimization variables

$$z = \left(\tilde{q}_1^{[t_1]}, \tilde{q}_2^{[t_1]}, \ldots, \tilde{q}_1^{\left[t_{N_\mathrm{d}}\right]}, \tilde{q}_2^{\left[t_{N_\mathrm{d}}\right]}, L \right)^\mathsf{T},$$

the decomposed problem attains the following form:

$$
\begin{aligned}
\underset{z}{\text{minimize}} \quad & \frac{1}{N_\mathrm{m}} \sum_{i=1}^{N_\mathrm{m}} \left(\tilde{q}_1^{[\bar{t}_i]} - \bar{q}_1^{[\bar{t}_i]} \right)^2 \\
\text{subject to} \quad & \tilde{q}_1^{[t_{k+1}]} - \tilde{q}_1^{[t_k]} - h\tilde{q}_2^{[t_k]} = 0, \quad k = 1, \ldots, N_\mathrm{d} - 1 \\
& \tilde{q}_2^{[t_{k+1}]} - \tilde{q}_2^{[t_k]} + h\frac{g}{L} \sin\left(\tilde{q}_1^{[t_k]} \right) = 0, \quad k = 1, \ldots, N_\mathrm{d} - 1 \\
& 0 \leq L \\
& L \leq 2.
\end{aligned}
$$

Note that the scaling factor $\frac{1}{2}$ is replaced by $\frac{1}{N_\mathrm{m}}$ to eliminate the dependency on the number of measurement points. With $N_\mathrm{d} = 2001$ discretization points, the problem exhibits 4003 variables with 4002 constraints, including two box constraints.
In this example, we want to compare the behavior of `WORHP`'s SQP algorithm for this and the corresponding reduced problem formulation. We restrict the initial values to be close to their nominal ones (this is necessary for visualization aspects — for differing initial values, the shape in Figure 3.3 would no longer be valid). Next, we initialize the parameter as $L_\mathrm{ini} = 1.4$ and the state variable $\tilde{q}_{1\mathrm{ini}}$ by the measurements, $\tilde{q}_{2\mathrm{ini}}$ is initialized constantly by zero (which corresponds to the nominal initial value). For discretization points without corresponding measurement points, a linear interpolation is applied for initialization.
The results are presented in Figure 3.5, which shows the objective function's shape (of the reduced problem formulation, see Figure 3.3). It is not surprising that the algorithm applied to the reduced formulation does not converge to the better minimum: the implicit constraints require that the iterates remain on the blue line. When the algorithm is applied to the decomposed problem formulation, several observations can be made. The initialization allows starting at an excellent objective function value near zero and the incorporated integration allows moving through the parameter space in a more flexible way: the better minimum can be attained.

Computation of Derivatives

In this approach, the number of optimization variables and equality constraints is usually large. This can have a severe impact on computation times, especially for second-order methods as they are used in this work. Due to the specific form of the constraints, the derivatives possess a specific structure, which can be exploited.

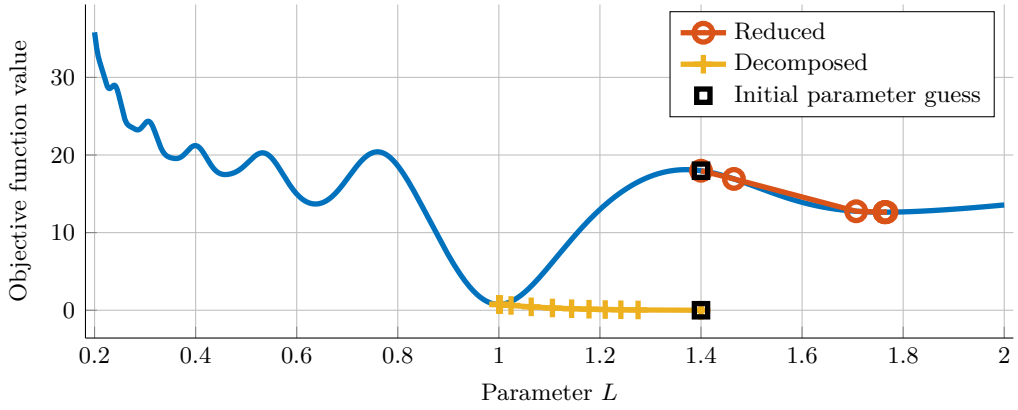


Figure 3.5: Optimization paths projected onto the objective function of the reduced formulation for the pendulum example.

In the following, we denote the objective function in Problem 3.21 as F_{D}, while the equality constraints consisting of the integration scheme will be called H_{D}. In this approach, the optimization variables amount to

$$z \coloneqq \begin{pmatrix} \tilde{q}^{[t_1]\mathsf{T}} & \tilde{q}^{[t_2]\mathsf{T}} & \dots & \tilde{q}^{[t_{N_\mathrm{d}}]\mathsf{T}} & p^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}} \in \mathbb{R}^{N_\mathrm{d} n_q + n_p}.$$

We assume that there exist measurements for all states (that is $\mathcal{J} = \{1, \dots, n_q\}$) and hence, the objective function depends on all discrete state variables. We obtain

$$\nabla_{\tilde{q}} F_{\mathrm{D}}(z) = \frac{2}{N_\mathrm{d}} \begin{pmatrix} \tilde{q}^{[t_1]} - \bar{q}^{[t_1]} \\ \tilde{q}^{[t_2]} - \bar{q}^{[t_2]} \\ \vdots \\ \tilde{q}^{[t_{N_\mathrm{d}}]} - \bar{q}^{[t_{N_\mathrm{d}}]} \end{pmatrix}$$

and since it does not explicitly depend on p, we have $\nabla_p F_{\mathrm{D}}(z) = 0_{n_p}$. If only $\mathcal{J} \subsetneq \{1, \dots, n_q\}$, there would be additional structural zero elements.

To investigate the function of the constraints

$$H_{\mathrm{D}}(z) \coloneqq \begin{pmatrix} \tilde{q}^{[t_2]} - \tilde{q}^{[t_1]} - h\phi\left(\tilde{q}^{[t_1]}, \tilde{q}^{[t_2]}, p\right) \\ \vdots \\ \tilde{q}^{[t_{N_\mathrm{d}}]} - \tilde{q}^{[t_{N_\mathrm{d}-1}]} - h\phi\left(\tilde{q}^{[t_{N_\mathrm{d}-1}]}, \tilde{q}^{[t_{N_\mathrm{d}}]}, p\right) \end{pmatrix} \in \mathbb{R}^{(N_\mathrm{d}-1)n_q}$$

we make the following abbreviations for the purpose of a better readability:

$$k_{i,j} \coloneqq -I_{n_q \times n_q} - h \nabla_{\tilde{q}^{[t_i]}} \phi\Big(\tilde{q}^{[t_j]}, \tilde{q}^{[t_{j+1}]}, p\Big) \in \mathbb{R}^{n_q \times n_q},$$
$$\hat{k}_{i,j} \coloneqq I_{n_q \times n_q} - h \nabla_{\tilde{q}^{[t_i]}} \phi\Big(\tilde{q}^{[t_j]}, \tilde{q}^{[t_{j+1}]}, p\Big) \in \mathbb{R}^{n_q \times n_q},$$

and

$$r_i \coloneqq \nabla_p \phi\Big(\tilde{q}^{[t_i]}, \tilde{q}^{[t_{i+1}]}, p\Big) \in \mathbb{R}^{n_q \times n_p}$$

with I being the identity matrix of respective dimension. Due to the integration scheme given as a one-step method, the partial derivative of the constraints with respect to the discretized states has a sparse structure, as shown in the following pattern:

$$\nabla_{\tilde{q}} H_{\mathrm{D}}(z) = \begin{pmatrix} k_{1,1} & \hat{k}_{2,1} & & & & \\ & k_{2,2} & \hat{k}_{3,2} & \cdots & & \\ & & \ddots & \ddots & \ddots & \\ & & & & k_{N_{\mathrm{d}}-1,N_{\mathrm{d}}-1} & \hat{k}_{N_{\mathrm{d}},N_{\mathrm{d}}-1} \end{pmatrix} \in \mathbb{R}^{(N_{\mathrm{d}}-1)n_q \times N_{\mathrm{d}} n_q}.$$

Here we assume that the integration scheme is implicit; for explicit ones the structure would differ slightly. Since the ODE-defining function s depends on the model parameters p, the partial derivative of H_{D} with respect to p is generally dense:

$$\nabla_p H_{\mathrm{D}}(z) = -h \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N_{\mathrm{d}}-2} \\ r_{N_{\mathrm{d}}-1} \end{pmatrix} \in \mathbb{R}^{(N_{\mathrm{d}}-1)n_q \times n_p}.$$

In the following, the structure of the Hessian (with respect to z) of the Lagrange function

$$\mathcal{L}_{\mathrm{D}}(z, \mu) \coloneqq F_{\mathrm{D}}(z) + \sum_{i=1}^{N_{\mathrm{d}}-1} \mu_i^{\mathsf{T}} H_{\mathrm{D}i}(z)$$

with Lagrange multipliers $\mu \in \mathbb{R}^{(N_{\mathrm{d}}-1)n_q}$ will be derived. It can be easily seen that

the gradients

$$\nabla_{\tilde{q}}\mathcal{L}_{\mathrm{D}}(z,\mu) = \frac{2}{N_{\mathrm{d}}} \begin{pmatrix} \tilde{q}^{[t_1]} - \bar{q}^{[t_1]} \\ \tilde{q}^{[t_2]} - \bar{q}^{[t_2]} \\ \vdots \\ \tilde{q}^{[t_{N_{\mathrm{d}}-1}]} - \bar{q}^{[t_{N_{\mathrm{d}}-1}]} \\ \tilde{q}^{[t_{N_{\mathrm{d}}}]} - \bar{q}^{[t_{N_{\mathrm{d}}}]} \end{pmatrix} + \begin{pmatrix} s_{1,1}\mu_1 \\ \hat{s}_{2,1}\mu_1 + s_{2,2}\mu_2 \\ \vdots \\ \hat{s}_{N_{\mathrm{d}}-1,N_{\mathrm{d}}-2}\mu_{N_{\mathrm{d}}-2} + s_{N_{\mathrm{d}}-1,N_{\mathrm{d}}-1}\mu_{N_{\mathrm{d}}-1} \\ \hat{s}_{N_{\mathrm{d}},N_{\mathrm{d}}-1}\mu_{N_{\mathrm{d}}-1} \end{pmatrix} \in \mathbb{R}^{N_{\mathrm{d}}n_q}$$

and

$$\nabla_p \mathcal{L}_{\mathrm{D}}(z,\mu) = -h \sum_{i=1}^{N_{\mathrm{d}}-1} \left(\mu_i^{\mathsf{T}} r_i\right)^{\mathsf{T}} \in \mathbb{R}^{n_p}$$

are generally dense, but the Hessian matrix has a sparse structure. Although the partial second derivatives $\nabla^2_{\tilde{q}p}\mathcal{L}_{\mathrm{D}}$ and $\nabla^2_{pp}\mathcal{L}_{\mathrm{D}}$ are usually dense, the second derivative with respect to the discretized states is sparse with the following pattern:

$$\nabla^2_{\tilde{q}\tilde{q}}\mathcal{L}_{\mathrm{D}}(z,\mu) = \begin{bmatrix} \times & \times & & & & \\ \times & \times & \times & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & \times & \times & \times & \\ & & & \times & \times & \end{bmatrix} \in \mathbb{R}^{N_{\mathrm{d}}n_q \times N_{\mathrm{d}}n_q}.$$

Hence, the Hessian matrix

$$\nabla^2_{zz}\mathcal{L}_{\mathrm{D}} = \begin{pmatrix} \nabla^2_{\tilde{q}\tilde{q}}\mathcal{L}_{\mathrm{D}} & \nabla^2_{\tilde{q}p}\mathcal{L}_{\mathrm{D}} \\ \nabla^2_{p\tilde{q}}\mathcal{L}_{\mathrm{D}} & \nabla^2_{pp}\mathcal{L}_{\mathrm{D}} \end{pmatrix} \in \mathbb{R}^{n_z \times n_z}$$

is also sparse, since the number of discretization points is usually significantly larger than the number of model parameters.[16]

[16] For an example of sparsity properties, we refer to Subsection 4.2.1 within this work.

Literature Review

The decomposed approach is often referred to as full discretization, since each discrete point in the integration scheme is treated as an optimization variable. This transcription strategy can often be found in the context of optimal control problems, for example in [9] or in [8], where also a section on parameter estimation can be found. Fliege et al. [25] apply the full discretization approach to trajectory optimization in the application of disaster assessment. For parameter identification, the decomposed approach has found less attention than the multiple shooting approach. Still, it is used in [52] for aircraft applications, in [124] for dry machining in industrial manufacturing, or in [31] for estimating parameters within a rover model.

3.3 Decomposition in Bilevel Programming

In this section, we consider finite-dimensional optimization problems that require the solution of a nested, parameterized problem. More precisely, the objective function and the constraints depend on *upper-level variables* and *lower-level variables*. The latter minimize another objective function with respect to constraints, both of them can be parameterized in the upper-level variables. These types of problems are called *bilevel programs* and are typically difficult to solve due to their intricate structure.

Bilevel optimization problems appear in a rising number of areas, such as transportation [113], economics [5], chemical industry [23], or inverse optimal control [2]. The corresponding problems usually differ in their structure, which requires a careful choice of solution methods.

The development of algorithms for the global solution of general nonlinear bilevel programs is often a challenging task. Guaranteed convergence is usually given under strict conditions on a specific problem. Therefore, it is desirable to find general reformulations that interact well with existing nonlinear programming methods. Since this involves the efficient computation of local solutions, one needs to find problem formulations that are *more likely* to converge to global solutions. Equally important for a newly designed algorithm is its general applicability: it should be able to solve a wide variety of problems, not just perform well on a few illustrative examples.

Bilevel programming fits into the framework presented in Subsection 3.1.1, since an evaluation of the problem-defining functions requires obtaining the solution of

another optimization problem. From the numerical point of view, this solution is obtained by applying an iterative algorithm.

In this section, it is demonstrated how decomposition can be applied to this class of optimization problems. In particular, a novel reformulation method to transform the given problem into a single-level one is presented. We present a reduction technique based on the idea that the numerical solution process of the lower-level problem is incorporated into the upper level by artificial constraints and variables. More precisely, we choose an SQP method. In contrast to solving the lower-level problem in each iteration, it is implicitly solved once the whole problem is solved, and we therefore allow iterates that are infeasible for the lower-level problem during the course of the outer iteration. The usage of local methods like SQP generally comes at the price of converging to local solutions. With the presented reformulation, however, a lower-level response is no longer dictated by upper-level variables, which increases the possible search space for a chosen solver. With the help of sophisticated initializations, the chance of converging to better or even global solutions can be influenced.

The approach presented in the following is inspired by direct methods for dynamical parameter identification problems as in the previous section. We adapt the idea of full discretization by adding SQP iteration steps for solving the lower-level problem as artificial constraints to the upper level of the original bilevel problem considered here. The aim is to achieve similar effects as in ODE-constrained optimization.

3.3.1 Original Problem Formulation

We consider nonlinear optimization problems that involve a subordinate nonlinear problem, which we call lower-level problem (LL). This problem depends on a parameter $x \in \mathbb{R}^{n_x}$, while it is solved for $y \in \mathbb{R}^{n_y}$. Its objective function is given by $f\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$, constraints are called $g\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$:

Problem 3.23 (LL)

Find variables $y \in \mathbb{R}^{n_y}$ that

$$\begin{aligned} &\text{minimize} && f(x, y) \\ &\text{subject to} && g(x, y) \leq 0_{n_g}. \end{aligned}$$

As we do not make any assumptions on the convexity of (LL), there may exist a set of solutions instead of a single one. Our bilevel problems of interest depend on the variables x and y, while it is a constraint that the latter is a solution of (LL). The upper-level objective function is $F\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$, the upper-level constraints function is called $G\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_G}$. With these ingredients, we can formulate the *Original Bilevel Problem (O-BP)*:

Problem 3.24 (O-BP)

Find upper-level variables $x \in \mathbb{R}^{n_x}$ and lower-level variables $y \in \mathbb{R}^{n_y}$ that

$$\begin{aligned} \text{minimize} \quad & F(x,y) \\ \text{subject to} \quad & G(x,y) \leq 0_{n_G} \\ & y \in \arg\min_{v} \quad f(x,v) \\ & \qquad \text{s.t.} \quad g(x,v) \leq 0_{n_g}. \end{aligned}$$

This problem class is topic of many books (for example [30] or [104]) and the following descriptions partly stem from them. Whenever necessary, more precise references are given.

Remark 3.25 We formulate the problems using inequality constraints only. This is justified by the fact that, on the one hand, the following considerations also apply to equality constraints, and, on the other hand, all the numerical examples that follow only involve inequality constraints.

Due to the constraint

$$\begin{aligned} y \in \arg\min_{v} \quad & f(x,v) \\ \text{s.t.} \quad & g(x,v) \leq 0_{n_g}, \end{aligned} \tag{3.5}$$

bilevel problems represent an intricate class of optimization problems and finding a solution is typically a challenging task. Since there may not exist a unique solution of (LL), but several ones, different strategies for choosing one of them can be applied. In the so-called *optimistic* approach, that solution is selected which is the best suitable for the upper-level problem. Here, both levels cooperate. Contrarily, in the *pessimistic* approach, all lower-level solutions are taken into account and hence, the worst one needs to be selected. From now on, we follow the optimistic perspective.

There are different solution strategies to solve Problem 3.24. To directly address (O-BP), there exist techniques such as a bounding algorithm in [79] or a tabu search in [61]. Most strategies, however, rely on reformulating the problem as (or reducing it to) a *single-level* problem. Once such a single-level reformulation is found, one can use the theory, methods, and software mentioned in Chapter 2 to solve it.

3.3.2 Single-Level Reformulation

To make use of the theory and algorithms for standard nonlinear programming, one often reformulates (O-BP) as a problem of type (NLP). It is desirable to construct problems that share the same solutions and are thus equivalent. However, this is only possible in rare cases, for example when the lower-level problem is convex. For general nonlinear and nonconvex problems, one can usually not assume to find an equivalent problem formulation.

Similar to the description in Subsection 3.2.2, we present different ways of reformulation and outline some of their characteristics: a *reduced formulation* (Subsection 3.3.3), the well-known *KKT approach* (Subsection 3.3.4), and an alternative method, which will be called *decomposed formulation* (Subsection 3.3.5) in the following. They all have in common that they can be represented in terms of a standard NLP and thus, numerical routines such as SQP algorithms can be applied to find a solution candidate. However, all of them have their individual advantages and drawbacks.

Often, the requirement to find global solutions is reduced to finding local solutions by a problem transformation that allows the use of efficient local methods. Nevertheless, finding global solutions is the subject of many research articles. In [116], the KKT approach is used to compute global solutions of quadratic and linear problems. A combination with global search algorithms is given in [108]. Kleniati and Adjiman developed a special branch-and-bound technique using a value function reformulation [67, 68]. In [79], an algorithm is developed that computes the global minimum of bilevel programs with nonconvex lower-level problems. In addition to the mentioned references, a comprehensive overview of applications, solution methods, and theoretical foundations is given in [104].

Since we aim to apply second-order optimization methods to solve problems in the presented formulations, we assume that all functions are sufficiently smooth. In particular, the decomposed approaches in Subsection 3.3.5 require the functions to be four times continuously differentiable.

3.3.3 Reduced Formulation

In the general case, a solution of the lower-level problem depends on its parameterization $x \in \mathbb{R}^{n_x}$. Therefore, it is obvious to formulate a reduced version of (O-BP), which corresponds to the reduced problem (R-NLP). We consider the upper-level variable x as primary variable, while a solution $y^\star$ of (LL) represents the secondary variable. As introduced in Subsection 3.1.1, we optimize only for the primary variables. Since the lower-level solution $y^\star$ must be accessed within the objective function, the main idea, roughly speaking, is to solve (LL) whenever needed. This is where the expression $\psi[x]$ comes in again.

Remark 3.26 As already mentioned, there may exist more than one solution of the lower-level problem. In this case, one has to be careful which solution to choose. From a theoretical point of view, one would have to compute all existing solutions to select the best or worst one, dependent on whether an optimistic or pessimistic approach is used.

To avoid computing all existing solutions, we make the following assumption:

Assumption 3.27 We assume that there exists a unique mapping from x to a lower-level solution $y^\star$. This is also called, with additional specifications, *strong stability* [30].

In simple examples, it may be possible to formulate ψ explicitly. In practice, however, this mapping results in the numerical solution of (LL) parameterized in x, a possibly highly nonlinear operation. A finite sequence of algorithm steps is performed until an approximate solution is found, depending only on the initialization, the parameterization, and possibly algorithm configurations. All in all, this is a complex and intricate behavior that depends on the problem at hand, with the advantage that a solution is approximated if the convergence requirements of the algorithm are met. The expression ψ connects the upper- with the lower-level problem, we call the resulting problem a *Reduced Bilevel Problem (R-BP)*:

Problem 3.28 (R-BP)

Find variables $x \in \mathbb{R}^{n_x}$ that

$$\begin{aligned} &\text{minimize} && F(x, \psi(x)) \\ &\text{subject to} && G(x, \psi(x)) \leq 0_{n_G}. \end{aligned}$$

To solve this problem, there are decisions that have to be made:

- Which algorithm to use for the lower-level problem?
- How to initialize the lower-level problem?
- How to proceed in case the algorithm fails to compute a solution of the lower-level problem?

Problem 3.28 also has some advantages, such as reduced problem dimensions (n_x optimization variables, n_G constraints) or the reduced implementation effort.

Initialization of (R-BP)

Since we consider general bilevel problems, it is not easy to make an intelligent initial guess without further information. In the reduced problem formulation, one only needs to initialize the upper-level variables x. However, to evaluate F or G, (LL) has to be solved numerically, for which we use a local algorithm that depends on an initialization. There are several possible strategies. One could use the same initialization over and over again, or one could use the previously-found optimal solution as the starting point for the next run. If the lower-level solution is not too sensitive to its parameterization, the latter strategy can reduce the computation time or even help the solver to find a solution.

Computation of Derivatives

Computing derivatives of problem functions in (R-BP) can be complicated. If the mapping ψ is unique and continuously differentiable, the derivatives depend on $\nabla_x \psi$, which can be difficult to compute if given only implicitly. It requires computing directional derivatives, which in turn requires solving another quadratic program, as noted in [30]. Thus, using a gradient-based algorithm to solve a bilevel problem in the reduced formulation can quickly become difficult. Alternatively, $\nabla_x \psi$ can be seen as the sensitivity of the lower-level solution to the parameterization x. Since this sensitivity can be obtained efficiently (see for example [19]), it is an interesting option. From a structural point of view, the resulting matrices are usually dense.

Literature Review

The reduced approach, also known as *implicit function approach*, is often used in gradient-free respective evolutionary methods [45]. In this regard, a genetic algorithm is presented by Li and Wang [73] for nonlinear bilevel optimization problems. In [103], it is presented how to avoid repeated lower-level optimizations. If the lower-level problem is unique, y is implicitly given by x, and descent methods can be applied.

For example, this is presented by Vicente et al. [114] for quadratic programs. To avoid the uniqueness condition, Dempe [29] developed a bundle algorithm based on regularization of the lower-level problem. In [90], non-smooth lower-level problems are considered with a focus on adding an *abstract algorithm* to the upper-level problem, which is closely related to this reduced formulation.

3.3.4 KKT Formulation

A classical way of reformulating (O-BP) is to replace the condition of being a solution of the lower-level problem (condition (3.5)) by first-order necessary optimality conditions (KKT conditions) of the respective problem. Since these conditions involve lower-level Lagrange multipliers $\lambda \in \mathbb{R}^{n_g}$, they are additionally introduced as optimization variables. With the Lagrange function $\ell\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_g} \to \mathbb{R}$, given by

$$\ell(x, y, \lambda) \coloneqq f(x, y) + \lambda^\mathsf{T} g(x, y),$$

the resulting problem reads as follows (compare KKT conditions in Theorem 2.8), we call it a *KKT Bilevel Problem (KKT-BP)*:

Problem 3.29 (KKT-BP)

Find variables $x \in \mathbb{R}^{n_x}$, $y \in \mathbb{R}^{n_y}$, and $\lambda \in \mathbb{R}^{n_g}$ that

$$\begin{aligned} \text{minimize} \quad & F(x, y) \\ \text{subject to} \quad & G(x, y) \leq 0_{n_G} \\ & \nabla_y \ell(x, y, \lambda) = 0_{n_y} \\ & g(x, y) \leq 0_{n_g} \\ & \lambda^\mathsf{T} g(x, y) = 0 \\ & \lambda \geq 0_{n_g}. \end{aligned}$$

The problem dimensions have increased in comparison to the reduced approach: there are $n_z \coloneqq n_x + n_y + n_g$ optimization variables and $n_C \coloneqq n_G + n_y + 2n_g + 1$ constraints.

In case the lower-level problem is convex and suitable regularity assumptions are fulfilled, the solution sets of (O-BP) and (KKT-BP) coincide. However, for the practical computation of a solution, one can show that for every feasible point the

regularity assumptions for Theorem 2.8 are not fulfilled. This is due to the conditions

$$\begin{aligned} g(x,y) &\leq 0_{n_g}, \\ \lambda^\mathsf{T} g(x,y) &= 0, \\ \lambda &\geq 0_{n_g}, \end{aligned} \tag{3.6}$$

which transform (KKT-BP) into a so-called mathematical program with complementarity constraints (MPCC), which not only entails theoretical issues, but is also difficult to solve numerically. To circumvent these issues, one often replaces the system (3.6) by an equivalent reformulation. The Fischer-Burmeister (FB) function $\Phi\colon \mathbb{R}^2 \to \mathbb{R}$ (see [37]), given by

$$\Phi(a,b) = a + b - \sqrt{a^2 + b^2},$$

possesses the desired properties, namely

$$\Phi(a,b) = 0 \iff a \geq 0,\ b \geq 0,\ ab = 0.$$

The non-differentiability of Φ can be overcome by using *smoothing* techniques as discussed in [37] or [64]. Utilizing the FB function instead of system (3.6) leads to the following problem formulation:

Problem 3.30 (KKT-FB-BP)

Find variables $x \in \mathbb{R}^{n_x}$, $y \in \mathbb{R}^{n_y}$, and $\lambda \in \mathbb{R}^{n_g}$ that

$$\begin{aligned} \text{minimize} \quad & F(x,y) \\ \text{subject to} \quad & G(x,y) \leq 0_{n_G} \\ & \nabla_y \ell(x,y,\lambda) = 0_{n_y} \\ & \Phi(\lambda_i, -g_i(x,y)) = 0 \quad \text{for } i = 1,\dots,n_g. \end{aligned}$$

Initialization of (KKT-BP)

In addition to the upper-level variables x, the lower-level variables $y \in \mathbb{R}^{n_y}$ and multipliers $\lambda \in \mathbb{R}^{n_g}$ need to be initialized. There are several possibilities: For a given x_{ini}, the lower-level problem could be solved in advance to use the corresponding solution and Lagrange multipliers as an initial guess for (KKT-BP). Alternatively,

the Lagrange multipliers are often initialized as either 0_{n_g} or 1_{n_g}. In [43], they are initialized as $\max\{0.01, -g(x_{\text{ini}}, y_{\text{ini}})\}$.

Computation of Derivatives

For the following investigations, we use the formulation (KKT-BP), in which we denote the system of constraints as C. The Jacobian of C then exhibits the following sparsity pattern:

$$\nabla C = \begin{bmatrix} \times & \times & 0_{n_G \times n_g} \\ \times & \times & \times \\ \times & \times & 0_{n_g \times n_g} \\ \times & \times & \times \\ 0_{n_g \times n_x} & 0_{n_g \times n_y} & \times \end{bmatrix}.$$

Due to the introduction of the Lagrange multiplier λ as an optimization variable, some entries are structural zeros.

The Lagrange function of (KKT-BP) is $\mathcal{L}_{\text{K}}(x, y, \lambda, \mu) = F(x, y) + \mu^{\mathsf{T}} C(x, y, \lambda)$ with Lagrange multipliers $\mu \in \mathbb{R}^{n_C}$. It is easy to see that $\begin{pmatrix} \nabla_x \mathcal{L}_{\text{K}}^{\mathsf{T}} & \nabla_y \mathcal{L}_{\text{K}}^{\mathsf{T}} & \nabla_\lambda \mathcal{L}_{\text{K}}^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}$ does not contain structural zeros (except those of the given problem). The Hessian matrix of the Lagrange function only possesses one block of zeros, caused by λ again:

$$\nabla^2 \mathcal{L}_{\text{K}} = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & 0_{n_g \times n_g} \end{bmatrix}.$$

Hence, sparsity depends mainly on the given problem and less on the KKT formulation.

Literature Review

The KKT formulation has received much attention and is considered to be one of the standard approaches to bilevel programming. Since the problem formulation is an MPCC, research in this direction can be found in [27] or [128] (in which the corresponding constraints are called “equilibrium constraints”). Theoretical studies on optimality conditions and constraint qualifications can be found in [28]. Allende and Still [3] developed a smoothing algorithm for the complementarity conditions within the KKT approach and present numerical results. The KKT formulation can be solved in several ways, such as vertex enumeration or branch-and-bound

algorithms. Kim et al. [66] investigate the use of NLP methods to solve such problems.

3.3.5 Decomposed Formulation

The KKT approach can already be seen as being a decomposition of the original problem, since artificial variables and constraints are introduced. In this part, we build upon this by a further decomposition. The main idea is to incorporate the lower-level iterative process — assumed to be given by an SQP method — into the upper-level problem via artificial constraints and variables. For that purpose, condition (3.5) is replaced by a set of constraints which imitate the process of numerically solving the lower-level problem. Similar to the KKT reduction scheme, this approach can be seen as simultaneously optimizing the upper- and the lower-level problem. Since the lower-level variables are no longer dictated by the upper-level ones, they can be optimized independently. This procedure is presented in the following in more detail.[17]

Solving the Lower-Level Problem

In this part, we consider the lower-level problem (LL) for a fixed x. Thus, it is parameterized in x, while it is solved for y. The corresponding Lagrange function is given by

$$\ell(x, y, \lambda) \coloneqq f(x, y) + \lambda^\mathsf{T} g(x, y).$$

As presented in Subsection 2.2.1, the basic idea in SQP methods is to iteratively improve an initial guess by solving quadratic approximations of the original nonlinear program. With regard to the setting described here, the QP has the following form:

Problem 3.31 (LL-QP)

Find variables $d \in \mathbb{R}^{n_y}$ that

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} d^\mathsf{T} \nabla^2_{yy} \ell(x, y, \lambda) d + \nabla_y f(x, y)^\mathsf{T} d \\ \text{subject to} \quad & g(x, y) + \nabla_y g(x, y)^\mathsf{T} d \leq 0_{n_g}. \end{aligned}$$

[17] Propositions 3.35 and 3.39 originate from a private communication with Prof. Dr. Jörg Fliege in 2019.

Once a solution d with corresponding dual variable ν is found, the iterates are updated in the following way with α_k being some step size (compare Subsection 2.2.1):

$$y_{k+1} = y_k + \alpha_k d_k, \tag{3.7}$$
$$\lambda_{k+1} = \nu_k. \tag{3.8}$$

As a necessary requirement for the following investigations, we use a full step $\alpha_k = 1$ instead of performing a line search. The procedure of solving the lower-level QP and updating the variables is repeated until necessary optimality conditions are satisfied up to a given tolerance. The process of solving the lower-level problem for a given initial guess $(y_{\text{ini}}, \lambda_{\text{ini}})$ can be formulated as follows:

Step 1 solve (LL-QP) with x, y_{ini}, λ_{ini} and obtain (d_1, ν_1)
update $y_2 = y_{\text{ini}} + d_1$
update $\lambda_2 = \nu_1$

$\vdots$ $\vdots$

Step $N_{\text{it}} - 1$ solve (LL-QP) with x, $y_{N_{\text{it}}-1}$, $\lambda_{N_{\text{it}}-1}$ and obtain $(d_{N_{\text{it}}-1}, \nu_{N_{\text{it}}-1})$
update $y_{N_{\text{it}}} = y_{N_{\text{it}}-1} + d_{N_{\text{it}}-1}$
update $\lambda_{N_{\text{it}}} = \nu_{N_{\text{it}}-1}$

In practice, solving (LL-QP) comes down to finding a point $(d, \nu) \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_g}$ that fulfills the corresponding KKT conditions

$$g(x,y) + \nabla_y g(x,y) d \leq 0_{n_g}, \tag{3.9}$$
$$\nabla^2_{yy} \ell(x,y,\lambda) d + \nabla_y f(x,y) + \nabla_y g(x,y)^\mathsf{T} \nu = 0_{n_y}, \tag{3.10}$$
$$\nu^\mathsf{T} (g(x,y) + \nabla_y g(x,y) d) = 0, \tag{3.11}$$
$$\nu \geq 0_{n_g}. \tag{3.12}$$

Note that such a KKT point is a solution only if (LL-QP) is convex. All in all, the procedure of numerically solving (LL) can be formulated as blocks of equality and inequality constraints for which a feasible point has to be found. In analogy to the concept presented in Subsection 3.1.1, the main idea for finding a reformulation of (O-BP) is to replace solving the lower-level problem by new constraints that represent the iteration scheme described above using the KKT conditions of (LL-QP). Once these constraints are satisfied, $y_{N_{\text{it}}}$ hopefully represents an approximation to a solution of the lower-level problem, depending on N_{it}.

Embedding the Algorithm into the Upper-Level Problem

We introduce artificial variables

$$y_1, \dots, y_{N_{\text{it}}-1} \in \mathbb{R}^{n_y},$$

which represent the values of the SQP iterations. They refer to the update step

$$y_{k+1} = y_k + d_k.$$

Then, we replace the optimization variable y with $y_{N_{\text{it}}}$, which represents the variable at iteration step N_{it}. Under certain assumptions (those that the SQP method convergences, compare Theorem 2.16), and especially when N_{it} is chosen sufficiently large, $y_{N_{\text{it}}}$ represents an approximate lower-level KKT point found by the imitated SQP algorithm at the end of the (upper-level) optimization process.

Note that the iteration steps y_i are no longer implicitly dependent on each other as it is the case in the usual iterative process of the SQP method. Here, they can be optimized independently, the iterative behavior is decomposed. We replace the process of solving a QP by adding its KKT conditions (3.9), (3.10), (3.11), and (3.12) with corresponding dual variables ν as constraints. By incorporating them into the problem, we obtain a reformulation of the original bilevel problem that can be interpreted as solving the upper and the lower level simultaneously. Thus, all dependencies are now of an explicit nature, as desired.

Within the described process, we have to deal with two types of Lagrange multipliers. λ corresponds to the Lagrange function of the lower-level problem, while ν represents the dual variable that results from solving (LL-QP). However, due to the update step (3.8), we only need to add the Lagrange multipliers $\lambda_1, \dots, \lambda_{N_{\text{it}}}$ as new variables. They represent the sequence of Lagrange multipliers generated by the imitated SQP process. In particular, λ_1 is the initial guess for solving the first QP, $\lambda_2 = \nu_1$ represents both the solution of the first QP and the initial guess for the second QP, and so on.

Due to (3.7), we further replace each d_k by $y_{k+1} - y_k$ for $k = 1, \dots, N_{\text{it}} - 1$ to obtain a problem formulation that depends on x, $y_1, \dots, y_{N_{\text{it}}}$, and $\lambda_1, \dots, \lambda_{N_{\text{it}}}$. Thus, we can finally formulate another single-level reformulation of (O-BP), we call it *Decomposed Bilevel Problem (D-BP)*.

Problem 3.32 (D-BP)

Find variables $x, y_1, \ldots, y_{N_\mathrm{it}}, \lambda_1, \ldots, \lambda_{N_\mathrm{it}}$ that

$$
\begin{aligned}
\text{minimize} \quad & F(x, y_{N_\mathrm{it}}) \\
\text{subject to} \quad & G(x, y_{N_\mathrm{it}}) \leq 0_{n_G} \\
& g(x, y_k) + \nabla_y g(x, y_k) d_k \leq 0_{n_g},\ k = 1, \ldots, N_\mathrm{it} - 1 \\
& \nabla^2_{yy} \ell_k d_k + \nabla_y f_k + \nabla_y g(x, y_k)^\mathsf{T} \lambda_{k+1} = 0_{n_y},\ k = 1, \ldots, N_\mathrm{it} - 1 \\
& \lambda_{k+1}^\mathsf{T} (g(x, y_k) + \nabla_y g(x, y_k) d_k) = 0, \quad k = 1, \ldots, N_\mathrm{it} - 1 \\
& \lambda_k \geq 0_{n_g},\ k = 1, \ldots, N_\mathrm{it}
\end{aligned}
$$

with $d_k \coloneqq y_{k+1} - y_k$, $\ell_k \coloneqq \ell(x, y_k, \lambda_k)$, and $f_k \coloneqq f(x, y_k)$ for $k = 1, \ldots, N_\mathrm{it} - 1$.

Since the Hessian matrix $\nabla^2_{yy} \ell(x, y, \lambda)$ of the lower-level problem needs to be computed and this problem shall be solved using second-order optimization methods later in this work, we require that the lower-level functions be four times continuously differentiable.

The artificial constraints and variables aim to imitate an SQP algorithm, which itself aims to approximate KKT points. Consequently, the corresponding conditions are not explicitly part of the problem formulation, they are aimed to be satisfied implicitly. Hence, an alternative to the problem formulation above is to include the conditions

$$
\begin{aligned}
\nabla_y \ell(x, y_{N_\mathrm{it}}, \lambda_{N_\mathrm{it}}) &= 0_{n_y}, \\
g(x, y_{N_\mathrm{it}}) &\leq 0_{n_g}, \\
\lambda_{N_\mathrm{it}}^\mathsf{T} g(x, y_{N_\mathrm{it}}) &= 0, \\
\lambda_{N_\mathrm{it}} &\geq 0_{n_g}
\end{aligned}
$$

explicitly in addition to the existing ones. The condition

$$\lambda_{N_\mathrm{it}} \geq 0_{n_g}$$

is already included in (D-BP). This allows investigating another decomposed approach, for which a feasible point directly satisfies the lower-level KKT conditions. Herein, the number of constraints increases by $n_y + n_g + 1$. A *Decomposed-KKT Bilevel Problem (D-KKT-BP)* can be formulated as follows:

Problem 3.33 (D-KKT-BP)

Find variables $x, y_1, \dots, y_{N_{\mathrm{it}}}, \lambda_1, \dots, \lambda_{N_{\mathrm{it}}}$ that

$$
\begin{aligned}
\text{minimize} \quad & F(x, y_{N_{\mathrm{it}}}) \\
\text{subject to} \quad & G(x, y_{N_{\mathrm{it}}}) \leq 0_{n_G} \\
& g(x, y_k) + \nabla_y g(x, y_k) d_k \leq 0_{n_g}, \ k = 1, \dots, N_{\mathrm{it}} - 1 \\
& \nabla^2_{yy} \ell_k d_k + \nabla_y f_k + \nabla_y g(x, y_k)^{\mathsf{T}} \lambda_{k+1} = 0_{n_y}, \ k = 1, \dots, N_{\mathrm{it}} - 1 \\
& \lambda_{k+1}^{\mathsf{T}} (g(x, y_k) + \nabla_y g(x, y_k) d_k) = 0, \quad k = 1, \dots, N_{\mathrm{it}} - 1 \\
& \nabla_y \ell(x, y_{N_{\mathrm{it}}}, \lambda_{N_{\mathrm{it}}}) = 0_{n_y} \\
& g(x, y_{N_{\mathrm{it}}}) \leq 0_{n_g} \\
& \lambda_k \geq 0_{n_g}, \ k = 1, \dots, N_{\mathrm{it}}
\end{aligned}
$$

with $d_k \coloneqq y_{k+1} - y_k$, $\ell_k \coloneqq \ell(x, y_k, \lambda_k)$, and $f_k \coloneqq f(x, y_k)$ for $k = 1, \dots, N_{\mathrm{it}} - 1$.

This formulation is a combination of (KKT-BP) and (D-BP). The following investigations, however, consider (D-BP) exclusively. Still, this variation is practically tested and compared against other approaches within the corresponding sections in Chapter 4.

The proposed reformulation offers both numerical advantages and disadvantages. When comparing the different sets of constraints that are present in the original problem and in the decomposed one, compromises have to be made. In the original problem (O-BP), we can formulate the condition that y solves the lower-level problem as follows:

$$y \in \psi[x] \coloneqq \arg\min_y \{ f(x, y) \colon g(x, y) \leq 0_{n_g} \}.$$

When comparing this constraint to those introduced in (D-BP), we immediately notice that the original requirement of having a global solution (as one of possibly many solutions) has to be alleviated to finding a single local one (also compare Assumption 3.27). This is due to the local character of the embedded SQP scheme and may become necessary if the lower-level problem is not convex. Thus, for $x \in \mathbb{R}^{n_x}$ and an initial guess $(y_1, \lambda_1) \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_g}$, the variable $y_{N_{\mathrm{it}}}$ is uniquely given by applying N_{it} SQP iteration steps. Hence, there exists a function $\psi_{N_{\mathrm{it}}} \colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_g} \to \mathbb{R}^{n_y}$ with

$$y_{N_{\mathrm{it}}} = \psi_{N_{\mathrm{it}}}(x, y_1, \lambda_1).$$

Under specific regularity assumptions, the fulfillment of both types of constraints leads to the same solution in the limit: ψ can be represented by $\psi_{N_{\mathrm{it}}}$:

Remark 3.34 As long as the LICQ (Definition 2.6) and sufficient second-order conditions (Theorem 2.12) hold and (y_1, λ_1) is in the region of attraction of a lower-level solution $(y^\star, \lambda^\star)$, it holds that

$$\lim_{N_{\mathrm{it}} \to \infty} \psi_{N_{\mathrm{it}}}(x, y_1, \lambda_1) \in \psi[x]$$

for all $x \in \mathbb{R}^{n_x}$ (compare Theorem 2.16).

Under specific assumptions, we do not lose solutions when solving the reformulated problem.

Proposition 3.35 *Let $(\hat{x}, \hat{y})$ be a solution of (O-BP). If the lower-level problem is convex and the LICQ holds at $\hat{y}$ with respect to the constraint function $g(x, \cdot)$ for all x that fulfill $G(x, \hat{y}) \leq 0_{n_G}$, then there exists a λ such that*

$$(\hat{x}, \hat{y}, \ldots, \hat{y}, \lambda, \ldots, \lambda)$$

is a solution of Problem 3.32.

Proof Being a solution of (O-BP) implies that $\hat{y}$ is a lower-level solution. Thus, there exists a λ for which the corresponding KKT conditions

$$\begin{aligned} \nabla_y f(\hat{x}, \hat{y}) + \nabla_y g(\hat{x}, \hat{y})^\mathsf{T} \lambda &= 0_{n_y}, \\ g(\hat{x}, \hat{y}) &\leq 0_{n_g}, \\ \lambda^\mathsf{T} g(\hat{x}, \hat{y}) &= 0, \\ \lambda &\geq 0_{n_g} \end{aligned}$$

are fulfilled. With $d_k = y_{k+1} - y_k = \hat{y} - \hat{y} = 0_{n_y}$ for $k = 1, \ldots, N_{\mathrm{it}} - 1$, (D-BP) reduces to the KKT approach with additional redundant constraints. Under the given assumptions, this approach is known to be equivalent to the original problem (see for example [3]). □

The proposition reveals that redundancies in (D-BP) can occur. Multiple combinations of $y_1, \ldots, y_{N_{\mathrm{it}}-1}$ and $\lambda_1, \ldots, \lambda_{N_{\mathrm{it}}-1}$ may lead to the same values of $y_{N_{\mathrm{it}}}$ and $\lambda_{N_{\mathrm{it}}}$. Hence, we cannot expect to find strict local minima (compare Definition 2.3).

The problem dimensions depend on N_{it}, the number of lower-level SQP iteration steps: (D-BP) exhibits $n_G + (N_{\mathrm{it}} - 1)(n_g + n_y + 1) + n_g N_{\mathrm{it}}$ constraints and $n_x + N_{\mathrm{it}}(n_y + n_g)$ variables. Since one does not know in advance how many iteration steps the SQP algorithm needs to solve the lower-level problem, an intuitive way is to choose N_{it} large, which leads to high problem dimensions. However, the general problem formulation exhibits sparse structures (which will be examined later), independent of the specific problem at hand.

A clear advantage is that the difficult constraint (3.5) in the original problem formulation is replaced by a set of constraints that allows us to use standard NLP solvers. In particular, it is not necessary to invoke an external solver, since the lower-level problem is solved implicitly once the overall problem is solved.

Another drawback is the introduction of the $N_{\mathrm{it}} - 1$ constraint blocks of type

$$\begin{aligned} g + \nabla g d &\leq 0_{n_g}, \\ \lambda^{\mathsf{T}}(g + \nabla g d) &= 0, \\ \lambda &\geq 0_{n_g}, \end{aligned}$$

which transform the problem into an MPCC causing numerical difficulties. However, in analogy to the KKT approach (compare Problem 3.30), the above-mentioned constraints can be replaced by using FB functions. In practice, however, smoothed version would be applied in order to overcome the nonsmoothness caused by Φ. This allows formulating another version of the decomposed approach:

Problem 3.36 (D-FB-BP)

Find variables $x, y_1, \ldots, y_{N_{\mathrm{it}}}, \lambda_1, \ldots, \lambda_{N_{\mathrm{it}}}$ that

$$\begin{aligned} \text{minimize} \quad & F(x, y_{N_{\mathrm{it}}}) \\ \text{subject to} \quad & G(x, y_{N_{\mathrm{it}}}) \leq 0_{n_G} \\ & \nabla^2_{yy}\ell_k d_k + \nabla_y f_k + \nabla_y g_k^{\mathsf{T}} \lambda_{k+1} = 0_{n_y}, \quad k = 1, \ldots, N_{\mathrm{it}} - 1 \\ & \Phi\big((\lambda_{k+1})_i, -(g_k + \nabla_y g_k d_k)_i\big) = 0, \quad i = 1, \ldots, n_g, \\ & \qquad\qquad\qquad\qquad\qquad\qquad k = 1, \ldots, N_{\mathrm{it}} - 1 \\ & \lambda_1 \geq 0_{n_g} \end{aligned}$$

with $d_k := y_{k+1} - y_k$, $\ell_k := \ell(x, y_k, \lambda_k)$, $g_k := g(x, y_k)$, and $f_k := f(x, y_k)$ for $k = 1, \ldots, N_{\mathrm{it}} - 1$.

Avoiding the complementarity constraints can also be applied to the other approach, in which the lower-level optimality conditions are directly included:

Problem 3.37 (D-KKT-FB-BP)

Find variables $x, y_1, \ldots, y_{N_\mathrm{it}}, \lambda_1, \ldots, \lambda_{N_\mathrm{it}}$ that

$$
\begin{aligned}
\text{minimize} \quad & F(x, y_{N_\mathrm{it}}) \\
\text{subject to} \quad & G(x, y_{N_\mathrm{it}}) \leq 0_{n_G} \\
& \nabla^2_{yy} \ell_k d_k + \nabla_y f_k + \nabla_y g_k^\mathsf{T} \lambda_{k+1} = 0_{n_y}, \quad k = 1, \ldots, N_\mathrm{it} - 1 \\
& \nabla_y \ell(x, y_{N_\mathrm{it}}, \lambda_{N_\mathrm{it}}) = 0_{n_y} \\
& \Phi\big((\lambda_{k+1})_i, -(g_k + \nabla_y g_k d_k)_i\big) = 0, \quad i = 1, \ldots, n_g, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad k = 1, \ldots, N_\mathrm{it} - 1 \\
& \Phi\big((\lambda_{N_\mathrm{it}})_i, -(g(x, y_{N_\mathrm{it}}))_i\big) = 0, \quad i = 1, \ldots, n_g \\
& \lambda_1 \geq 0_{n_g}
\end{aligned}
$$

with $d_k \coloneqq y_{k+1} - y_k$, $\ell_k \coloneqq \ell(x, y_k, \lambda_k)$, $g_k \coloneqq g(x, y_k)$, and $f_k \coloneqq f(x, y_k)$ for $k = 1, \ldots, N_\mathrm{it} - 1$.

In practice, these formulations are applied sequentially: Φ is replaced by a smoothed version, for which a smoothing parameter will be driven to zero. This approach is described in detail in Section 4.5. For the remaining investigations within this part, however, we consider (D-BP) exclusively.

Remark 3.34 gives a statement about the behavior in the limit of the underlying SQP method. We are, however, interested in the numerical solution of bilevel problems, which implies fixing N_it to a finite number. Thus, we cannot expect the solutions to be exact. The choice of N_it is left to the user and is not trivial. In case the lower-level problem has a specific form, however, N_it is known beforehand:

Theorem 3.38 *If the lower-level problem (LL) is a convex QP, a solution of (D-BP) with $N_{it} = 2$ is a solution of (O-BP).*

Proof $N_\mathrm{it} = 2$ corresponds to embedding one SQP iteration into the problem. Since one step is sufficient to find a solution of convex quadratic programs, $y_{N_\mathrm{it}} = y_2$ solves the lower-level problem in case a feasible point is found. In addition, the constraints representing the iterative behavior reduce to the KKT conditions of the lower-level problem. They are sufficient, since the problem is convex. □

For arbitrary problem types, however, it is not possible to determine a suitable N_{it} beforehand. The number of required iterations depends not only on the initial guess, but also on the user's choice of solver options, such as optimality or feasibility tolerances. Still, there are some strategies for determining N_{it}:

- Solve the lower-level problem in advance with the implemented SQP algorithm and use the required number of iterations as lower bound for N_{it}.
- Start with $N_{\text{it}} = 2$ and solve (D-BP). Use the solution as an initial guess for another solver run with increasing values of N_{it}.[18]
- Use an arbitrarily large value for N_{it}.

Directly related is the search for criteria which guarantee that, for a given N_{it}, a solution corresponds to a solution of the lower-level problem. A suitable criterion is the search direction $d_k = y_{k+1} - y_k$, which vanishes, once a KKT point is found. This fact can be used to formulate the following:

Proposition 3.39 *Let $(x, y_1, \ldots, y_{N_{it}}, \lambda_1, \ldots, \lambda_{N_{it}})$ be a solution of (D-BP), in which the lower-level problem is assumed to have the same properties as in Proposition 3.35. Suppose that $y_{N_{it}-1} = y_{N_{it}}$ and $\lambda_{N_{it}-1} = \lambda_{N_{it}}$. Then $(x, y_{N_{it}})$ is a solution of (O-BP).*

This is proven by similar arguments as in the proof for Proposition 3.35 and based on the observation that a similar KKT formulation is obtained. In practice, however, one would consider $\|y_{N_{\text{it}}-1} - y_{N_{\text{it}}}\|_2 < \epsilon$ instead of $y_{N_{\text{it}}-1} = y_{N_{\text{it}}}$.

Example 3.40

We demonstrate several aspects of (D-BP) by investigating an example from [78]. For upper-level variables $x \in \mathbb{R}$ and lower-level variables $y \in \mathbb{R}$, the task is to

$$\begin{aligned} \text{minimize} \quad & \left(x + \frac{1}{2}\right)^2 + \frac{1}{2}y^2 \\ \text{subject to} \quad & -1 \leq x \leq 1 \\ & y \in \arg\min_v \quad \frac{1}{2}xv^2 + \frac{1}{4}v^4 \\ & \qquad \text{s.t.} \quad -1 \leq v \leq 1. \end{aligned}$$

A local optimal solution is given by $(x^\star, y^\star) = (-0.25, 0.5)$ and the corresponding upper-level function value is 0.1875. We solve this problem in formulation (D-BP)

[18] This strategy is applied in Section 4.5.

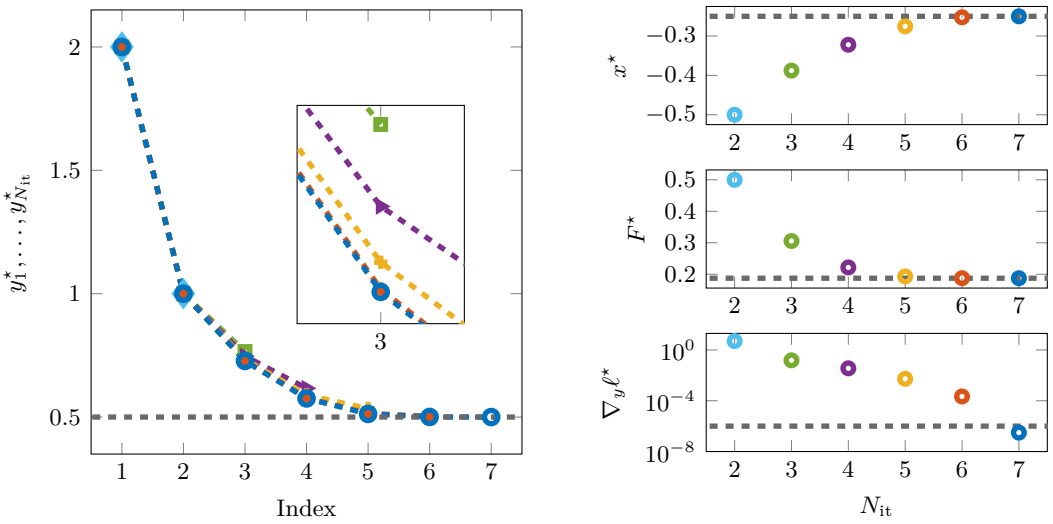


Figure 3.6: Influence of the number of imitated iteration steps N_{it} on solving the bilevel problem in Example 3.40 in the decomposed formulation. Dashed lines mark reference or tolerance (only bottom right) values. The abbreviations $F^\star := F\left(x^\star, y_{N_{\text{it}}}^\star\right)$ and $\nabla_y \ell^\star := \nabla_y \ell\left(x^\star, y_{N_{\text{it}}}^\star, \lambda_{N_{\text{it}}}^\star\right)$ are used.

using the initial guesses

$$\begin{aligned} x_{\text{ini}} &= -1, \\ y_{k\text{ini}} &= 2, \\ \lambda_{k\text{ini}} &= \begin{pmatrix} 5 & 5 \end{pmatrix}^\mathsf{T} \end{aligned}$$

for $k = 1, \ldots, N_{\text{it}}$. The NLP is solved by `WORHP`'s SQP algorithm for varying values of N_{it}, which is shown in Figure 3.6. Although `WORHP` terminates successfully for each use of $N_{\text{it}} = 2, \ldots, 6$, the solutions do not represent a solution of the lower-level problem. To verify this, we consider the lower-level stationarity condition $\nabla_y \ell(x^\star, {y_{N_{\text{it}}}}^\star, {\lambda_{N_{\text{it}}}}^\star) = 0$. When allowing $N_{\text{it}} = 7$ SQP iteration steps, the solution finally is a lower-level minimum (up to the used tolerance 10^{-6}). We can also observe that the solutions as well as their corresponding objective function value converge to their reference values. This example demonstrates that on the one hand, N_{it} has to be chosen carefully, and on the other hand, that a post-processing of a computed solution should be performed to check for lower-level solutions.[19]

[19] This example is published by Sch., Fliege, Flaßkamp, and Büskens in [98].

Initialization of (D-BP)

Due to the potentially large number of variables, there are several strategies for initialization. As mentioned earlier, one strategy for choosing N_{it} is to solve the lower-level problem in advance for given x_{ini}, y_{ini}, and λ_{ini}. The resulting values at the intermediate variables have to be stored and can be used to initialize (D-BP). This would allow starting close to a feasible solution. Alternatively, an interpolation between the initial guess and the solution can be applied instead of storing the values. An adaptive initialization used within a sequential solution strategy (see Subsection 4.5.3) may also be suitable. Still, finding an adequate guess is not trivial and depends on the given problem.

Computation of Derivatives

For large values of N_{it}, the resulting NLP (Problem 3.32) can easily become high-dimensional and difficult to solve, even if the corresponding original bilevel problem has only a few variables. However, the required derivatives have sparse structures that can be exploited by modern solvers.

The objective function depends only on the variables x and $y_{N_{\mathrm{it}}}$, which leads to the following structure of its gradient ∇F:

$$\nabla F = \begin{bmatrix} \times & & \times & \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{n_x + N_{\mathrm{it}}(n_y+n_g)}.$$

Due to the embedded iterative process in the constraints (which we summarize as C), the Jacobian matrix ∇C consists of three parts: a column block representing the dependency on the upper-level variable x and two series of column blocks for the iteration steps of y and λ, respectively. This is illustrated for $N_{\mathrm{it}} = 4$:

$$\nabla C = \begin{array}{c} \begin{array}{ccccccccc} x & y_1 & y_2 & y_3 & y_4 & \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 \end{array} \\ \left[\begin{array}{ccccccccc} \times & & & & \times & & & & \\ \hdashline & \times & \times & & & & & & \\ & \times & \times & & & \times & \times & & \\ & \times & \times & & & & \times & & \\ \hdashline & & \times & \times & & & & & \\ & & \times & \times & & & \times & \times & \\ & & \times & \times & & & & \times & \\ \hdashline & & & \times & \times & & & & \\ & & & \times & \times & & & \times & \times \\ \times & & & \times & \times & & & & \times \end{array}\right] \end{array} \begin{array}{l} \\ \\ k{=}1 \\ \\ \\ k{=}2 \\ \\ \\ k{=}3 \\ \\ \end{array}.$$

For optimization variables $z \coloneqq \begin{pmatrix} x^\mathsf{T} & y_1^\mathsf{T} \cdots y_{N_\mathrm{it}}^\mathsf{T} & \lambda_1^\mathsf{T} \cdots \lambda_{N_\mathrm{it}}^\mathsf{T} \end{pmatrix}^\mathsf{T}$ and Lagrange multipliers $\mu \in \mathbb{R}^{n_C}$, we denote the Lagrange function of (D-BP) as

$$\mathcal{L}(z, \mu) = F(x, y_{N_\mathrm{it}}) + \sum_{i=1}^{n_C} \mu_i C_i(z).$$

Its Hessian matrix has the following sparsity pattern:

$$\nabla_{zz}^2 \mathcal{L} = \begin{bmatrix}
\times & \times & \times & \times & \times & \cdots & \times & \times & \times & \times & \times & \times & \times & \times & \cdots & \times & \times & \times & \times \\
\times & \times & \times & & & & & & & & \times & \times & & & & & & & \\
\times & \times & \times & \times & & & & & & & \times & \times & \times & & & & & & \\
\times & & \times & \times & \times & & & & & & & \times & \times & \times & & & & & \\
\times & & & \times & \times & \ddots & & & & & & & \times & \times & \ddots & & & & \\
\vdots & & & & \ddots & \ddots & \ddots & & & & & & & \ddots & \ddots & \ddots & & & \\
\times & & & & & \ddots & \times & & & & & & & & \ddots & \times & & & \\
\times & & & & & & \times & \times & \times & & & & & & \times & \times & \times & & \\
\times & & & & & & \times & \times & \times & & & & & & & \times & \times & \times & \\
\times & & & & & & & \times & \times & & & & & & & & \times & \times & \\
\times & \times & \times & & & & & & & & & & & & & & & & \\
\times & \times & \times & \times & & & & & & & & & & & & & & & \\
\times & & \times & \times & \times & & & & & & & & & & & & & & \\
\times & & & \times & \times & \ddots & & & & & & & & & & & & & \\
\vdots & & & & \ddots & \ddots & \ddots & & & & & & & & & & & & \\
\times & & & & & \ddots & \times & & & & & & & & & & & & \\
\times & & & & & & \times & \times & \times & & & & & & & & & & \\
\times & & & & & & & \times & \times & & & & & & & & & & \\
\times & & & & & & & & \times & \times & & & & & & & & &
\end{bmatrix}.$$

Since we expect all functions to depend on x, the first row and the first column are dense. The large block consisting of structural zeros in the south-east corner is due to the Lagrange multipliers $\lambda_1, \ldots, \lambda_{N_\mathrm{it}}$, which enter the problem formulation linearly.

Exploiting the patterns[20] of $\nabla_z F$, $\nabla_z C$, and $\nabla_{zz}^2 \mathcal{L}$ influences the computation time, in particular for high-dimensional problems in combination with a large N_it. This aspect is illustrated in the following example:

[20] Sparse matrices, as introduced in Subsection 2.2.4, can be stored efficiently to save memory and computation time. In the *coordinate storage format*, a non-zero value is stored as a triplet consisting of the row and column indices and the value. By imposing certain restrictions, more efficient algorithms can be used for matrix operations. `WORHP` uses this format internally [106]. Users have to provide the coordinates of non-zero values.

Example 3.41

We consider the following bilevel problem (as in [6]): For upper-level variables $x \in \mathbb{R}^2$ and lower-level variables $y \in \mathbb{R}^2$,

$$\begin{aligned}
\text{minimize} \quad & -x_1^2 - 3x_2 - 4y_1 + y_2^2 \\
\text{subject to} \quad & x_1^2 + 2x_2 - 4 \leq 0 \\
& x_1,\, x_2 \geq 0 \\
& y \in \arg\min_v \quad 2x_1^2 + v_1^2 - 5v_2 \\
& \qquad\qquad \text{s.t.} \quad g(x, v) \leq 0_2 \\
& \qquad\qquad\qquad v_1,\, v_2 \geq 0
\end{aligned}$$

with

$$g(x, v) \coloneqq \begin{pmatrix} -x_1^2 + 2x_1 - x_2^2 + 2v_1 - v_2 - 3 \\ -x_2 - 3v_1 + 4v_2 + 4 \end{pmatrix}.$$

To demonstrate the effects of exploiting sparse structures, we solve this problem using the decomposed approach for different values of N_{it}. We consider four cases, all of them with respect to (D-BP).

1. Both the Jacobian J of the constraints and the Hessian H of the Lagrange function are assumed to be dense.
2. The Jacobian's sparsity pattern is exploited, the Hessian's is not.
3. The Jacobian is assumed to be dense, while the Hessian is sparse.
4. For both matrices, the sparsity patterns are exploited.

For each of these possibilities, we solve the problem using $N_{\text{it}} = 2, \ldots, 50$ and measure the overall computation time. We divide it by the number of required iterations the solver `WORHP` has to perform to eliminate this dependency. Hence, an average value per iteration is obtained and the actual minimum found (compare Example 3.40, where different values of N_{it} lead to different solutions) plays only a minor role. In this case, we can even observe convergence to the same minimum for each value of N_{it}. The results are visualized in Figure 3.7. For the purpose of a consistent comparison, each bar corresponds to a solver run in which the *relative error* to the reference objective function value is less than 5 % (for details on the relative error definition, we refer to Section 4.5). Hence, a missing bar represents an unsuccessful solver run and does not distort the result. Not surprisingly, the average computation time is large when structure exploitation is not used. In

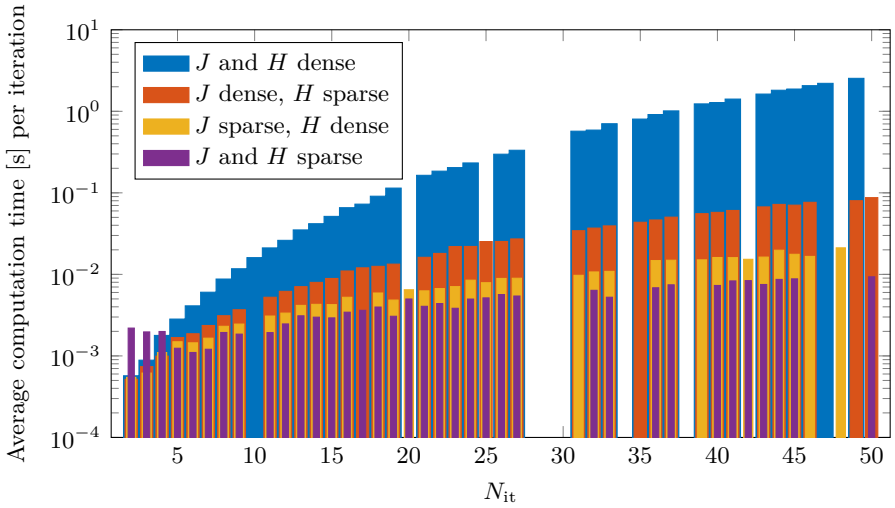


Figure 3.7: Influence of N_{it} on the computation time required to solve the bilevel problem in Example 3.41 for different levels of sparsity.

all other cases, however, the computation time can be kept at a desirable low level despite increasing problem dimensions. Only for small values of N_{it} and full sparsity exploitation does the average computation time unexpectedly deviate from the present trend. For comparison, the reduced approach here needs 1.83×10^{-2} s, the KKT approach 3.7×10^{-4} s on average per iteration.

Literature Review

This approach is inspired by the decomposed formulation within the field of dynamical parameter identification. To the best of the author's knowledge, such a reformulation of (O-BP) has not been investigated before. An initial reference by the author can be found in [98].

3.4 Connections

Both parameter identification and bilevel programming fit into the framework of decomposition (as introduced in Section 3.1). Here, we provide a summary of the respective connections and outline similarities and differences between them.

Formulation	Number of variables	Number of constraints
(R-DPIP)	$n_p + n_q$	0
(MS-DPIP)	$n_p + N_\mathrm{s} n_q$	$(N_\mathrm{s} - 1) n_q$
(D-DPIP)	$n_p + N_\mathrm{d} n_q$	$(N_\mathrm{d} - 1) n_q$
(R-BP)	n_x	n_G
(KKT-BP)	$n_x + n_y + n_g$	$n_G + n_y + 2n_g + 1$
(KKT-FB-BP)	$n_x + n_y + n_g$	$n_G + n_y + n_g$
(D-BP)	$n_x + N_\mathrm{it}(n_y + n_g)$	$n_G + (N_\mathrm{it} - 1)(n_g + n_y + 1) + N_\mathrm{it} n_g$
(D-KKT-BP)	$n_x + N_\mathrm{it}(n_y + n_g)$	$n_G + N_\mathrm{it}(n_g + n_y) + N_\mathrm{it} n_g + N_\mathrm{it} - 1$
(D-FB-BP)	$n_x + N_\mathrm{it}(n_y + n_g)$	$n_G + (N_\mathrm{it} - 1)(n_y + n_g) + n_g$
(D-KKT-FB-BP)	$n_x + N_\mathrm{it}(n_y + n_g)$	$n_G + N_\mathrm{it}(n_y + n_g) + n_g$

Table 3.3: Problem dimensions after transcription (parameter identification) or single-level reduction (bilevel programming).

3.4.1 Relation to the Concept

The problem class of dynamical parameter identification is a prototype for decomposition. Here, the original problem is an infinite-dimensional optimization problem involving an ODE which is parameterized in terms of model parameters and initial values, both of which serve as primary variables. The resulting state trajectories are secondary variables. After transcription, one can formulate a reduced variant, a decomposed variant, and direct multiple shooting as a mixture of the two. The complexity lies in the numerical solution of the ODE, which can be embedded in the problem in several ways.

The problem class of bilevel programming also fits seamlessly into the framework presented here. In this context, the original problem is a bilevel problem that involves a subordinate optimization problem parameterized in upper-level variables that serve as primary variables. The corresponding solution, required in the upper level, represents the secondary variables. For this problem, one can — as well — formulate a reduced variant and two decomposed variants of different complexity.

A comparison of problem dimensions is given in Table 3.3, while the findings and relations to the concept are summarized in Table 3.4.

	Parameter Identification for Dynamical Systems	
Original problem	O-DPIP	(Problem 3.9)
Expression ψ	Numerical solution of IVP	
	$\dot{q}(t) = s(t, q(t), p)$, $t \in [t_a, t_b]$, $q(t_a) = q_a$	
Primary variables z	Initial state values q_a and model parameters p	
Secondary variables w	Discretized state trajectories $\tilde{q}$	
Formulations	R-DPIP	(Problem 3.14)
	MS-DPIP	(Problem 3.18)
	D-DPIP	(Problem 3.21)
	Bilevel Programming	
Original problem	O-BP	(Problem 3.24)
Expression ψ	Numerical solution of NLP	
	$\min_y \ f(x, y)$ s.t. $g(x, y) \leq 0_{n_g}$	
Primary variables z	Upper-level variables x	
Secondary variables w	Lower-level variables y	
Formulations	R-BP	(Problem 3.28)
	KKT-BP	(Problem 3.29)
	KKT-FB-BP	(Problem 3.30)
	D-BP	(Problem 3.32)
	D-KKT-BP	(Problem 3.33)
	D-FB-BP	(Problem 3.36)
	D-KKT-FB-BP	(Problem 3.37)

Table 3.4: Connections between the presented problem classes and the concept of decomposition.

3.4.2 Comparison

The investigated problem classes relate to the concept similarly. In the reduced formulation, an iterative procedure is applied in the background, invisible to the optimization algorithm. For this, standard implementations can be used, which makes an integration into the overall problem convenient. The complexity of this evaluation depends on the specific problem. If the underlying routine is very complex, the total effort for solving the overall problem can quickly become large. In the decomposed formulation, the computational cost depends mainly on the chosen number of discretization and iteration points. However, in both problem classes,

sparse matrices resulting from the artificial constraints can be exploited to reduce the computational effort. Not only the computational cost, but also the accuracy of the solution of the underlying task is influenced by this number. For the numerical solution of an ODE, a sufficiently small step size (defined through the number of discretization points) needs to be given in order to obtain an adequate approximation of the solution. Too few points can quickly lead to undesirably high truncation errors. In bilevel programming, the number of iteration steps available to solve the lower-level problem can influence the solution's accuracy as well, which is demonstrated in Example 3.40.

Despite the obvious similarities, closer inspection reveals many differences. The intermediate values — either the discretized state trajectories or the imitated lower-level iteration steps — are used in the corresponding objective function, so proper initialization is beneficial. While in parameter identification all intermediate values are used and their initialization is even intuitive, decomposition in bilevel programming is different: Only the last iteration point, as an approximation to the lower-level solution, is used in the objective function; all other intermediate values occur only in the constraints. Therefore, their initialization may not be as essential as in parameter identification. Providing useful initializations is also not an easy task.

Chapter 4

Applications and Numerical Results

This chapter is a collection of examples that explore the idea, reveal characteristics, and analyze the behavior of decomposition methods for parameter identification and bilevel programming. The presented applications build upon the decomposition techniques introduced in Chapter 3. We present a mixture of examples, from illustrative academic examples with extensive numerical evaluations to problems involving real-world data up to testing methods on a full library of academic problems. Starting in the field of parameter identification, extensive numerical comparisons of transcription methods are presented. The academic example of a mathematical pendulum (Section 4.1) reveals several interesting properties, while similar experiments are conducted on a real-world example of an industrial robot (Section 4.2). For the area of bilevel programming, the region of attraction is in the focus of two examples (Section 4.3 and Section 4.4). Since the decomposed approach marks a novel development in this area, it is tested on a library of problems in Section 4.5. The chapter closes with the idea of a homotopy-optimization approach for parameter identification (Section 4.6).

4.1 Parameter Identification for a Pendulum

In this section, we compare the presented transcription techniques (reduced, multiple shooting, decomposed formulation) against each other from a numerical point of view. In particular, we study the algorithmic behavior when the NLP solver `WORHP` is applied to solve the parameter identification problem initially introduced in Example 3.10. The comparison criteria are manifold. Although computation times and convergence rates always play an important role when comparing different setups, the focus of this study is on the distribution of computed solutions. In particular, we are interested in making statements about the occurrence of different local minima.

We aim to answer the question which formulation is more likely to generate desired solutions and give reasons why this may be the case.

At first, it has to be distinguished between the question of why a given problem possesses multiple local solutions and why an algorithm in combination with the chosen problem formulation converges to a particular solution. For the first question, there are many possible answers.

- Local minima can emerge from long time horizons coupled with an oscillatory system behavior.
- Local minima can originate from a problem's ill-posedness.
- Local minima occur due to redundancies within the dynamical model.

This list is not exhaustive. Parameter identification problems are known to have multiple local solutions, as demonstrated by Moles et al. [80] using a biochemical example. In general, the latter question can be answered by the locality of the solution algorithm, whose behavior is mainly influenced by a user-given initialization.

Many research works deal with comparing transcription techniques for dynamical parameter identification problems in a numerical way. Peifer and Timmer [91] compare the reduced formulation against a multiple shooting approach for two biochemical problems. They consider the number of convergent runs, the occurrence of the global minimum, and the computational load. Hamilton [57] compares the number of convergent runs for different noise levels for the above-mentioned methods for two chaotic systems.

4.1.1 Comparison Setup

We continue with the examples already presented (Examples 3.10, 3.15, and 3.22) and define the region of interest to be the rectangle

$$[p_{\text{low}}, p_{\text{upp}}] \times \left[\tilde{q}_{2_{\text{low}}}^{[t_1]}, \tilde{q}_{2_{\text{upp}}}^{[t_1]}\right] = [0.2, 2] \times [-1, 1]. \tag{4.1}$$

The second initial value $\tilde{q}_2^{[t_1]}$ can be varied since measurements only exist for the first state. The first initial value is fixed, mainly for visualization purposes. For the following comparisons, 1000 initial guesses are chosen randomly from the region in (4.1). They are obtained by *latin hypercube sampling*, which has the property

of giving a representative selection while having a moderate sampling size. This is realized using the `MATLAB` routine `lhsdesign`[1].

The investigation in Example 3.22 demonstrates that the initialization of parameter identification problems can heavily influence the algorithmic behavior and also, to which solution an algorithm converges. To make quantitative statements about the distribution of solutions and to compare the presented transcription methods consistently, several initialization strategies are considered. The numerical analyses are then of Monte–Carlo-type and aim at finding the average behavior of a method by performing numerous experiments using different initial guesses that sufficiently cover the region of interest.

4.1.2 Initialization Strategies

At first, the unmeasured initial state values $\tilde{q}|_{\mathcal{J}^{\mathrm{c}}\,\mathrm{ini}}^{[t_1]}$ and initial parameters p_{ini} are generated randomly by the previously described approach. To initialize $\tilde{q}|_{\mathcal{J}\,\mathrm{ini}}^{[t_1]}$, we use the corresponding measurement, which is in general an adequate choice. In the decomposed formulation as well as in the multiple shooting approach, the discretized trajectories (or parts thereof in the latter case) also need to be initialized. For this purpose, we make use of the following strategies:

Strategy I All the discretized state values $\tilde{q}^{[t_i]}$, $i = 2, \ldots, N_{\mathrm{d}}$ (respectively $\tilde{q}^{[\tau_i]}$, $i = 2, \ldots, N_{\mathrm{s}}$) are obtained by numerically integrating the ODE with initial values and parameters as described above. The integration method is identical to the one used in the optimization process. For comparison purposes, this is the most intuitive strategy.

Strategy II The measurements are used for initialization. If there are fewer measurements than discretization points ($N_{\mathrm{m}} < N_{\mathrm{d}}$), a linear interpolation is used to initialize the intermediate discretized points. Unmeasured state variables are initialized by the outcome of a forward-integration.

Strategy III The only difference to Strategy II is that unmeasured state variables are initialized by the initial value $\tilde{q}|_{\mathcal{J}^{\mathrm{c}}}^{[t_1]}$.

[1] A documentation of `lhsdesign` can be found at `https://de.mathworks.com/help/stats/lhsdesign.html`.

Variables	**Initial guess**		
p	Random		
$\tilde{q}\big\vert_{\mathcal{J}}^{[t_1]}$	Measurements		
$\tilde{q}\big\vert_{\mathcal{J}^c}^{[t_1]}$	Random		
	Strategy I	**Strategy II**	**Strategy III**
$\tilde{q}\big\vert_{\mathcal{J}}^{[t_i]}$, $i = 2, \ldots, N_\mathrm{d}$	Integrated	Measurements	Measurements
$\tilde{q}\big\vert_{\mathcal{J}^c}^{[t_i]}$, $i = 2, \ldots, N_\mathrm{d}$	Integrated	Integrated	Constant $\tilde{q}\big\vert_{\mathcal{J}^c}^{[t_1]}$

Table 4.1: Initialization strategies for solving the pendulum parameter identification problem.

On NLP level, the Lagrange multipliers also need to be initialized. Although WORHP offers a heuristic approach, we initialize them all with the constant zero, which is a common choice. A summary of initialization strategies is given in Table 4.1. Although shooting methods have the explicit advantage that a solution of the dynamical system can be arbitrarily well approximated, we restrict ourselves to using the same step size as in the full discretization approach. This allows us to compare the accuracy of the computed solutions when the same integration scheme is used in both formulations. In the following examples, we highlight specific features and algorithmic behavior.

4.1.3 Local Solutions

As there are only two primary variables, the shape of the objective function can be visualized (see Figure 4.1). In comparison to the one with only one variable (in Figure 3.3), the overall shape does not change qualitatively. Several valleys again reveal the existence of multiple local solutions.

For each random point in the specified initialization area, we solve the problem in its reduced version to obtain an overview of all solutions attained. We do not only obtain the number of attained minima, but also approximations of the regions of attraction of each minimum. The outcome, visualized in Figure 4.2, is not surprising. The initial parameter guess strongly influences the algorithmic behavior and consequently, it determines to which local solution the algorithm converges. Individual solutions may deviate from the expected position due to numerical inaccuracies or algorithm-specific features. In total, the regions of attraction approximately agree with the ones presumed from Figure 4.1.

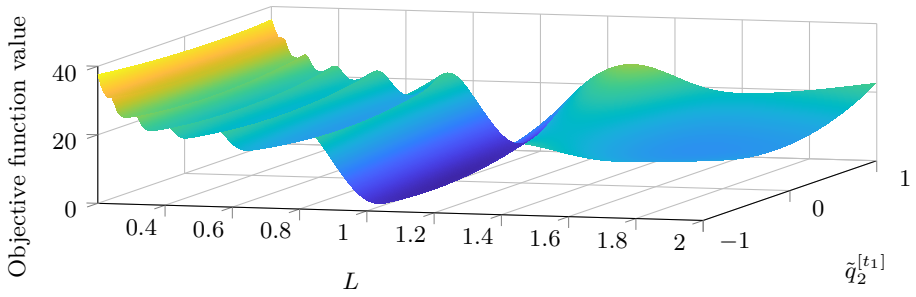


Figure 4.1: Shape of the objective function using the reduced formulation of the pendulum example.

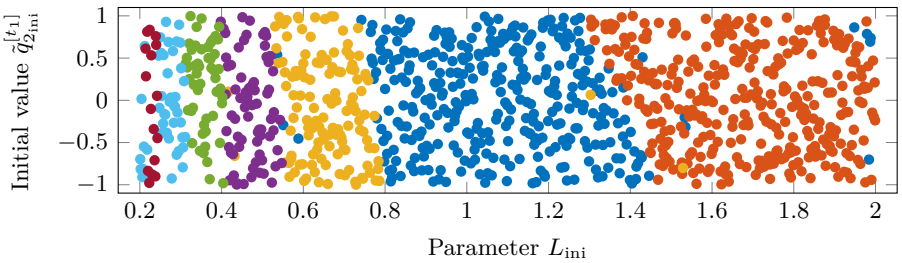


Figure 4.2: Regions of attraction of local solutions (indicated by color) using the reduced formulation.

In a first step, we directly compare the reduced problem with its decomposed counterpart, and again we are mainly interested in the regions of attraction. We use both algorithms offered by `WORHP` with standard configurations with the modifications listed in Table 2.1. Derivatives are computed by finite differences. The corresponding results are summarized in Table 4.2. When applying Strategy I — using initial trajectory guesses resulting from an integration with the reduced problem initializations — the results are qualitatively similar. Thus, simply increasing the problem dimensions does not lead to any advantages in finding better solutions. This is true for both algorithms, the differences are only minimal. With Strategy II — initializing states with measurements if possible — the higher problem dimensions pay off: the solution corresponding to the nominal value $L = 1$ is always obtained with the IP algorithm. This makes the optimization independent of the choice of the initial parameter. The same results are obtained for Strategy III. The SQP algorithm produces qualitatively similar results with only a few deviations. In particular, there are a few unsuccessful runs for all formulations except when using Strategy I.

In summary, decomposing the problem by adding dimensions and embedding the inner

Problem formulation	Local minimum attained (in %, sorted by objective function value)								
WORHP IP	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	†
Reduced	34.5	33.8	13.2	7.2	5.1	4.5	1.7	0	0
Decomposed (I)	33	35.4	12.9	7.4	5.1	3.7	2.5	0	0
Decomposed (II)	100	0	0	0	0	0	0	0	0
Decomposed (III)	100	0	0	0	0	0	0	0	0
WORHP SQP	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	†
Reduced	30.6	33.2	13.5	5.9	6.9	2.8	1.2	0.1	5.8
Decomposed (I)	33.2	35.1	13	7.4	5.1	3.7	2.5	0	0
Decomposed (II)	84.7	0	0.2	3.8	5.1	3.4	2.3	0	0.5
Decomposed (III)	91.9	0	2	2.6	0.5	2.3	0	0	0.7

Table 4.2: Frequencies of convergence to different local solutions for the reduced and decomposed formulation with different initialization strategies. Unsuccessful runs are marked by a †.

integration process into the problem in combination with an intelligent initialization enables the solver to overcome local, undesired solutions.

Next, we investigate the role that multiple shooting plays in this numerical analysis. In particular, we are interested in the number of shooting nodes one has to introduce to attain similar results as in the fully decomposed variant. The multiple shooting procedure is therefore implemented with the purpose of a consistent comparison (as explained in full details in Subsection 3.2.4):

- Variables at shooting nodes are initialized with respect to the strategies from Table 4.1.
- For the numerical integration of the ODE on the shortened time intervals, the same scheme as for single shooting with identical step size is used.

Thus, we focus on the influence of additional shooting nodes on the algorithmic behavior of a solver and again, we accept the fact that more accurate solutions might be computable when a more accurate ODE solution is obtained between shooting nodes. In principle, we could add up to $N_\mathrm{d} - 1$ shooting nodes, observe the outcome, and study the algorithmic behavior. Our experiments, however, indicate that this is not necessary. Figure 4.3 shows the regions of attraction for different values of N_s. One shooting node corresponds to the reduced formulation (compare Figure 4.2). By successively increasing N_s, one can observe that the global minimum is attained

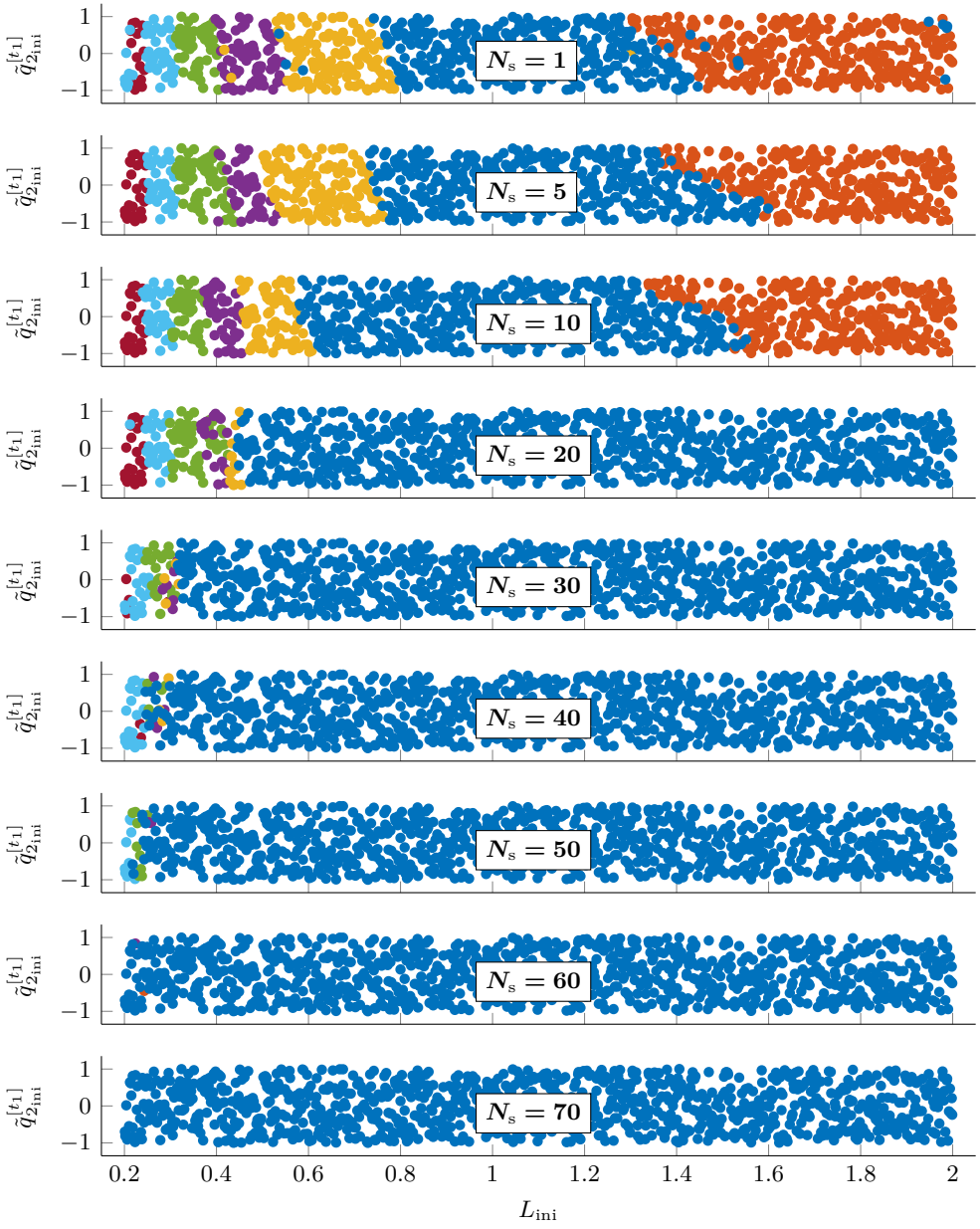


Figure 4.3: Regions of attraction of local solutions (indicated by color) of the pendulum example for multiple shooting using initialization strategy II.

more often, while the regions of attraction of the other local solution decrease in their size and finally vanish for $N_s = 70$. Hence, the parameter identification becomes independent of the initialization of the primary variables. Due to the increased number of optimization variables, the shape of the reduced objective function is only valid along a feasible trajectory. Further, these variables are initialized by the corresponding measurement values, which allows the solver to remain close to the measurements during the integration.

In summary, local and undesired solutions can be overcome by adding shooting nodes with corresponding initialization. It can also be observed that in rare cases, the solver is not able to converge to a solution. Such an outcome can have different reasons, but further experiments indicate that this can be avoided by adjusting solver-specific settings.

4.1.4 Computational Characteristics

Due to highly enlarged problem dimensions, it is reasonable to investigate the computation time. The reduced formulation benefits from small problem dimensions and therefore, the NLP can be solved quickly. The largest fraction of the computation time, however, is spent on repeatedly solving the ODE. This is avoided in the decomposed variant, but enlarged problem dimensions come into play. In this specific example, the solver has to handle 4003 variables with 4002 constraints. Thus, the Hessian matrix of the corresponding Lagrange function has 16024009 entries. Due to the sparsity structure, most of them are structural zeros and can therefore be neglected.

The computation times are compared in the left part of Figure 4.4. We only take those optimization runs into account that converge to the best found solution. For the purpose of a consistent comparison, we exclude those runs that started at a point from which the other investigated method did not converge, respectively. In other words, we use the intersection of initial guesses that lead to computing the desired solution. In the median, the reduced formulation only needs approximately 0.03 s respectively 0.04 s until the global minimum is found and also the interquartile ranges (IQRs) are favorably low. Although the problem dimensions are much larger in the decomposed formulation, the required computation time is only marginally increased. The IP algorithm needs 0.18 s, while the SQP method needs 0.29 s in the median. It is not surprising that the reduced approach converges quickly as it only converges for initial guesses in the close vicinity of the global minimum.

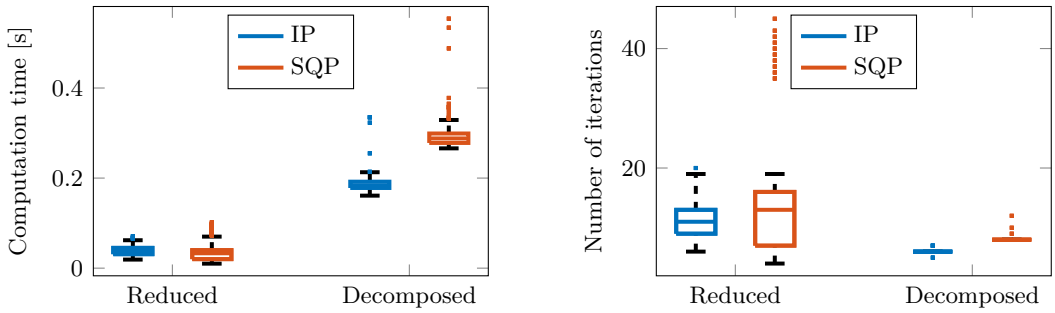


Figure 4.4: Computational characteristics of the parameter identification for the pendulum example.[2]

When considering the average number of major iterations (right part in Figure 4.4), a clear advantage of the decomposed approach can be observed. Not only are the median numbers of iterations (six and eight) smaller than those of the reduced approach (11 and 13), but also the IQRs. The reduced method shows scatter, while the decomposed variant yields stable values. In other words, decomposition stabilizes the solution process. This can be beneficial if dynamical parameter identification problems have to be solved repeatedly in online settings. Here, a practitioner relies upon a uniform solution process, time-consuming and unexpected iterative behavior should be avoided while an adequate solution quality should still be maintained.

Regarding the algorithm, we can conclude that the IP method of `WORHP` is preferable for this example, although the differences are far from being significant.

4.2 Parameter Identification for a Robotic System

Parameter identification with simple or academic underlying models can manifest interesting properties, as shown in the previous section. For more complex models originating from the real applications, parameter identification can become a challenging task, especially for multiple model parameters being identified simultaneously. In this section, the transcription methods are applied to solve parameter identification problems for a multi-link robotic system. The aim is to compare these methods

[2]The box plot diagrams used in this chapter (Figures 4.4 and 4.17) share the following conventions: The boxes range from the 25^{th} to the 75^{th} sample percentile, their distance is called the interquartile range (IQR). The middle line denotes the sample median. The whiskers extend the IQR by a factor of 1.5, all other points are assumed to be outliers.

(a) Idealized robotic system. **(b)** Industrial robot DENSO VS-050.

Figure 4.5: Robotic systems.

with respect to robustness and their ability to find desired solutions. The following investigations originate from two conference papers:

[95] K. Schäfer, M. Runge, K. Flaßkamp, and C. Büskens. "Parameter Identification for Dynamical Systems Using Optimal Control Techniques". In: *2018 European Control Conference (ECC)*. 2018, pp. 137–142. DOI: 10.23919/ECC.2018.8550045

[96] K. Schäfer, K. Flaßkamp, and C. Büskens. "A Numerical Study of the Robustness of Transcription Methods for Parameter Identification Problems". In: *Proceedings in Applied Mathematics and Mechanics* 18.1 (2018), e201800101. DOI: 10.1002/pamm.201800101

First, we consider an idealized robotic system (see Figure 4.5a). After that, the methods are validated on a real-world application (see Figure 4.5b).

4.2.1 Idealized Example

To demonstrate and analyze different transcription methods, an academic example of an idealized robotic system with two degrees of freedom is used. Both links of the robot have cylindrical shapes and are connected via joints that can rotate as illustrated in Figure 4.5a. We assume that the joints can be controlled externally, for example by torques or currents.

To derive the equations of motion, we use the Euler-Lagrange approach. The joint angles $\theta = (\theta_1, \theta_2)$ serve as generalized coordinates and the corresponding joint angular velocities $\dot{\theta} = \left(\dot{\theta}_1, \dot{\theta}_2\right)$ as their derivatives. We formulate the system's

Lagrangian L_r as the difference between kinetic and potential energy in the following way:

$$L_\mathrm{r}\Big(\theta,\dot{\theta}\Big) = \frac{1}{2}\sum_{i=1}^{2} m_i \Big\|\dot{S}_i\Big\|^2 + \frac{1}{2}\sum_{i=1}^{2} \omega_i^\mathsf{T}\mathcal{I}_i\omega_i - \sum_{i=1}^{2} m_i g S_{i,z}.$$

Herein, m_1 and m_2 are the masses of the links, $S_i = S_i(\theta)$ is the position of the center of mass of link i relative to the base frame, $\omega_i = \omega_i\Big(\dot{\theta}\Big)$ is the angular velocity of link i relative to the base frame, $\mathcal{I}_i$ is the inertia tensor of link i, and g is the gravitational constant. The dynamical behavior is described by forced Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial L_\mathrm{r}}{\partial\dot{\theta}} - \frac{\partial L_\mathrm{r}}{\partial\theta} = f_\mathrm{r} \tag{4.2}$$

with the external forcing term

$$f_\mathrm{r}\Big(\dot{\theta}, I\Big) = -\kappa_1 \tanh\Big(a\dot{\theta}\Big) - \kappa_2\dot{\theta} + \kappa_3 I,$$

in which $I = (I_1, I_2)$ represents the currents flowing through the joint motors. After calculating the relevant terms in (4.2), this equation can be written in the common form

$$M_\mathrm{r}(\theta)\ddot{\theta} = F_\mathrm{r}\Big(\theta,\dot{\theta}\Big) + f_\mathrm{r}\Big(\dot{\theta}, I\Big). \tag{4.3}$$

As these equations are standard formulations for robotic motions, we abstain from a detailed derivation of the dynamics and instead refer to [83].

Equation (4.3) is highly nonlinear due to the complex dynamics in this robotic system. It can also be seen that all model terms (M_r, F_r, and f_r) involve time-independent (physical or modeling) parameters, which we collect in the parameter vector $p \in \mathbb{R}^{n_p}$. All in all, there are $n_p = 19$ parameters that can be identified. To create artificial measurements, an arbitrary set of parameters (see Table 4.3) and arbitrary — yet meaningful — inputs I for the currents are chosen. The system is then simulated for a given time interval of three seconds with a higher-order integration method in combination with an adequate step size.

The goal is to simultaneously identify the parameters collected in p by using the methods introduced in Section 3.2. The *identifiability* of p, however, cannot be guaranteed and would have to be checked beforehand. In fact, it is likely that this is not the case due to parameter redundancies. We refrain from this check in the

Model parameter	Symbol	Link 1	Link 2	Unit
Mass	m	6	23	kg
Center of mass: x	s_x	0	0	m
Center of mass: y	s_y	0	0	m
Center of mass: z	s_z	—	0.2875	m
Moment of inertia: x	$\mathcal{I}_x$	—	0.6912	kgm^2
Moment of inertia: y	$\mathcal{I}_y$	—	0.6912	kgm^2
Moment of inertia: z	$\mathcal{I}_z$	0.03	0.115	kgm^2
Viscous friction factor	κ_1	1	2	Nm/s
Dry friction factor	κ_2	8	3	Nm
Current factor	κ_3	0.7	0.4	—
Modeling factor	a	50	50	—

Table 4.3: Nominal model parameters used for the idealized robotic system.

following, since we focus on comparing methods instead of finding a realistic set of parameters.

4.2.2 Comparison of Transcription Methods

In the following, the setup of the parameter identification problem is described: We choose to approximate the simulated joint angles and joint angular velocities. The model parameters are bounded by box constraints: $p_{i\text{low}} \leq p_i \leq p_{i\text{upp}}$ for each $i \in \{1, \ldots, n_p\}$. The measurements serve as initial guesses for the state variables, whereas the initial model parameters are set arbitrarily, but in their respective order of magnitude. To solve the optimization problems, second-order information is approximated by using BFGS methods provided by the solver. To make computations comparable with each other, we use the Explicit Euler method with the constant step size $h = 0.01$ in all the applied methods.

Figure 4.6 shows the measurements and the state trajectories resulting from solving the NLPs originating from the multiple shooting method with five shooting nodes as well as the full discretization approach. Both of the methods converge to a local minimum, but the solution by full discretization obviously (objective value of 6.5×10^{-2} against 5.016) finds a better one. It can be seen that the multiple shooting approach only reproduces the general dynamical behavior of the system, while the full discretization approach is more accurate. An explanation is that

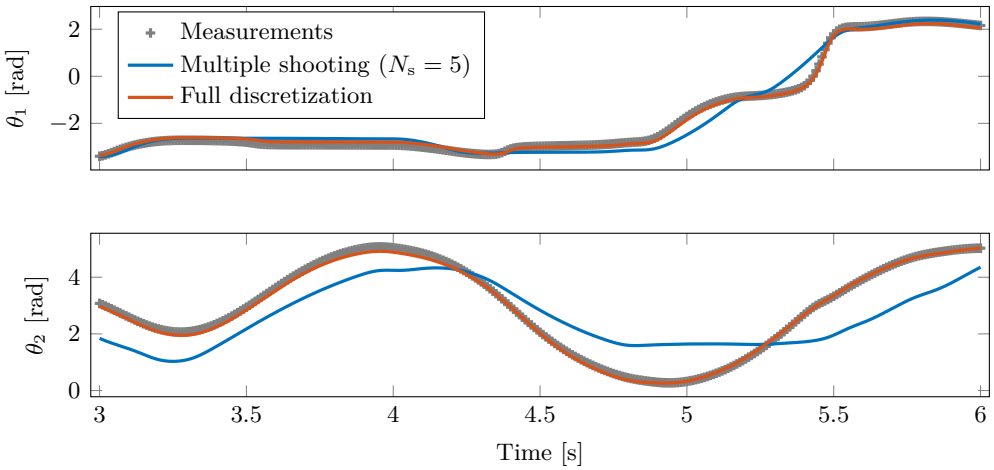


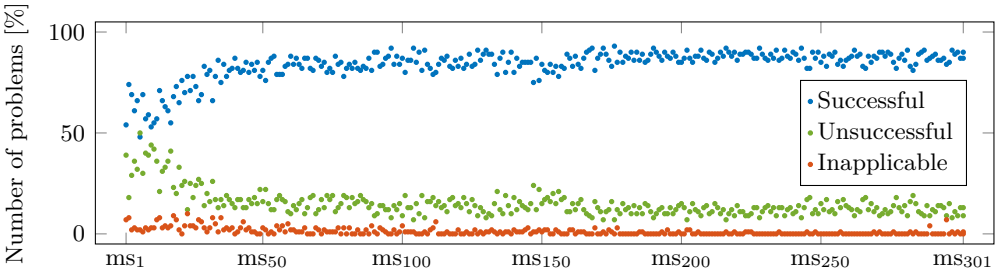
Figure 4.6: Solution trajectories obtained using full discretization and multiple shooting with 5 shooting nodes for the idealized robotic system.

multiple shooting converges to a local solution, while the other method is able to converge to a desired one.

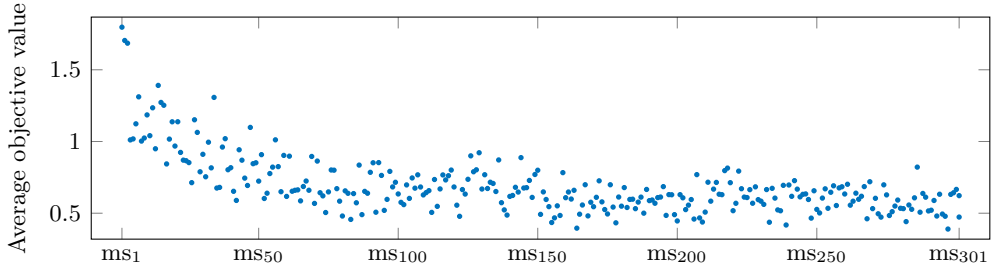
These results only cover a specific use case, namely for a single given initialization. For different initial guesses, the solutions might be qualitatively different. To make more general statements, we aim to have a closer look at the performance of multiple shooting regarding the fitting quality and also, to which extent the different methods tend to converge to a solution. Thus, we are interested in both the robustness with respect to different initializations and the average solution quality a method is able to produce.

To investigate the methods for robustness, experiments of Monte–Carlo-type are conducted. The initial model parameters, which are given to the solver, are sampled randomly. The initial guesses for the remaining optimization variables (in the case of multiple shooting and full discretization) are chosen to coincide with the corresponding measurement values. We solve each NLP using multiple shooting (ms_k[3]) with k shooting nodes for $k = 1, \ldots, 301$[4], and full discretization. The optimization outcomes are divided into three groups: *successful* meaning that a local minimum is found; *inapplicable solution* meaning that a minimum is found, but the corresponding objective function value is larger than a given threshold, indicating that the solution

[3]Single shooting is expressed as ms_1.
[4]This is due to a three-second time interval with a step size of 0.01.



(a) Robustness.



(b) Solution quality.

Figure 4.7: Influence of the number of shooting nodes on robustness and solution quality for the robot example.

cannot be meaningful; and *unsuccessful* meaning that the solver does not find a solution of the NLP.

The results are shown in Figure 4.7a, in which a clear tendency can be observed: an introduction of shooting nodes leads to a more robust behavior in the sense that the NLP solver was able to find a local minimum more often. With only a few shooting nodes, about half of all problems can be solved. However, the robustness increases and stabilizes already with about 50 shooting nodes. With even more nodes, about 90 % of all problems can be solved. Consequently, the unsuccessful runs behave contrarily. Inapplicable solutions occur in about 10 % of all problems when only a few shooting nodes are considered. These numbers also stabilize until only a few outliers produce these solutions using many shooting nodes.

Finding a local minimum in parameter identification does not automatically mean approximating the measurements well. Here, we observe that more robust methods generally find solutions that approximate measurements better (see Figure 4.7b). When only successful runs are considered, the average objective function value

shrinks from about 1.8 for single shooting to 0.39 for 297 shooting nodes. Again, a clear trend can be seen to favor increasing the number of shooting nodes.

As already mentioned, it is not guaranteed that the parameters (or a subset of them) are identifiable, possibly due to parameter redundancies or measurements that carry too little information. Therefore, it is possible to find combinations of parameters that lead to close or even identical values of the objective function. This also favors the occurrence of multiple local solutions. However, the aspect described here does not degrade the findings made in this section, since the focus is on the problem formulations and their ability to find desired solutions.

4.2.3 Sparsity

The application of multiple shooting or full discretization leads to sparse matrices that describe the NLP-specific derivatives. To illustrate this behavior dependent on the number of shooting nodes, we make use of the relative density of a matrix δ_{rel} introduced in Subsection 2.2.4. It depends on the number of non-zero elements n_{nz} and the total number of entries. If we denote the corresponding objective function by F_{ms} and the constraints' function by C_{ms}, the relative densities of ∇F_{ms} and ∇C_{ms} read

$$\delta_{\text{rel}}(\nabla F_{\text{ms}}) = \frac{n_{\text{nz}}(\nabla F_{\text{ms}})}{n_z}$$

and

$$\delta_{\text{rel}}(\nabla C_{\text{ms}}) = \frac{n_{\text{nz}}(\nabla C_{\text{ms}})}{n_z n_C},$$

and for the Hessian matrix H of the Lagrange function, since it is symmetric, we have

$$\delta_{\text{rel}}(H) = \frac{n_{\text{nz}}(H)}{\frac{1}{2} n_z (n_z + 1)}.$$

As mentioned in Subsection 3.2.4, n_{nz} increases strictly monotonically with the number of shooting nodes. Modern NLP solvers are able to exploit these sparsity structures. By increasing the number of shooting nodes, we increase the dimensionality of the NLP, but due to the specific structure in the objective function and new constraints, the relative densities of the Jacobian of the constraints and the Hessian matrix of the Lagrange function decrease. This is illustrated in Figure 4.8.

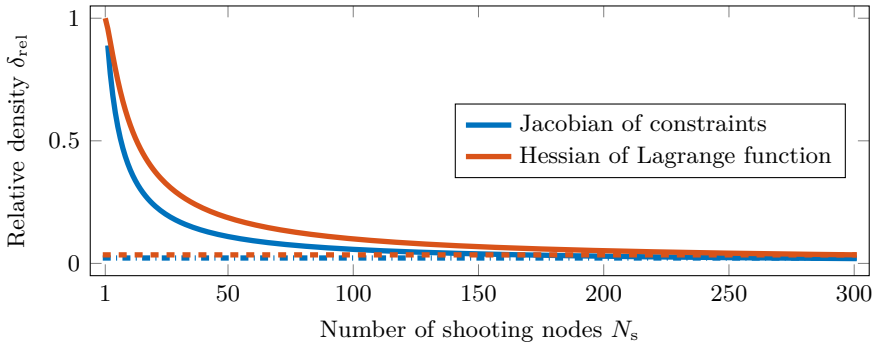


Figure 4.8: Relative densities of NLP matrices. Dashed and dotted lines belong to full discretization, solid lines to multiple shooting for different numbers of shooting nodes.

4.2.4 Real-World Scenario

We consider the 6-link robot *DENSO VS-050* (see Figure 4.5b), which finds applications in many industrial fields. In contrast to the idealized robotic system, this one does not have easily-describable shapes and especially a higher number of links and joints. This makes the parameter identification a challenging task. We focus on parameters of the first two links and therefore assume that the links three to six serve as an extension of link two, such that the respective joints are fixed. This allows us to use the dynamical system from the idealized robot. We are further able to produce real measurements via the robot's sensors. As in the previous investigation, we use the joint angles and joint angular velocities as quantities to be fitted and apply the full discretization method, in which the ODE is solved using the trapezoidal rule.

Figure 4.9 shows the obtained optimal state trajectories, which approximate the measurements in a qualitatively sufficient way. Since we are interested in finding a set of parameters that is applicable not only for one scenario, but in a more general case, we validate the identified parameters by simulating the system for a different experimental measurement, which is shown in Figure 4.10. The identified parameters lead to a model output (using again the trapezoidal rule) that describes the real measurements well.

However, the validation fails for qualitatively different measurements, which might be explained by providing too little information in the measurements. Still, full discretization proves to be applicable even in real-world scenarios.

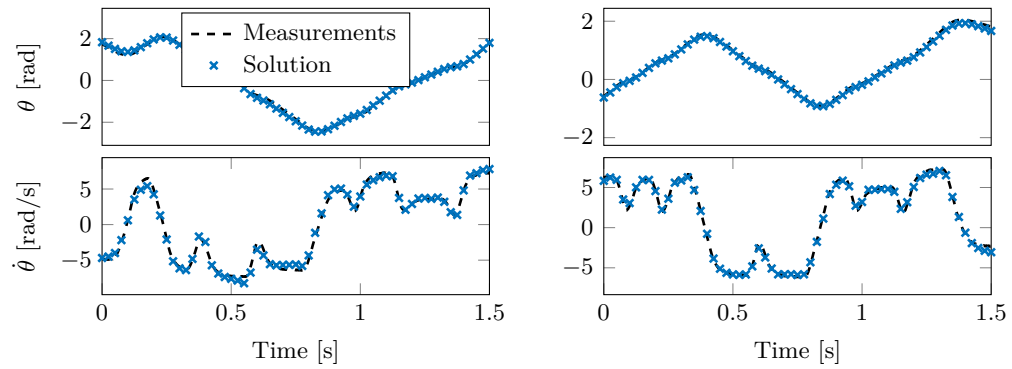


Figure 4.9: Solution trajectories obtained using the full discretization approach for the real-world robotic system.

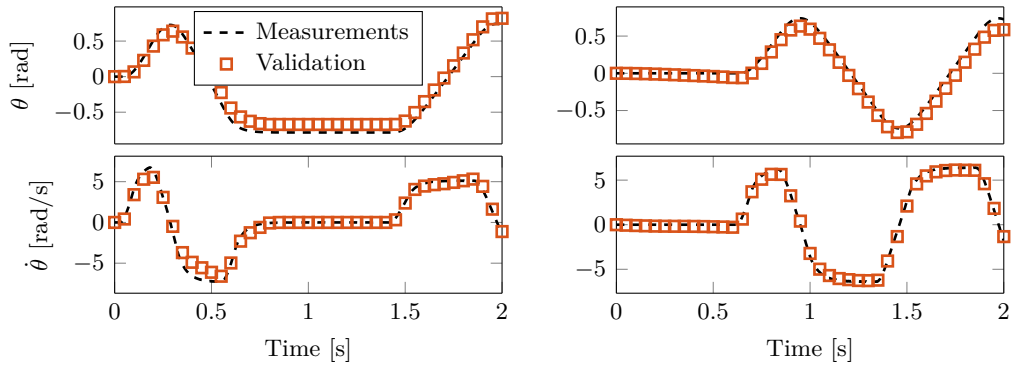


Figure 4.10: Simulation of the robotic system with the identified model parameters using measurements different to those used for parameter identification.

4.3 A Non-Unique Lower-Level Problem

Bilevel optimization problems are often transformed into single-level problems, which are then solved with local algorithms. These algorithms tend to find the closest local solution, depending on a provided initial guess. In bilevel programming, the lower-level problem can be non-unique: multiple local solutions may exist (which is to say that Assumption 3.27 does not hold). Besides, the KKT approach cannot distinguish between minima and maxima, which can be disadvantageous. The reduced approach as well as the decomposed approaches developed in this work can overcome these problems, at least partly. This section illustrates this phenomenon by analyzing an example problem.

4.3.1 Problem Description

We consider an example from [78], which has the following form:

Problem 4.1
Find upper-level variables $x \in \mathbb{R}$ and lower-level variables $y \in \mathbb{R}$ that

$$\begin{aligned} \text{minimize} \quad & (x + 0.6)^2 + y^2 \\ \text{subject to} \quad & x \in [-1, 1] \\ & y \in \arg\min_{v} \quad f(x, v) \\ & \qquad \text{s.t.} \quad v \in [-1, 1] \end{aligned}$$

with

$$f(x, v) \coloneqq v^4 + \frac{4}{30}(1 - x)v^3 + \Big(-0.02x^2 + 0.16x - 0.4\Big)v^2 + \Big(0.004x^3 - 0.036x^2 + 0.08x\Big)v.$$

Both the upper- and the lower-level problem are one-dimensional, have a nonlinear objective function and box constraints. According to [78], the lower-level problem has three KKT points for all $x \in [-1, 1]$:

$$\begin{aligned} y_1 &= -0.5 + 0.1x, \\ y_2 &= 0.4 - 0.1x, \\ y_3 &= 0.1x. \end{aligned}$$

They are visualized in Figure 4.11 in combination with the objective function. For

$$-1 \leq x < -\frac{1}{2},$$

the unique global minimum is y_2; for

$$-\frac{1}{2} < x \leq 1,$$

y_1 is the unique global minimum; for $x = -\frac{1}{2}$, two minima exist: $y = -0.55$ and $y = 0.45$; y_3 describes a maximum. This non-uniqueness of KKT points of the lower-level problem has consequences for all investigated problem formulations. In

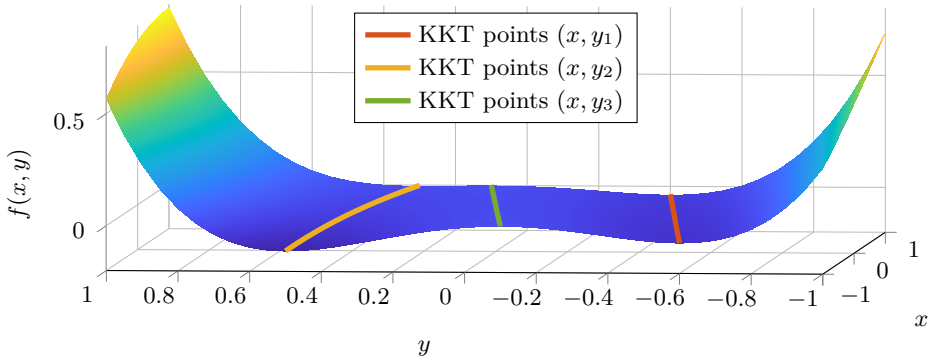


Figure 4.11: Lower-level objective function and KKT points of Problem 4.1.

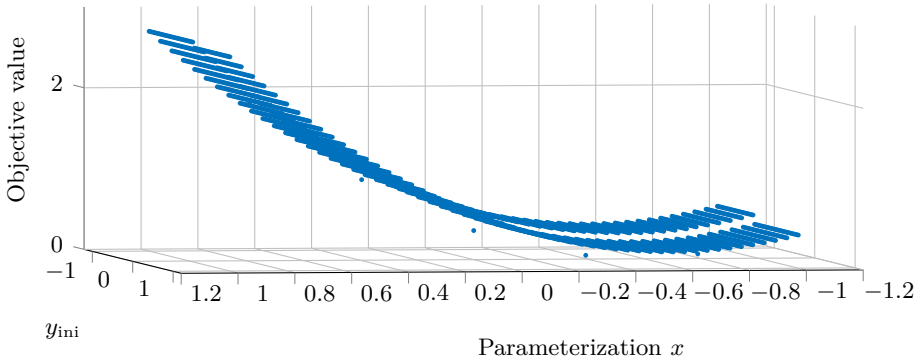


Figure 4.12: Shape of the objective function of the reduced formulation of Problem 4.1. Isolated points represent maximum points of the lower-level problem.

the reduced approach, for instance, the expression ψ mapping x to a KKT point y is not unique. It depends on both the current x and the initial guess chosen for solving the lower-level problem, which solution is finally attended. This can lead to a nonsmooth behavior.

4.3.2 Reduced Objective Function

The objective function of the reduced approach is visualized in Figure 4.12. Although there is only one optimization variable, one has to take the y-dependency into account. The function $\psi(x)$ requires solving the lower-level problem numerically by an algorithm, which is initialized by y_{ini}. For visualization, the region $[-1, 1]^2$ is approximated by a grid and for each point on this grid, the objective function is

evaluated. Figure 4.12 reveals the complex nature of Problem 4.1. Two separated regions indicate two solutions, which we denote as the local and the global one. For some points — those satisfying $y = 0.1x$ — the inner minimization converges to a maximum. As these points correspond to better upper-level objective values, they falsify the shape. Thus, it depends on the initialization of the lower-level problem, to which minimum the algorithm is likely to converge.

4.3.3 Regions of Attraction

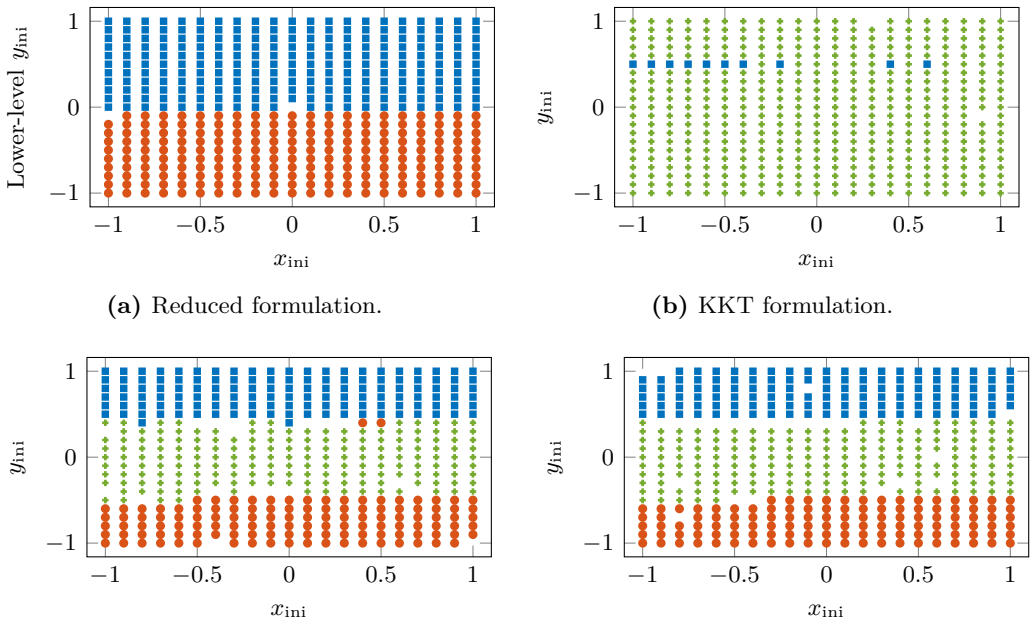
To illustrate the behavior of the approaches introduced in Section 3.3, we again select a grid inside $[-1, 1]^2$ to be used as initializations for the problem formulations. As for the generation of the reduced objective function visualization, we solve problems of type (R-BP) for Problem 4.1 using initializations x_{ini} and lower-level initializations y_{ini} from this grid. Similarly, we solve (KKT-BP) for initializations x_{ini} and y_{ini} from the grid and Lagrange multipliers

$$\lambda_{\text{ini}} = \max\{0.01, -g(x_{\text{ini}}, y_{\text{ini}})\},$$

as in [43].

The results are visualized in Figure 4.13. For the reduced formulation (see Figure 4.13a), it depends heavily on the solver initialization, which minimum is attained. This is not surprising in view of the shape of the reduced objective function. It is noteworthy that the lower-level maximum point is never attained for all initializations. A possible explanation for this behavior is that a sophisticated NLP algorithm for solving the inner problem is used. Thus, the goal of minimization is taken into account. This is in contrast to the KKT formulation (see Figure 4.13b), for which the algorithm converges to the maximum point in the majority of cases. This happens due to the specifics of the formulation: the constraints only demand to find a KKT point, which also includes maximum points. Since the upper-level objective value of the lower-level maximum point is smaller than those of the actual minima, this point is attained frequently in this formulation. In the reduced approach, a minimization process is utilized, and this information is missing in the KKT approach: a clear disadvantage, verified by the obtained results.

As this minimization process is imitated in the decomposed approaches, they show a different picture. Next to the global and local minimum, the maximum point is also attained. Figures 4.13c and 4.13d show the respective results for $N_{\text{it}} = 10$, where the variables are initialized as $y_{i\text{ini}} = y_{\text{ini}}$ and $\lambda_{i\text{ini}} = \max\{0.01, -g(x_{\text{ini}}, y_{i\text{ini}})\}$



(a) Reduced formulation.

(b) KKT formulation.

(c) Decomposed formulation.

(d) Decomposed-KKT formulation.

Figure 4.13: Regions of attraction of local solutions for Problem 4.1 using different problem formulations. Blue dots indicate convergence to the global minimum, red dots convergence to the local one. Green dots represent convergence to the lower-level maximum point. Empty grid points indicate unsuccessful solver terminations.

for $i = 1, \ldots, N_{\text{it}}$. Here, both approaches behave qualitatively similar. Additional experiments indicate that the outcome does not change significantly by varying N_{it}.

This example illustrates that the KKT approach suffers from a missing minimization information, which is naturally given in the reduced approach and, to some extent, given in the decomposed methods.

4.4 Increasing the Region of Attraction

One desired property of the decomposition idea is to make the problem more robust with respect to poor solver initializations. Directly connected to this is an enlarged region of attraction: a desired solution should be found for many initial guesses,

not only for those in the solution's close vicinity. In this section, we illustrate this property in the case of bilevel optimization with a simple example.

4.4.1 Problem Description

We consider the following bilevel problem:

Problem 4.2
Find upper-level variables $x \in \mathbb{R}$ and lower-level variables $y \in \mathbb{R}$ that

$$\begin{aligned} \text{minimize} \quad & 16x^2 + 9y \\ \text{subject to} \quad & -4x + y \leq 0 \\ & -x \leq 0 \\ & y \in \arg\min_{v} \quad (x + v - 20)^4 \\ & \qquad \text{s.t.} \quad 4x + v - 50 \leq 0 \\ & \qquad\qquad -v \leq 0. \end{aligned}$$

According to [102], Problem 4.2 has a local solution at $z^\star_{\text{loc}} \coloneqq (x^\star_{\text{loc}}, y^\star_{\text{loc}}) = (7.2, 12.8)$ with the upper-level objective value $F(x^\star_{\text{loc}}, y^\star_{\text{loc}}) = 2304$ and a global solution at $z^\star_{\text{glo}} \coloneqq \left(x^\star_{\text{glo}}, y^\star_{\text{glo}}\right) = (11.25, 5)$ with the function value $F\left(x^\star_{\text{glo}}, y^\star_{\text{glo}}\right) = 2250$. The lower-level feasible objective function[5] — in comparison to the one from the previous example (see Figure 4.11) — has a unique global minimum for each x.

4.4.2 Reduced Objective Function

To visualize the feasible objective function of the reduced approach, we evaluate it on 501 equidistant values of x on the interval $[0, 20]$. Evaluating this function requires solving the lower-level problem numerically. The underlying optimization is initialized by $y_{\text{ini}} = 4$, but any other (meaningful) value would lead to the same results, which are shown in Figure 4.14. Several observations can be made:

– As mentioned before, two minima exist: a local one at $x^\star_{\text{loc}} = 7.2$ and a global one at $x^\star_{\text{glo}} = 11.25$. Hence, it depends on x_{ini}, which minimum will be attained, since we use local algorithms.

[5] The objective function is evaluated only at points satisfying the constraints.

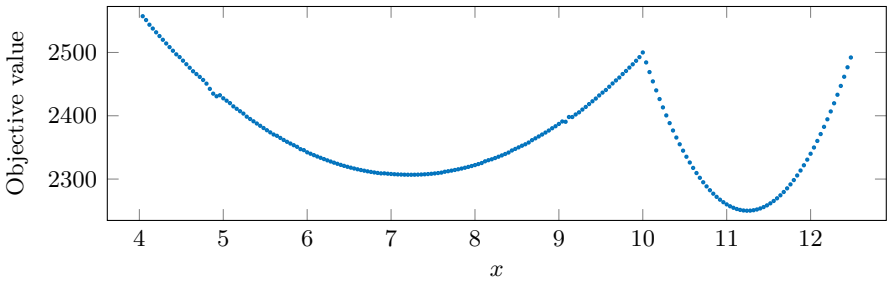


Figure 4.14: Shape of the feasible objective function of the reduced formulation of Problem 4.2.

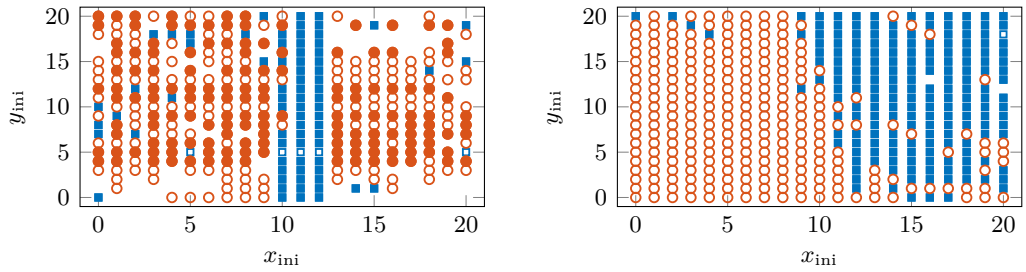
- This function is non-smooth with a kink at $x \approx 10$, which may lead to difficulties for a solver.
- At $x \approx 5$, for example, numerical inaccuracies lead to function values which deviate slightly from the expected ones.

Due to the existence of several solutions, it depends on the algorithm initialization, which one will be attained. Figure 4.14 already indicates that the reduced approach may have difficulties in finding the global solution.

4.4.3 Regions of Attraction

As in previous parts, we define the region of attraction for $z^\star_{\text{loc}}$ and $z^\star_{\text{glo}}$ as the set of initializations $z_{\text{ini}} := (x_{\text{ini}}, y_{\text{ini}})$ within the region $[0, 20]^2$ which lead to the respective solutions when a suitable algorithm is applied. For computational reasons, the region is represented by a grid of 21 equidistant points in both variables. Hence, for each investigated formulation, we solve the corresponding problem 441 times using different initializations and count how often the local or the global solution is found. Here, we treat a computed value as a local or global solution if it does not deviate more than 0.5 % from the respective optimal function value. An unsuccessful solver run is defined as a failure of the solver to find a KKT point (up to the given tolerance). Although the resulting outcome is no longer trustworthy, it is still interesting to observe the algorithmic behavior. Thus, the solver may converge unsuccessfully to a solution. Only small deviations from the reference values indicate that the solver may be close to satisfying the optimality conditions.

Although the reduced approach operates only on the upper-level variables x, the lower-level problem has to be initialized as well. Hence, we solve the problem for all combinations. The results are visualized in Figure 4.15a. Due to the shape of the



(a) Reduced formulation.

(b) KKT formulation.

Figure 4.15: Regions of attraction of local solutions for Problem 4.2 using different problem formulations. Blue dots indicate convergence to the global minimum, red dots convergence to the local one. Filled dots indicate a successful solver termination, unfilled ones the opposite. Empty grid points indicate convergence to a point different from the local or global solution.

objective function (see Figure 4.14), the distribution of solutions is not surprising. For $x_{\text{ini}} \in \{10, 11, 12\}$, the global solution can be found for almost all y_{ini}. For other values of x_{ini}, either the local solution is found or the solver terminates unsuccessfully at a completely different point. Overall, we can observe that the reduced approach is not robust, since it does not converge reliably.

The KKT approach, on the other hand, performs differently. Here, we initialize the Lagrange multipliers as follows:

$$\lambda_{\text{ini}} = \max\{0.01_{n_g}, -g(x_{\text{ini}}, y_{\text{ini}})\},$$

as it is also done in [43] (the max-function is evaluated component-wise). The results are visualized in Figure 4.15b. The global solution is found in approximately 41.3 % of all cases. With this formulation, the algorithm also converges to the local solution, but it does not converge successfully. This can be interpreted as an advantage, since the global minimum is the one to be preferred.

For the decomposed formulations, we are interested in the influence of N_{it}, the number of provided lower-level iteration steps. We initialize all variables y_i $(i = 2, \ldots, N_{\text{it}})$ as y_{ini} and λ_i identical to those in the KKT approach for $i = 2, \ldots, N_{\text{it}}$. As before, we solve the problem using 441 combinations of initial guesses, which is now done for $N_{\text{it}} = 2, \ldots, 50$. As we consider two variants of the decomposed approach, a total of 43218 problems are solved. Again, we count how often the global or the local solution is found.

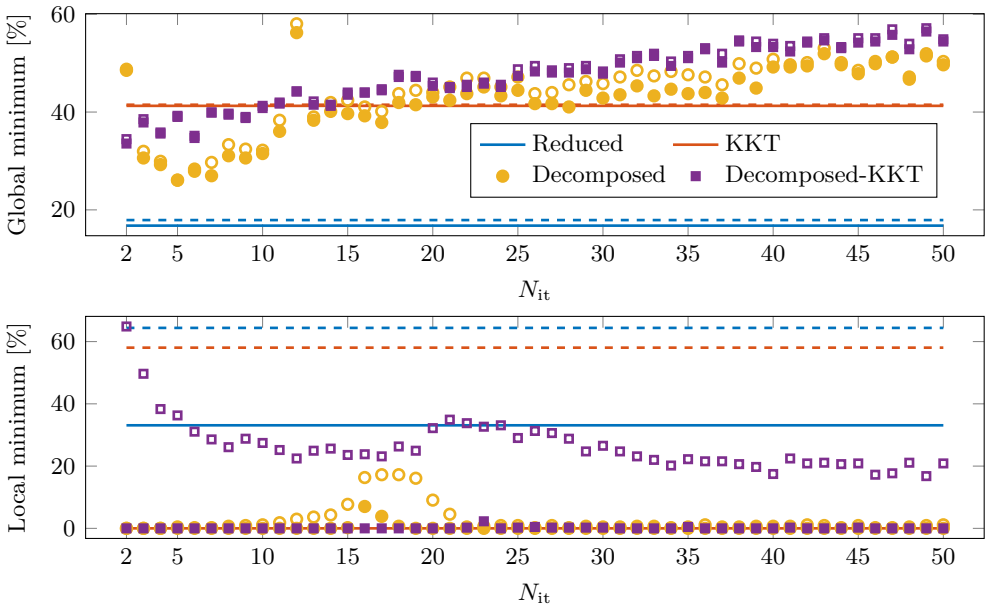


Figure 4.16: Fractions of algorithm runs that converge to the global (upper) or local (lower) solution for different problem formulations. Filled dots and solid lines represent successful solver runs.

The results can be seen in Figure 4.16. Analogous to the reduced and KKT approach, we distinguish between successful and unsuccessful solver terminations. The upper graphic shows, how often the global solution is computed for different problem formulations. With the reduced formulation, the global solution can only be found in approximately 16.8 % of all tries, while the KKT approach finds it in 41.3 % of the cases. The decomposed approach (the one without explicitly having KKT conditions) already performs well for $N_{\text{it}} = 2$: the global solution can be found in 48.5 % of the cases. In view of the results for the remaining values of N_{it}, however, this needs to be regarded as an outlier. The same holds for $N_{\text{it}} = 12$, where an extraordinary value of 56.2 % is attained. Except for these two outliers, a trend is observable: for increasing values of N_{it}, the global solution can be found more often. This approach outperforms the KKT formulation for $N_{\text{it}} \geq 18$ (with only a single exception). In most of the cases, the solver terminates successfully when the global solution is attained. Only for $N_{\text{it}} \in [26, 39]$, the solver has more difficulties.

For this example, the combination of the decomposed and the KKT approach appears to be even more beneficial. For each N_{it} (except for outliers), it performs better than

the corresponding decomposed variant. Further, it outperforms the KKT approach already for values of $N_{\mathrm{it}} \geq 11$. The best solution is attained for $N_{\mathrm{it}} = 49$ with 56.5 % of all cases converging to the global solution. Moreover, this approach shows fewer differences between successful and unsuccessful solver terminations, which indicates higher robustness. In total, we observe that for increasing values of N_{it}, the global solution is found more often, leading to an increased region of attraction.

The lower graphic shows the same evaluations for the local solution. Here, the reduced approach converges successfully in 33.1 % of the cases and without a successful termination code in 64.4 %. The KKT approach converges to the local solution in 58 %, but fails to terminate successfully. The decomposed approach shows an interesting behavior: it converges to the local solution in some cases only for $N_{\mathrm{it}} \geq 8$ and $N_{\mathrm{it}} \leq 22$, but the solver fails to terminate successfully. Only for some outliers around $N_{\mathrm{it}} = 17$, the local solution can be found successfully in some rare cases. Combining the decomposed and the KKT approach yields a completely different behavior: Up to $N_{\mathrm{it}} = 17$, the number of times the local solution is found decreases, while it rises again up to $N_{\mathrm{it}} = 21$. For $N_{\mathrm{it}} \geq 22$, however, the tendency is again decreasing. In none of these cases, the solver was able to fulfill the KKT conditions (except for only one outlier at $N_{\mathrm{it}} = 23$). An interpretation of this behavior is not trivial. We can conclude that only the reduced approach is able to reliably compute local solutions, all other methods do not find them.

4.4.4 Computational Characteristics

We have seen that decomposition can help to increase the region of attraction. However, it comes at the price of higher problem dimensions, possibly leading to increased computation times. To obtain an overview of computational characteristics for this study, we evaluate statistical quantities that describe the computation time and the number of major iterations that the underlying algorithm has to perform to converge successfully. For a consistent comparison, we restrict the samples to correspond to those runs that converge to the global minimum. Similar to evaluations for the pendulum (see Subsections 4.1.3 and 4.1.4), we exclude those runs that started at a point from which the other investigated method does not converge, respectively. Thus, the intersection of initial guesses that yield the global minimum with a successful termination criterion is used. Since the reduced approach only converges in rare cases, taking it into account would reduce the sample size immensely. Hence, we exclude this method for the following investigations. Still, the experiments show

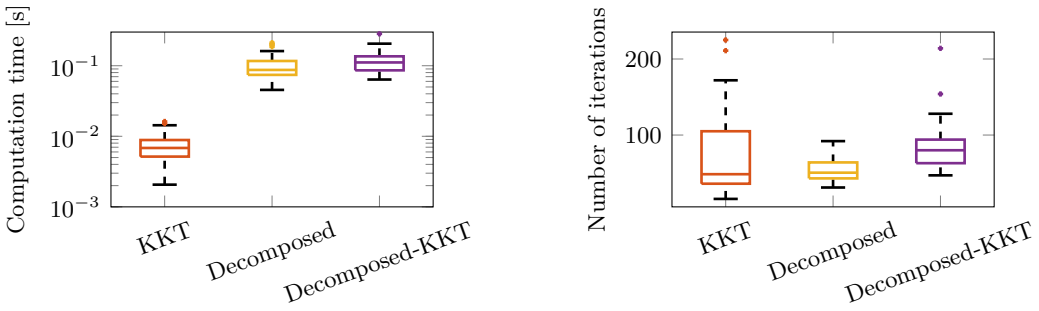


Figure 4.17: Computational characteristics for different formulations of Problem 4.2. Only those runs are considered for which all formulations converge to the global solution, which amounts to a sample size of 50. The decomposed variants show results for $N_{\mathrm{it}} = 18$.

that the reduced approach converges quickly with only a few iterations. This is not surprising, since it converges only for initial guesses close to the solution.

The computational characteristics of the other methods are visualized in Figure 4.17. With respect to computation time (on the left), the KKT approach converges quickly, the median time is 6.8325×10^{-3} s, which is significantly lower than the decomposed approaches. However, one has to take the dimensionality into account, as the results shown correspond to a choice of $N_{\mathrm{it}} = 18$. Under this aspect, the decomposed approaches still converge sufficiently fast from an absolute perspective. The graphic on the right is more expressive: The median numbers of iterations for the KKT approach and the decomposed one are quite similar (48.5 and 50.5), while the combination of those has the median iteration number 80. The IQR, instead, puts the decomposed variants in the foreground. While the KKT approach has an IQR of 69 iterations, the IQR of the decomposed approach is only 21. Here, the combination of the decomposed and the KKT approach exhibits a slightly larger IQR of 31. Hence, the algorithmic behavior of the decomposed methods is much more consistent and reliable.

4.5 Bilevel Decomposition Applied to a Problem Library

The aim of this section is to compare the methods introduced in Section 3.3 with respect to several criteria. In particular, we want to examine whether the novel ideas for decomposition can compete with established methods, such as the KKT approach. In the previous sections, the decomposed approaches have been applied

to several individual examples to demonstrate their specific characteristics. Here, they are applied to a larger set of problems representing different challenges. Thus, the applicability of the methods as well as their ability to reliably recover nominal solutions is of interest.

4.5.1 Comparison Setup

The methods developed in Section 3.3 are tested on several examples originating from the *Bilevel Optimization LIBrary of Test Problems* (`BOLib`) [130]. It consists of 173 bilevel problems, divided into 11 so-called simple, 24 linear, and 138 nonlinear ones. The library is given as `MATLAB` code[6]. It defines a problem by providing the functions F, G, f, and g as well as their first and second derivatives. In particular, the Jacobian matrices $\nabla_y f$ and $\nabla_y g$ as well as the Hessians $\nabla^2_{yy} f$ and $\nabla^2_{yy} g$ are given, which makes an exact computation of the Hessian matrix $\nabla^2_{yy} \ell$ possible. We focus on the 138 nonlinear bilevel problems, from which we exclude 14 problems that do not fit into our framework, two problems with non-differentiable functions, and six problems that do not have a known optimal solution. All in all, we consider 116 bilevel problems for our evaluations, which do not exceed the dimensions $n_x = 10$, $n_y = 10$, $n_G = 13$, and $n_g = 20$. The problem `HenrionSurowiec2011` is parametrized in $c = 1$ and `IshizukaAiyoshi1992a` in $M = 1.5$.

Extensive numerical tests are not yet established in the development of bilevel programming methods. Nevertheless, `BOLib` is already used in several works, for example in the area of stochastic linear bilevel programming [82] or, more often, in the area of lower-level value function reformulations [34, 38, 111]. This approach is also investigated by Fliege et al. [43], who penalize the value function in the objective of the upper-level problem. The resulting optimality conditions are then solved by a Gauss–Newton-type method. Although their approach differs from the one studied here by the type of reformulation (value function vs KKT-type), we can adopt some of their algorithmic choices and visualization options.

The problems are reformulated using the methods presented in Section 3.3. To solve them numerically, we use the NLP solver `WORHP`, for which the library is translated from `MATLAB` to C++. Within `WORHP`, we choose the SQP method. While the exact derivatives provided in the library are used to formulate the constraints for the problem formulations, the derivatives required by the SQP method are computed by finite differences. Although an exact computation would be possible, we use

[6] The source code is published at `https://biopt.github.io/bolib/`.

this approach to remain consistent with other investigations in this work. For all computations, a tolerance of 10^{-5} is used (optimality, feasibility, complementarity). The computation time is limited by 30 s.

For benchmarking purposes, `BOLib` provides the true or best known solutions x_{ref} and y_{ref} as well as the corresponding objective function values $F_{\text{ref}} \coloneqq F(x_{\text{ref}}, y_{\text{ref}})$ and $f_{\text{ref}} \coloneqq f(x_{\text{ref}}, y_{\text{ref}})$. In the following, we denote a computed solution as $x^\star$ and $y^\star$ and their corresponding upper- and lower-level objective function values as $F^\star \coloneqq F(x^\star, y^\star)$ and $f^\star \coloneqq f(x^\star, y^\star)$. To measure the quality of a computed solution, we introduce the *relative error*

$$r_X \coloneqq \frac{|X^\star - X_{\text{ref}}|}{1 + |X_{\text{ref}}|}$$

with $X \in \{F, f\}$ in accordance with [43]. We consider a computed solution to match its reference solution if the corresponding relative errors r_F and r_f are less than 5 %.

To compare the methods, several decisions have to be made. On the formulation level, there exist multiple possibilities to incorporate the complementarity constraints. They can either turn the problem into an MPCC or one can make use of the (smoothed) FB version. Another aspect lies in the nature of the decomposed approaches. Due to the parameterization in N_{it}, solving sequences of these problems may be a strategy superior to selecting a specific value of N_{it}. On the NLP level, an initial guess has to be provided.

To evaluate the results, several criteria have to be considered. The strongest one is the ability to recover the nominal solution, which is given for each problem instance of interest. Finding the nominal solution here means computing a solution with a relative error less than a given tolerance. Although the upper-level objective function value is the most important quantity here, the lower-level objective function value also needs to be close to its nominal solution. Otherwise, the constraint that the lower-level solution must be found may be violated. Another point of interest is the solver's ability to produce a positive termination message, preferably one that says that a local solution has been found. This means an approximate fulfillment of KKT conditions for that particular problem formulation. However, a solver may be able to approximate reference values up to the given relative error tolerance, but may fail to terminate successfully. This will also be considered in the following investigations. If the lower-level KKT conditions are not part of the respective problem formulation (for example, in the decomposed approach), they are evaluated after the optimization has terminated to make statements about this criterion as well. All in all, we come

up with several criteria of interest that can be analyzed on their own or even in combination with each other. Their purpose is to help to describe and analyze the numerical results.

$\mathbf{KKT_F}$ The solver terminates successfully, an approximate KKT point of the respective formulation is found. This criterion describes the ability of the solver to converge to a solution with a given formulation. Note that for the decomposed approach, this does not necessarily mean that a lower-level KKT point is found.

$\mathbf{KKT_{LL}}$ An approximate lower-level KKT point is found. For the KKT and the decomposed-KKT approach, this criterion is fulfilled whenever a feasible point is found. For the reduced approach, this criterion cannot be evaluated since it only operates on the upper-level variables. Thus, we assume that it is always fulfilled, unless the inner optimization fails. For the decomposed approach, an additional evaluation of KKT conditions in x and $y_{N_{\text{it}}}$ has to be done.

$\mathbf{KKT_{F,LL}}$ The solver terminates successfully for a given formulation and the corresponding solution is a lower-level KKT point.

$\mathbf{REF_{UL}}$ The upper-level reference objective function value F_{ref} is approximately found. For a tolerance ϵ_F, it holds that $r_F < \epsilon_F$.

$\mathbf{REF_{LL}}$ The lower-level reference objective function value f_{ref} is approximately found. For a tolerance ϵ_f, it holds that $r_f < \epsilon_f$.

$\mathbf{REF_{UL,LL}}$ Both reference objective function values are approximately found.

While the initializations of x and y or respectively $y_1, \ldots, y_{N_{\text{it}}}$ vary within the following sections, the Lagrange multipliers are always initialized in the same way:

$$\lambda_{\text{ini}} = \min\{\delta, -g(x_{\text{ini}}, y_{\text{ini}})\}$$

or respectively

$$\lambda_{k\text{ini}} = \min\{\delta, -g(x_{\text{ini}}, y_{k\text{ini}})\}$$

for $k = 1, \ldots, N_{\text{it}}$. Here, we follow the approach used in [43] and choose $\delta = 0.01$.

4.5.2 Number of Provided SQP Steps

The crucial part of formulating a bilevel problem as a D-BP is the choice of N_{it}, the provided lower-level iteration steps. Too few imitated iterations may not be able to produce a lower-level KKT point (compare Example 3.40), while too large values of N_{it} highly increase the problem dimensions, and — in spite of its inherent sparsity — computing a solution may be connected to lots of efforts. It is not possible to provide a general recommendation on how to choose N_{it}. For this reason, an explorative study on this choice is conducted. In particular, each problem instance (in total 116, see previous part) is solved using the decomposed approach for a fixed value of N_{it}. Here, N_{it} ranges from its minimum value of 2 up to a value of 50 (further experiments indicate that choosing even larger values is not advantageous). We choose initializations that are provided in the library.

For each solver run, we evaluate the criteria previously defined in Subsection 4.5.1. They are visualized in Figure 4.18 as bar charts. At the top, criteria for the formulation itself are displayed. Several aspects can be observed. For $N_{\mathrm{it}} = 2$, in almost 88 % of all problem instances the solver terminates successfully, a KKT point is found. However, this value needs to be interpreted carefully, since it does not necessarily mean that the original bilevel problem is solved. In particular, only in 47.4 % of the cases, the reference upper-level objective function value can be approximated. By increasing the value of N_{it}, one can observe a slight decrease in the successful solver termination rates, which is also not surprising. Larger values of N_{it} correspond to more complex problem formulations and, possibly, the occurrence of fewer local solutions. However, this decrease is not significant, which is contrary to the increase one can observe for the reference KKT solutions, at least for $N_{\mathrm{it}} \leq 11$. A rising trend is observable, which finds its peak at $N_{\mathrm{it}} = 11$: in 59.5 % of the cases, the reference solution is found. This behavior matches the expected one, since the solver now has more flexibility to imitate lower-level convergent algorithm runs. For $N_{\mathrm{it}} > 11$ — in contrast — the picture is not as clear as before and an interpretation is non-trivial. It is not true that an increase in N_{it} leads to finding the reference solution more often. There seems to be a breaking point at $N_{\mathrm{it}} = 11$, though. For only a small amount of problems, the reference solution is found, but it does not fulfill the corresponding KKT conditions. This can be explained by different tolerances that are used for the relative error and KKT conditions.

With regard to the lower level (the middle graphic), a similar behavior can be observed with one exception: the number of times the KKT conditions are fulfilled rises for increasing N_{it} (up to $N_{\mathrm{it}} = 8$) instead of dropping (as it is the case for the

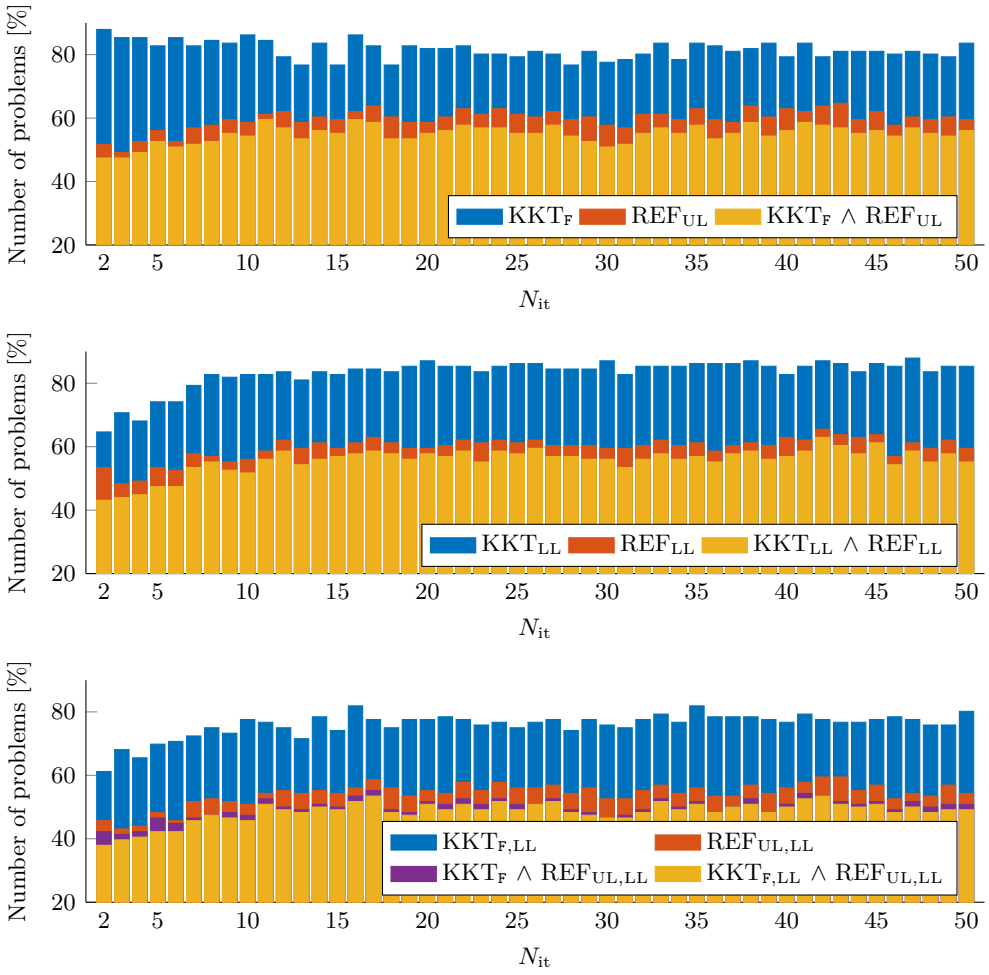


Figure 4.18: The influence of N_{it} on the number of problems solved with respect to several criteria. For each N_{it}, `BOLib` is solved with the decomposed approach. Top: criteria with regard to the problem formulation and the upper level. Middle: criteria with regard to the lower level. Bottom: mixed criteria.

upper level). This also matches the expected behavior, since many problems might not be solved with only a few lower-level iteration steps. Other than that, these results are similar to the ones for the upper level.

In the lower graphic, the most important criteria are visualized. One can again observe a rising trend for all criteria up to $N_{\mathrm{it}} \approx 11$, which then roughly stabilizes. Similar to the results above, one can observe that a lower-level reference point does not represent a KKT point in only a few cases. The best result can be found for $N_{\mathrm{it}} = 17$ or $N_{\mathrm{it}} = 42$: in 53.4 % of all cases, the computed solution is a KKT point in both levels and approximates the corresponding reference values as well.

This initial investigation already allows drawing some conclusions:

- For this kind of initialization, N_{it} should be chosen larger than 11.
- It is inevitable to check whether a computed solution really represents a lower-level KKT point.
- For the majority of problems, the reference solution can be recovered. However, there are also many problems that could not be solved.

Thus, it may not be the best idea to choose the same fixed N_{it} for all problems. Instead, choosing N_{it} individually for each problem may be a promising alternative.

4.5.3 Solution Strategies

In the following, the initialization is changed: the initial guess from the library is replaced by zeros. Hence, we put ourselves in the position that we do not have any information about the respective problem at hand. It is expected that this initialization is challenging. Further, we introduce a solution strategy for the decomposed approaches. Different to the investigations before, we no longer fix a specific N_{it}. Instead, a given problem is initially solved for $N_{\mathrm{it}} = 2$. After, it is checked whether the result fulfills the criterion $\mathrm{KKT}_{\mathtt{F},\mathrm{LL}}$. We choose this criterion because a reference solution is typically not available outside this analysis. Hence, until $\mathrm{KKT}_{\mathtt{F},\mathrm{LL}}$ is fulfilled, N_{it} is increased by one with a maximum value of 50. This approach allows applying an adaptive strategy: while $\mathrm{KKT}_{\mathtt{F},\mathrm{LL}}$ is not fulfilled, the variables can be initialized by those from the previous run. Since the number of variables increases, the updated y_N and λ_N can be chosen identical to $y_{N_{\mathrm{it}}-1}$ and $\lambda_{N_{\mathrm{it}}-1}$, respectively. Another obvious strategy is to only increase N_{it} and leave the variables as they are. Both approaches have their advantages and it is not directly clear which one is suited better. For comparison, both approaches are tested for the decomposed formulations, the results can be found in Table 4.4. The non-adaptive

	Decomposed		Decomposed-KKT	
	Adaptive	Non-adaptive	Adaptive	Non-adaptive
$\mathbf{KKT_F}$	100	106	103	109
$\mathbf{KKT_{LL}}$	102	102	102	109
$\mathbf{KKT_{F,LL} \wedge REF_{UL,LL}}$	62	62	59	64

Table 4.4: Number of problems solved by the decomposed and decomposed-KKT approach using different solution strategies.

strategy beats the adaptive one with respect to the criteria $\mathrm{KKT_F}$, $\mathrm{KKT_{LL}}$, and $\mathrm{KKT_{F,LL}} \wedge \mathrm{REF_{UL,LL}}$, both in the decomposed and the decomposed-KKT approach. Although the difference between the strategies is not significant, we consider the non-adaptive one exclusively in the following.

To compare all methods against each other, we focus on the relative error in the upper level, which we compute after each solver run for each formulation. Hence, four formulations compete against each other: (R-BP), (KKT-BP), (D-BP), and (D-KKT-BP) using the non-adaptive strategy. The results are visualized in Figure 4.19. The relative errors are sorted in an increasing order and presented against the number of problems exhibiting this error level or less.

The upper graphic shows the outcome of all solver runs, independent of any evaluation criterion. One can observe that all formulations produce similar outcomes at the error tolerance: for 69 and 70 problems, the reference upper-level value can be recovered. It is surprising that the reduced approach here performs similarly to the other approaches. A possible explanation may lie in the trivial nature of the problems. With regard to the accuracy, however, the reduced approach produces worse error values than the other approaches.

If we instead only consider those runs for which the solver constitutes to have found an optimal solution (the middle graphic), the picture changes. The reduced approach can only solve 36 problems, while the other approaches only lose some problem instances. Here, an initial tendency can be observed: the decomposed approaches perform better than the KKT approach, more problems can be solved up to the allowed error tolerance. Using this criterion, the reduced approach only solves 20 problems with a relative error of approximately 10^{-6}, while the decomposed approach solves 44 problems with an accuracy of 10^{-7}.

A similar trend can be observed in the lower graphic, in which only those runs are considered that lead to lower-level reference KKT points. Here, the combination

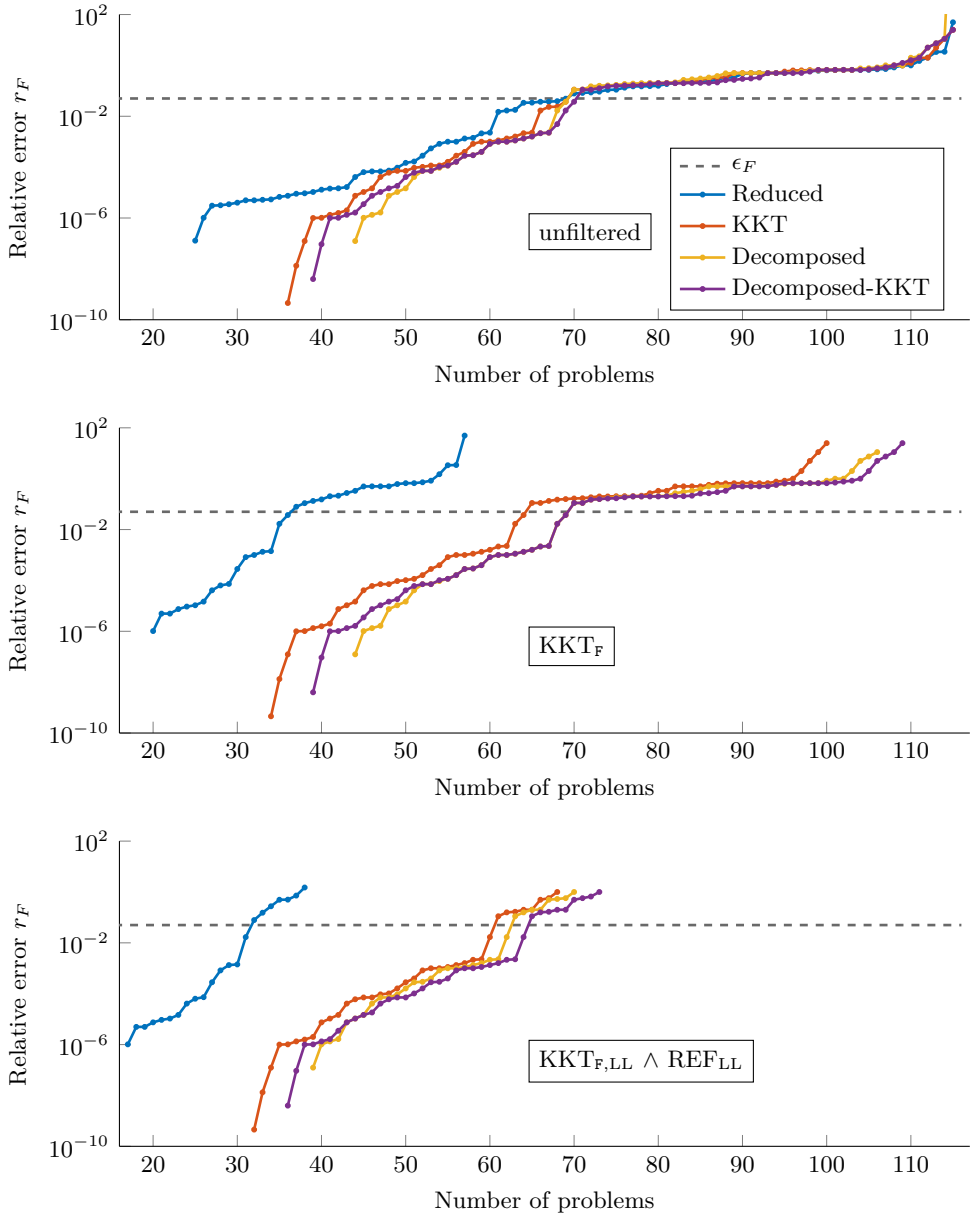


Figure 4.19: Relative error performance comparison using the non-adaptive strategy. The results are filtered according to the criterion in the box.

of the decomposed and the KKT approach performs slightly better than the other ones with respect to quality and quantity. The reduced approach can, in total, only solve 31 problems below the desired tolerance.

The presence of complementarity constraints in the KKT and decomposed approaches causes difficulties for numerical solvers. An often used alternative is FB smoothing. This is used within the following part with a different solver initialization.

4.5.4 Fischer-Burmeister Smoothing

Similar to the previous paragraph, we choose a trivial initialization for the NLPs, namely ones instead of zeros. Again, the non-adaptive strategy is chosen for the decomposed approaches. However, we modify the specific formulation: instead of using complementarity constraints of the form

$$\begin{aligned} g(x,y) &\leq 0_{n_g}, \\ \lambda^\mathsf{T} g(x,y) &= 0, \\ \lambda &\geq 0_{n_g}, \end{aligned}$$

we use the FB approach (compare Subsection 3.3.4). We extend this approach by using a *smoothing* version of it (as being done in [43]), which makes use of a perturbation parameter τ. Within this setting, the smoothed FB function Φ^τ is defined as

$$\Phi^\tau(\lambda, x, y) \coloneqq -\lambda_j + g_j(x,y) + \sqrt{\lambda_j^2 + g_j(x,y)^2 + 2\tau}.$$

For $\tau > 0$, it holds that $\Phi^\tau(\lambda, x, y) = 0$ if and only if $\lambda_j > 0$, $-g_j(x,y) > 0$, and $-\lambda_j g_j(x,y) = \tau$. Instead of solving a single problem using the non-smoothed FB function, we solve a series of problems with $\tau \to 0$. In particular, the procedure in Algorithm C is used. For the decomposed approaches and the strategy chosen therein, the number of problems to be solved increases drastically: for each N_{it}, the procedure described above is applied. The advantages of the FB formulation are presented in Table 4.5, in which it can be seen that in most cases this formulation is superior to the one as an MPCC. These results justify our choice to focus on FB smoothing and to neglect the MPCC formulations.

Again, as in the previous study, we consider the upper-level relative error in combination with various criteria. The results are visualized in Figure 4.20. The top graphic shows all solver runs, independent of any evaluation criterion. For 70 problems, the

Algorithm C Fischer-Burmeister smoothing

1: Given: a problem formulation P ▷ The KKT approach, for example
2: Set $k = 1$, initialize $\tau_k = 0.4$, choose $\beta = 10^{-6}$
3: **while** $\tau_k \geq \beta$ **do**
4: Solve problem P, obtain solution $z^\star$
5: Update $\tau_{k+1} = (\tau_k)^{k+1}$
6: Update $k \leftarrow k + 1$
7: Initialize P with the previously found solution $z^\star$
8: **end while**

	KKT		Decomposed		Decomposed-KKT	
	MPCC	FBs	MPCC	FBs	MPCC	FBs
$\mathbf{KKT_F}$	104	109	110	110	110	112
$\mathbf{KKT_{LL}}$	113	114	109	113	112	113
$\mathbf{KKT_{F,LL}} \wedge \mathbf{REF_{UL,LL}}$	61	69	65	63	65	70

Table 4.5: Number of problems solved by the KKT, the decomposed, and decomposed-KKT approach using different complementarity constraint formulations ("Fischer-Burmeister smoothing" is abbreviated by "FBs").

reduced approach is able to recover the reference value of the upper-level objective function. The decomposed and decomposed-KKT approaches follow, while the KKT approach is able to recover the reference values in 82 out of 116 cases. This is also the case for the accuracy, with one exception. When it comes to high precision (that is, low error values), the reduced approach outperforms the other methods. It solves 31 problems up to an accuracy of 4.4×10^{-9}, 10 problems more than the KKT approach for this error level.

However, if we consider only those runs with a positive solver termination (middle graphic), the picture changes significantly. The number of successes of the reduced approach shrinks to 43, while the other methods are more robust. The KKT and the decomposed-KKT approaches both find the reference value in 78 cases. This ranking is also reflected in the accuracy: the decomposed-KKT approach performs better than its decomposed counterpart.

If we go one step further and restrict the results to lower-level reference solutions only, the decomposed-KKT approach performs best with 70 problems solved to their reference value. Using this strict criterion, the reduced approach solves only 37 problems to the desired tolerance. It also has significantly worse error values than the other approaches.

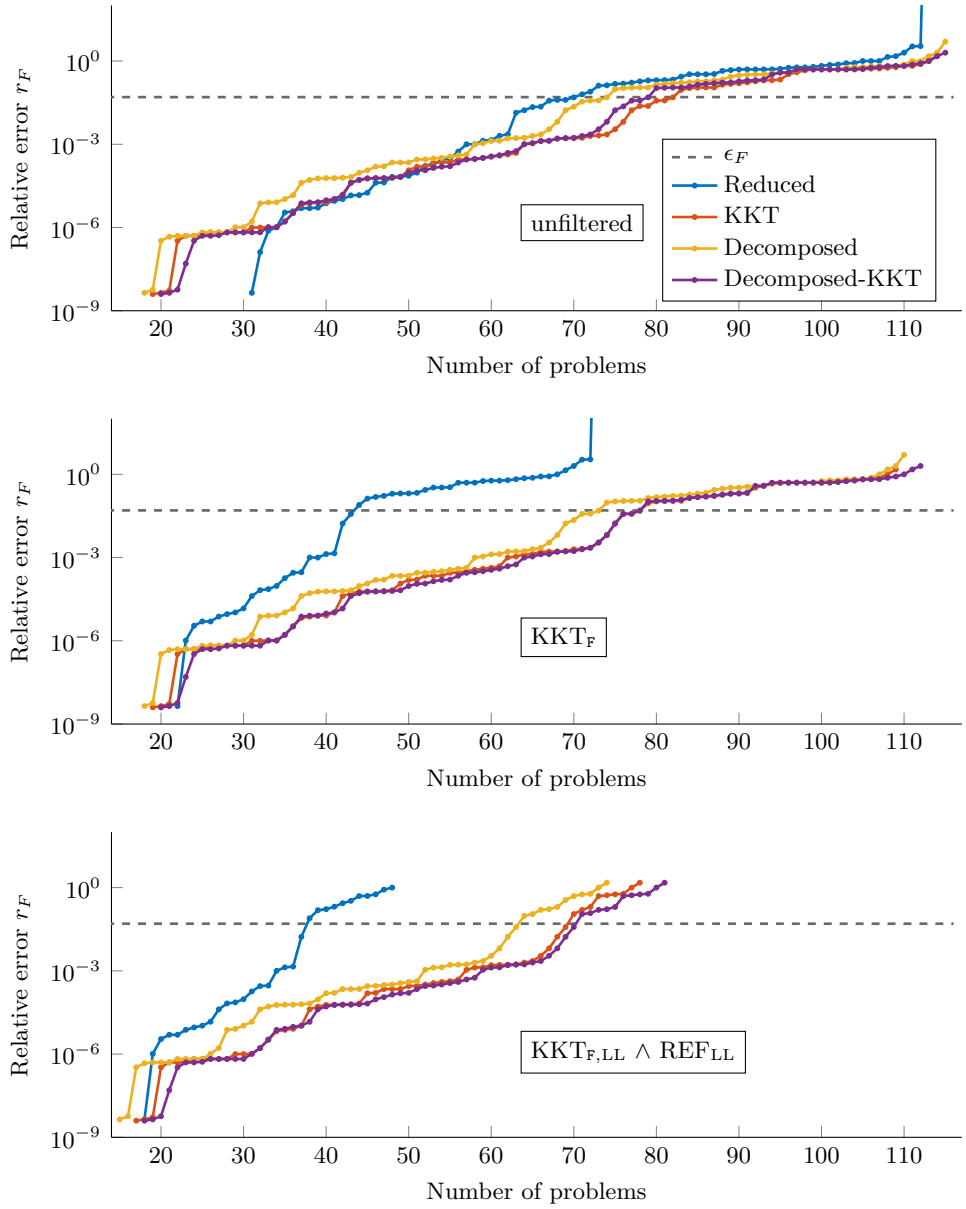


Figure 4.20: Relative error performance comparison using FB smoothing. The results are filtered according to the criterion in the box.

The results demonstrate that the newly developed decomposed approaches can compete with the established KKT approach and are able to outperform it under strict evaluation criteria.

Similar studies have been carried out by Fliege et al. in [43], as mentioned at the beginning of this section. They also consider `BOLib` as a benchmark library to test their methods, which are based on a lower-level value function reformulation. They achieve better results than those presented here, but under less restrictive conditions, especially with respect to the relative error criterion.

4.5.5 Computational Characteristics

In addition to the solution quality in terms of the relative error, other criteria must not be neglected. We are also interested in the effort required to obtain these results. Common quantities of interest are the computation time or the number of iterations required, which can be easily compared for a single problem instance. For a whole set of problems, instead, we use so-called *performance profiles*.

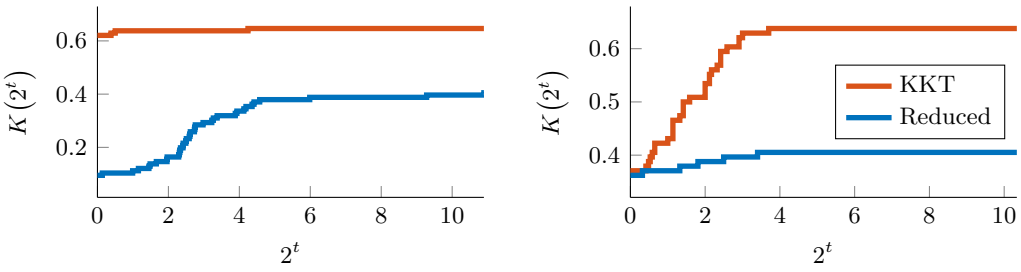
Remark 4.3 A common way to visually compare the performance of different solvers on a set of problems is to use performance profiles. For n_s solvers and n_p problems, the comparison is done with respect to a measurable quantity $q_{p,s} \neq 0$ for $p = 1, \dots, n_p$ and $s = 1, \dots, n_s$. The computation time, the number of iterations, or the number of function evaluations are examples of such quantities. The performance ratio is then defined as

$$r_{p,s} \coloneqq \frac{q_{p,s}}{\min\{q_{p,s} \colon s \in \{1, \dots, n_s\}\}}$$

and describes how a solver s performed in comparison to the solver with the best performance. If problem p cannot be solved by solver s, $r_{p,s}$ is ∞. The performance profile is defined via the cumulative distribution function

$$K_s(T) \coloneqq \frac{1}{n_p} |p \in \{1, \dots, n_p\} \colon r_{p,s} \leq T|.$$

When comparing two solvers, larger values of $K_s(T)$ indicate a better performance. The *efficiency* of a solver s is given by $K_s(1)$ and provides a criterion for comparison with the other solver. Its counterpart is the *robustness*, which is given by $\lim_{T \to \infty} K_s(T)$ and describes the percentage of problems that can be solved by this solver. For a thorough introduction, we refer to [32].



(a) Computation time. **(b)** Number of iterations.

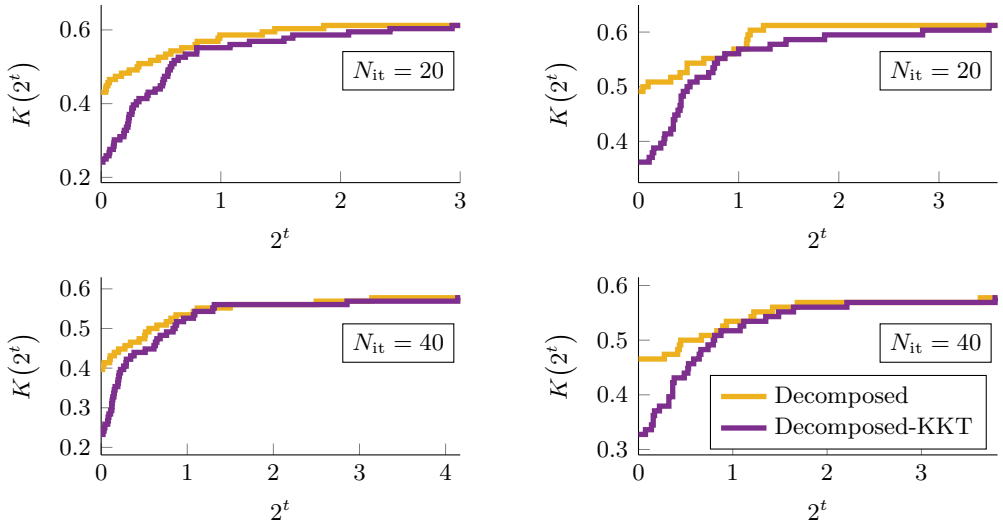
Figure 4.21: Performance profiles with reduced and KKT approaches in MPCC formulation. Success criterion is $\mathrm{KKT}_{\mathrm{F}} \wedge \mathrm{REF}_{\mathrm{UL}}$ with $\epsilon_F = 0.2$.

Independent of problem dimensions, the decomposed approaches in combination with the chosen strategy and the FB smoothing procedure require lots of effort due to the large number of problems that have to be solved. This is why we restrict this comparison to problems in MPCC formulation. With regard to the decomposed approaches, runs for specific values of N_{it} are compared instead of using one of the solution strategies from above.

The previously assumed criteria for success are accurate, but also restrictive. For this investigation, such a restriction is no longer necessary and we require a successful run to fulfill $\mathrm{KKT}_{\mathrm{F}} \wedge \mathrm{REF}_{\mathrm{UL}}$ only. The error tolerance is relaxed to $\epsilon_F = 0.2$. For the decomposed approach, an additional check for lower-level optimality conditions is made.

At first, we compare the reduced formulation against the KKT approach in Figure 4.21. With respect to the numbers of iterations (Figure 4.21b), each method outperforms the other one similarly often (36 % and 37 % at $T = 1$), but the KKT approach is more robust with 64 % solved problems against 41 %. With respect to computation time (Figure 4.21a), the KKT approach is superior. Contrary to the number of iterations, the computation time efficiency is significantly larger than the one from the reduced approach (62 % and 9 %). This is not surprising, since each function evaluation requires the numerical solution of another NLP.

In a second step, the decomposed approaches are compared against each other in Figure 4.22. The library is solved with both formulations exemplarily for $N_{\mathrm{it}} = 20$ and $N_{\mathrm{it}} = 40$. With respect to both the computation time and the required number of iterations, the decomposed approach is more efficient than the decomposed-KKT one. This behavior is expected for the computation time (Figure 4.22a), since the latter approach exhibits some more constraints. Still, it is interesting to see



(a) Computation time. **(b)** Number of iterations.

Figure 4.22: Performance profiles with decomposed approaches in MPCC formulation. Success criterion is $\text{KKT}_{\text{F}} \wedge \text{REF}_{\text{UL}}$ with $\epsilon_F = 0.2$.

that these additional constraints also have an impact on the number of iterations (Figure 4.22b). The decomposed approach performs significantly better here. In the long run, both formulations are able to solve a similar amount of problem instances. One should note that these results have to be handled with care. Although this behavior can be observed for many values of N_{it}, there exist cases with a contrary outcome.

4.6 A Combined Homotopy-Optimization Approach

In this section, an idea for enhancing the reduced formulation within the field of parameter identification is investigated. A particular advantage of the multiple shooting or the full discretization approach is the decoupling of the discretized states from the model parameters. The trajectories are allowed to remain near the measurements, independent of the current iterate of the parameter vector. In fact, it is not possible to transfer this property to the reduced problem formulation. Herein, the state trajectory is implicitly given by the current parameter value, which is one reason why undesired solutions occur during the course of optimization. In this

section, a modification of the underlying dynamical system is presented which allows the reduced formulation to have a similar property of remaining close to the desired trajectories.

The presented methodology is based on research works by Vyasarayani et al. It is initially presented in [118], mostly for mechanical systems with parameters that are allowed to enter nonlinearly into the dynamics. They present the main algorithm and validate it using several numerical examples. Further, they discuss the connection to the field of *homotopy continuation* [119]. In particular, the approach is introduced as a method of *natural parameter continuation* [33], and finally, compared to the *differential-equation-based continuation* approach [123]. Again, they verify their method by several numerical examples and also compare it to a linear regression approach. Finally, Vyasarayani et al. [117] investigate the method analytically and conclude that the originally highly nonlinear objective function becomes quadratic upon the introduction of an observer term in combination with a homotopy parameter. They present numerical results to support their findings. A similar ansatz is pursued by Leander et al. [72] for stochastic differential equations, for example.

In the following remark, a brief introduction to homotopy approaches is given.

Remark 4.4 For a given difficult problem, the general idea of homotopy methods is to solve an easy problem instead and to *transform* it back slowly into the original problem. Often, this mentioned transformation is alternatively stated as *deformation* or *morphing procedure*. Parts of the following short introduction to the topic stem from [4], to which we also refer for details. Originally, the difficult problem consists of solving a system of nonlinear equations

$$\mathcal{F}(v) = 0_{\mathcal{N}},$$

in which $\mathcal{F}\colon \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$ is a smooth function. If a zero point is at least approximately known, Newton's method can be used to compute a solution. However, this might not be the case for poor initial values, and therefore, the homotopy map $\mathcal{H}\colon \mathbb{R}^{\mathcal{N}} \times \mathbb{R} \to \mathbb{R}^{\mathcal{N}}$ with

$$\mathcal{H}(v, 1) = \mathcal{G}(v)$$

and

$$\mathcal{H}(v, 0) = \mathcal{F}(v)$$

is introduced with $\mathcal{G}\colon \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$ being a smooth function. It is important that $\mathcal{G}$ is

an *easy* function with known zero points. Often, the *convex homotopy* function

$$\mathcal{H}(v, \omega) \coloneqq \omega \mathcal{G}(v) + (1 - \omega)\mathcal{F}(v)$$

is used. Alternatively, the *global homotopy*

$$\mathcal{H}(v, \omega) \coloneqq \mathcal{F}(v) - \omega \mathcal{F}(\hat{v})$$

for a starting point $\hat{v}$ finds applications in many cases. To find a zero of $\mathcal{F}$, the idea is to trace a curve $c(s) \in \mathcal{H}^{-1}(0)$ from an initial guess $(v_{\text{ini}}, 1)$ to a solution $(v^\star, 0)$. This curve is (provided it exists) implicitly defined. Numerically, this is achieved by solving a sequence of problems, in which a solution is used as an initial guess for the next step, while slowly decreasing the homotopy parameter ω by a decrement $\Delta\omega$. One can show that under certain assumptions (especially a sufficiently small decrement), this process converges [4].

In [122], the problem under consideration is changed from finding zeros of nonlinear functions to solving optimization problems. For other problem classes, for instance parameter identification, an adequate homotopy map needs to be designed.

In the following, we present an extension to the single shooting homotopy algorithm from the literature. This extension is based on the idea of lifting, as described in Subsection 3.1.3. The homotopy parameter will be lifted to being an optimization variable, which reduces the effort for solving a series of NLPs (as originally proposed in the literature) to solving only a single, slightly larger one, which internally contains the before-mentioned homotopy continuation. All findings are accompanied by numerical examples.

4.6.1 Single Shooting Homotopy Method

In the introduction to parameter identification in Section 3.2, the transformation of the original problem takes place at transcription level. A different NLP formulation can offer advantages, for example the ability to find better solutions or a reduced computation time despite larger problem dimensions. Contrarily, the transformation presented here is performed at modeling level. The ODE in (3.3) is augmented by a so-called *observer term*. This has the effect that the behavior of the model is changed considerably. To solve the original problem, this newly added term is combined with a *homotopy parameter*, which is driven to zero while solving the perturbed problems.

In dynamical parameter identification problems, the shape of the objective function is not known in general and depends on many factors, for instance the length of the time interval, the measurement quality, or the accuracy of the integration scheme in use. It is also not obvious how to formulate a suitable function for the homotopy procedure. Thus, instead of modifying the objective function explicitly, the approach presented in [118] proposes to modify the model given by the dynamical system (3.3). Still, this has an implicit effect on the objective function due to the combination of integration and optimization within the single shooting method. In this approach, the right-hand side of the dynamical system is augmented by a so-called *observer term* $\mathfrak{e}\colon \mathbb{R} \to \mathbb{R}^{n_q}$ multiplied by a homotopy parameter $\omega \in [0,1]$ and a synchronization matrix $\mathfrak{K} \in \mathbb{R}^{n_q \times n_q}$ with non-negative entries:

$$\dot{q}(t) = s(t, q(t), p) + \omega \mathfrak{K} \mathfrak{e}(t). \tag{4.4}$$

Herein, the k-th component ($k \in \{1, \dots, n_q\}$) of the observer term $\mathfrak{e}(t)$ for $t \in [t_a, t_b]$ is defined as

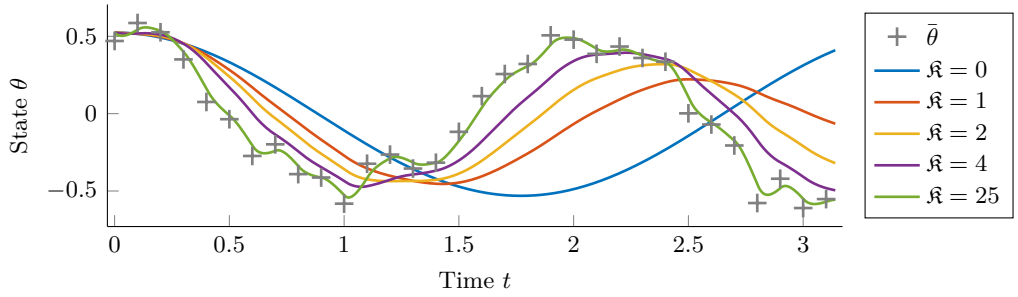
$$\mathfrak{e}_k(t) \coloneqq \begin{cases} 0 & \text{if } k \neq \mathcal{J}, \\ \bar{q}_k(t) - q_k(t) & \text{if } k \in \mathcal{J}, \end{cases}$$

in which $\bar{q}(t)$ describes an approximation to the measurements, for example via linear interpolation for t between two measurement points. In other words, the dynamical equations are augmented only for the observed states by the difference between measurements and states in combination with a homotopy parameter and a synchronization matrix.

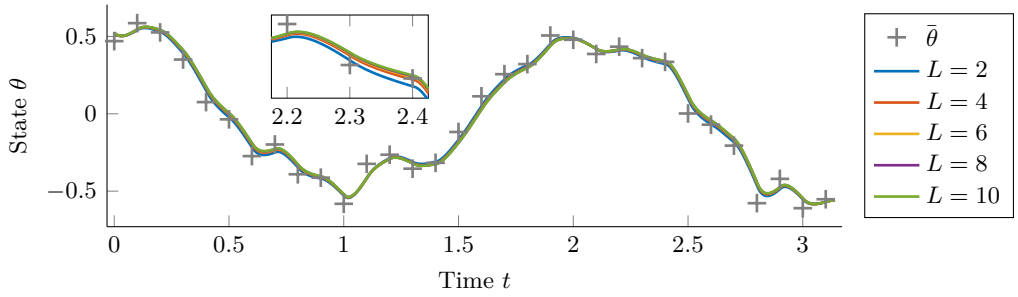
This has a strong effect on the dynamic behavior of the system. Obviously, for $\omega = 0$, the original system is recovered. For sufficiently large entries of $\mathfrak{K}$ and ω close to 1, the influence of the dynamics s is reduced, and a synchronization between observed states and measurements takes place. This also implies that the influence of the parameter vector p is reduced and the system is forced to stay close to the measurements, even if strongly differing parameter values (compared to the true ones) are assumed. To illustrate this effect, we use our pendulum example again.

Example 4.5

In this example, the influence of the synchronization matrix $\mathfrak{K}$ is investigated for the simple pendulum, see Example 3.10. We restrict ourselves to the case that $\mathfrak{K}$ is a scalar multiple of the identity matrix, and for the purpose of readability, the notation $\mathfrak{K}$ corresponds to both the matrix and the value of its elements. For



(a) Influence of the synchronization parameter $\mathfrak{K}$ on the model response for a fixed model parameter $L = 3$.



(b) Almost independence of the model parameter L on the model response for a fixed synchronization parameter $\mathfrak{K} = 25$.

Figure 4.23: Synchronization behavior of the extended dynamical model using the homotopy parameter value $\omega = 1$.

illustration purposes, we restrict the time interval to $[0, \pi]$. We again assume that there exist measurements for the angle θ corresponding to the true parameter value $L^\star = 1$. These are perturbed by an additive error with variance $\sigma^2 = 0.01$. We use the Explicit Euler scheme for ODE simulation. For $\omega = 1$, two scenarios are investigated:

(i) The behavior of the model for a fixed parameter $L = 3$ and varying synchronization matrices $\mathfrak{K}$, see Figure 4.23a.
(ii) The behavior of the model for fixed synchronization values $\mathfrak{K} = 25$ and varying parameter values L, see Figure 4.23b.

For $L = 3$ and $\mathfrak{K} = 0$ — which corresponds to the original model with a poor parameter guess, see also Figure 3.3 — the model cannot reproduce the measurements. A qualitatively different behavior is observed. For increasing values of $\mathfrak{K}$, however, the model response converges to the measurements, until a sufficient

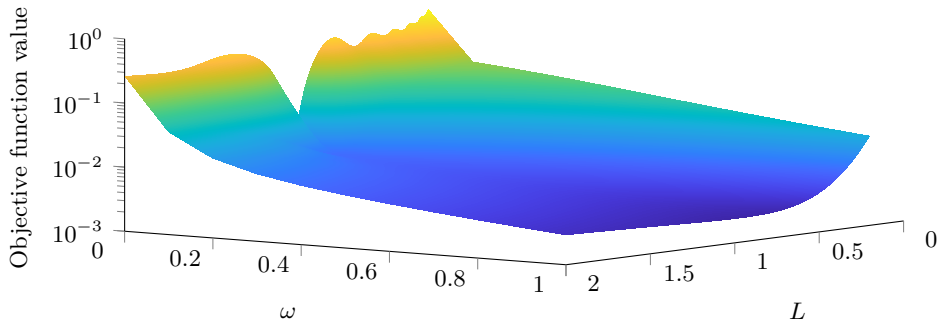


Figure 4.24: Shape of the objective function used in Example 4.6 with $\mathfrak{K} = 25$ and fixed initial values.

approximation is obtained at $\mathfrak{K} = 25$. Consequently, the influence of L almost vanishes. On the other hand, for fixed $\mathfrak{K} = 25$ corresponding to a dominating observer term, the model behaves almost independently of the parameter L and its response is almost identical for differing parameter values.

The modification of the dynamical system also impacts the corresponding objective function in (R-DPIP), as illustrated subsequently.

Example 4.6

We continue with Example 3.15 and focus on the objective function, which is shown in Figure 3.3 for the single shooting approach. Multiple local minima complicate the search for a good minimum if only a poor initial parameter guess is available. We can examine the shape of the introduced homotopy map (4.4) for an additional degree of freedom, the homotopy parameter ω. Figure 4.24 shows the objective function with synchronization values $\mathfrak{K} = 25$. As expected, the original shape (as in Figure 3.3) is recovered for $\omega = 0$. For $\omega = 1$, a quadratic shape is attained, and the only minimum is approximately at $L^\star = 1$. In between, the function is shaped like a valley.

The idea introduced by Vyasarayani et al. is to perform a numerical homotopy continuation in ω. For a given synchronization matrix $\mathfrak{K}$ with sufficiently large entries, a sufficiently small decrement $\Delta\omega$ and an initial value $\omega = 1$, the algorithm proceeds as follows: In a first step, the problem of consideration is solved to optimality. Next, the obtained solution $(q_a, p)^\star$ is used to initialize a following problem, in which ω is reduced by the amount $\Delta\omega$. This is iterated until the model is recovered

Algorithm D Homotopy continuation for single shooting

1: Given: a problem P in reduced formulation, $\mathfrak{K}$, and $(q_a, p)_{\text{ini}}$
2: Initialize $\omega = 1$ and $\Delta\omega$ satisfying $\omega \pmod{\Delta\omega} = 0$
3: **while** $\omega \geq 0$ **do**
4: Solve P, obtain a solution $(q_a, p)^\star$
5: Update $(q_a, p)_{\text{ini}} \leftarrow (q_a, p)^\star$
6: Update $\omega \leftarrow \omega - \Delta\omega$
7: **end while**

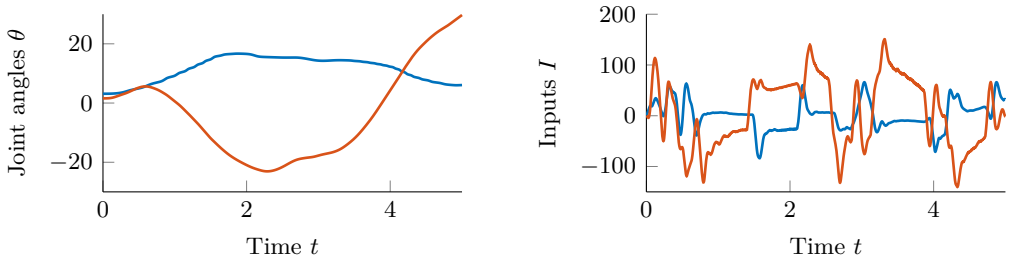
($\omega = 0$) and a solution of the original problem is found. The described procedure is summarized in Algorithm D.

If the algorithm parameters are chosen adequately, it can be shown that the procedure converges to the true solution. Local minima can be avoided by remaining close to the desired trajectory at each (homotopy) iteration step. Nevertheless, we are interested in finding decomposed problem formulations that can be characterized by a larger parameter space while leading to the same solutions. The described algorithm exhibits a different kind of complexity, hidden in Step 4. It requires the repeated application of an NLP solver for the numerical solution of a reduced dynamical parameter identification problem for varying parameters. In fact, the described procedure can be very inefficient. This depends mainly on the choice of $\Delta\omega$. For small decrement values, the reduced problem has to be solved many times. Although we expect that the solutions will not vary too much from each other, and therefore only a few iterations (at the NLP level) need to be applied, the overall performance can still be poor.

To circumvent these issues, we propose to modify the approach: instead of having a series of similar optimization problems, solve only one slightly larger problem using *parameter embedding*.

4.6.2 Homotopy Parameter Embedding

Although the approach described above is able to find desired solutions, it still exhibits disadvantages and leaves room for improvements. For small decrements, many problems have to be solved consecutively. Although our numerical observations indicate that small decrements are not necessary in the majority of cases, we suggest embedding the presented homotopy continuation into a more general nonlinear optimization framework. Up to now, ω is decreased by the same amount $\Delta\omega$ in each homotopy iteration. It is also conceivable that a much larger step towards



(a) Artificial state measurements. **(b)** Experimental control measurements.

Figure 4.25: Measurements used in Example 4.7.

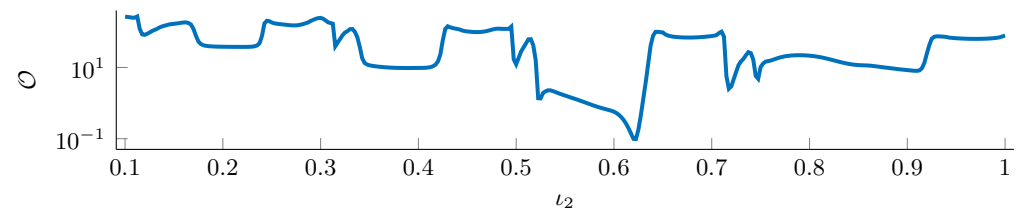
the desired value could be taken. Thus, we aim at an adjusted choice of homotopy parameters, which leads to more efficient homotopy paths. Possibly, this can achieve the same result by a smaller number of iterations. The general idea now is to include the homotopy parameter ω as an optimization variable in our problem formulation and take advantage of the fact that an initial guess must be chosen by initializing this variable as $\omega_{\text{ini}} = 1$. To recover the original problem formulation at the end of the optimization course, we add a penalty term $\rho\colon \mathbb{R} \to \mathbb{R}$ to the objective function. Since this transforms the strong condition $\omega = 0$ into a weak one, it has to be selected carefully. It also needs to be designed with the requirement that it vanishes for the desired homotopy value of zero, hence it should hold that

$$\rho(\omega = 0) = 0.$$

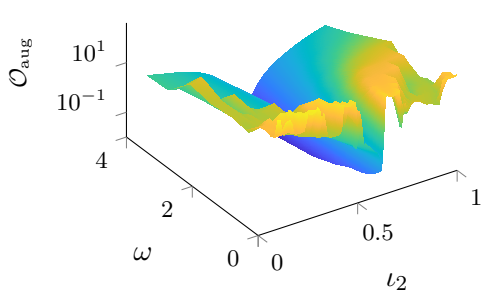
With the homotopy parameter embedded into the problem, the homotopy path is controllable by the solver and can be influenced by a user, for example by adjusting the penalty term. Furthermore, it is now possible to increase ω in the course of the optimization. Most of all, the series of problems is reduced to only one problem instance. We demonstrate the solution behavior with the help of an example.

Example 4.7

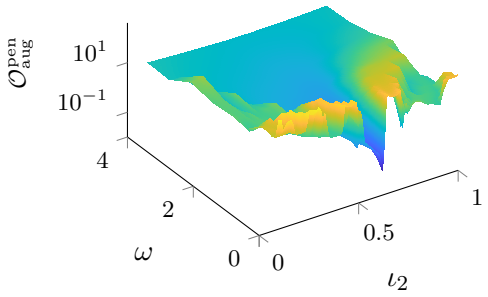
We consider the robotic system introduced in Section 4.2 with two links that are connected via joints. Given artificial joint angle measurements $\bar{\theta}$ (Figure 4.25a) and experimental input data I (Figure 4.25b) on the time interval $[0, 5]$, the aim is to identify only the single parameter ι_2 representing a moment of inertia of the second link. Its nominal value is 0.61.
This allows us to evaluate the corresponding objective function $\mathcal{O}$ at different values, see Figure 4.26a. In addition to the expected global minimum, the objective



(a) Single shooting.



(b) Homotopy single shooting.



(c) Homotopy-optimization single shooting.

Figure 4.26: Shapes of the objective functions used in Example 4.7.

function exhibits many local minima, which makes the parameter identification a challenging task for local methods. Augmenting the dynamics leads to an objective function $\mathcal{O}_{\text{aug}}$ (Figure 4.26b) that has a valley around the global minimum. The penalty term $\rho(\omega) = \omega$ gives the objective function $\mathcal{O}^{\text{pen}}_{\text{aug}}$ a different shape (Figure 4.26c): the valley remains close to $\omega = 0$, but transforms to an ascending plane for increasing values.
To demonstrate the algorithmic behavior, we set $\iota_{2\text{ini}} = 0.8$, $\omega_{\text{ini}} = 1$ using $\mathfrak{K} = 4$ and solve the problem with `WORHP`'s SQP method. As expected, single shooting converges quickly to a local minimum, while the homotopy formulations each find the global one (Figure 4.27). In this example, three steps are already sufficient to converge to the global minimum. We can observe that the presented method chooses a different homotopy path, which leads to a marginally faster convergence.[7]

Although the penalty term is responsible for driving the homotopy parameter to zero, there is no theoretical guarantee that an accurate value of zero is actually attained at the end of the optimization process. This is a major drawback of the

[7]This example is published by Sch., Flaßkamp, Fliege, and Büskens in [97].

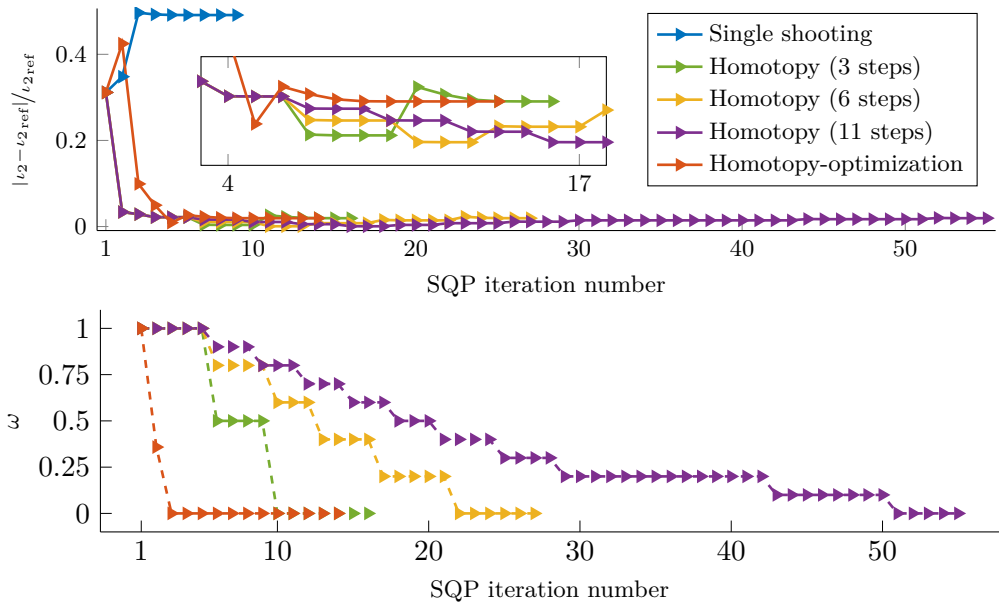


Figure 4.27: Course of SQP major iterations. Top: relative error in the model parameter ι_2. The box zooms onto the iterations 4 to 17. Bottom: value of the homotopy parameter ω.

proposed method. However, several strategies can be applied to make sure that the original problem really is solved. We propose to add a post-optimal solution analysis in case a homotopy parameter different from zero is computed. A promising strategy is to use the found solution as an initialization for the original problem formulation. If such a strategy is included, it makes sense to solve the initial problem on a coarse discretization grid for efficiency reasons. In the next step (with the adjusted initial guess), an advanced grid can be used to accurately approximate the underlying ODE model. Thus, the embedded homotopy-optimization can also be seen as an intelligent strategy to find a suitable initial guess.

Originally, the method was developed to overcome one of the disadvantages of reduced problem formulations, namely the convergence to undesired local solutions. However, we have seen in the previous sections that full discretization or multiple shooting can overcome this disadvantage. Nevertheless, it depends on the initialization which solutions are computed. To further improve the robustness, an extension would be to apply the combined homotopy-optimization also to full discretization. Since the necessary model modification takes place at the modeling level, it can be applied to all presented transcription techniques with minimal implementation effort.

Chapter 5

Conclusions

This chapter begins with a brief summary of the main results of this thesis, continues with a list of promising direct extensions, and ends with an outlook on potential future research directions.

5.1 Summary

The goal of this thesis was to develop reformulations of optimization problems that reliably find desired minima when standard nonlinear programming algorithms are used. To this end, the idea of decomposition was introduced: reformulating complex parts within a problem formulation by introducing new optimization variables and constraints. Since local algorithms are used, one goal was to overcome early convergence to undesired local solutions and to enable the solver to converge to desired minima. We considered two classes of problems in which decomposition can be applied.

In the area of parameter identification for dynamical systems, transcription techniques were surveyed under the aspects of local solutions, sparsity, and initialization possibilities. A numerical study on a mathematical pendulum showed that the introduction of shooting nodes in the direct multiple shooting approach in combination with a suitable variable initialization increases the region of attraction of the global minimum. Although the problem dimensions are significantly increased in the full discretization approach, the computation time could be kept desirably low. Moreover, a stabilized convergence process, indicated by the required number of iterations, could be observed. The decomposed approach requires only a few iterations, regardless of the solver initialization, in contrast to the reduced version. For a more complex robotic system with real measurements, the findings could be confirmed. A higher degree of decomposition leads to a higher robustness of the

solver and allows finding solutions with lower objective values. These investigations were based on Monte–Carlo-type experiments, in which the initialization of the primary variables was varied, while the secondary variables were initialized in a customized, advantageous way. Furthermore, a combined homotopy-optimization approach was developed. The idea is to treat a homotopy continuation parameter in the dynamical system as an additional optimization variable, thus reducing the number of problems to one. Initial experiments with the reduced approach provided a proof of concept.

In the bilevel optimization part, the focus was on developing a novel reformulation technique based on the idea of decomposition. While the reduced approach applies an NLP algorithm in the background to compute a KKT point, the KKT approach includes the necessary optimality conditions directly in the problem. The idea of the decomposed approach is to embed a given number of SQP steps for computing a KKT point into the problem. An extension of this formulation is a combination with the KKT approach. Monte–Carlo-type experiments varying the upper-level variables revealed interesting aspects for all considered problem formulations. In the case of a non-unique lower-level problem, an example demonstrates that the KKT approach may fail, while the reduced and the decomposed approaches perform advantageously. Another example focuses on the quality of solutions. Here, the region of attraction of the global minimum can be increased by incorporating more lower-level iteration steps into the problem. Finally, the decomposed approaches as well as the other investigated methods are applied to a library of bilevel problems. The results show that the newly developed reformulation technique is able to compete with existing approaches. For certain solution strategies even a superior behavior could be observed. With regard to the computation time, however, the decomposed approaches cannot compete with the other ones. Nevertheless, it is interesting to see that the KKT approach performs significantly better than the reduced formulation, even though the problem dimensions are increased. A superior convergence speed, indicated numerically by the required number of iterations, is one of the reasons for this.

It has been demonstrated that these two problem classes — despite their different nature — can be viewed from a common point and decomposition can be applied. Only by reformulating a given problem, we have managed to make the solution process more robust and to enlarge the region of attraction, while keeping the computational effort moderate. All in all, the hypotheses made in this work have been numerically confirmed.

5.2 Outlook

The concept of decomposition opens many doors for future research. Some extensions are listed and discussed below. A broader perspective is then taken to comment on possible future research directions.

Direct Extensions

In the field of parameter identification, it has been demonstrated that decomposed formulations can be advantageous in several facets. However, they are associated with an increased implementation effort. The question in which situations the reduced approach might be an adequate choice is still open. An intuitive answer would be to have an initial guess of the model parameters that is already close to the solution and thus local minima can be ignored. However, other factors may influence the algorithmic behavior, such as the dimensions of the dynamical system in the background, the measurement errors, the length of the time interval, or the algorithm used. Therefore, it would be desirable to have an automated decision-making process for the most promising formulation based on hyperparameters of the problem, for example based on machine learning.

In the multiple shooting method, the level of decomposition can be scaled by the number of shooting nodes. It is, however, not obvious how to choose this number. On the one hand, one aims to keep the problem dimensions small, but on the other hand, the findings in this work suggest incorporating many nodes to overcome local solutions. An automated decision-making process as described above could also be helpful here. Another idea is to modify the algorithm properties, which was not considered in this work. By using a penalty method instead of SQP in combination with a low weighting of the constraints' penalty term, the solver could focus on minimizing the objective function instead of fulfilling the constraints. Thus, adding only a single shooting node ($N_\mathrm{s} = 2$) could break up the rigid reduced objective function and pave the way to a global solution.

The results demonstrate that decomposition within ODE-constrained optimization is able to influence the algorithmic behavior. Hence, it would be desirable to transfer these effects to other problem classes, as it is already done in the area of recurrent neural networks. In [110], it is shown that the full discretization approach enables a reliable convergence to desired solutions. Another idea for transfer may lie in general nonlinear programs involving the term $\exp(z)$. If instead we consider the ODE $\dot{y}(t) = zy(t)$ with $y(0) = 1$ and $t \in [0, 1]$, we could replace $\exp(z)$ by the ODE's solution $y(t) = \exp(zt)$ evaluated at $t = 1$. Next, the IVP would be treated

as additional constraints, for which again a full discretization transcription can be applied. It would be interesting to investigate whether this "shift of nonlinearities" leads to changes in the algorithmic behavior.

In bilevel optimization, it would be of great interest to see if the idea of multiple shooting applies here as well. Following this, it could be fruitful to hide some lower-level iteration steps from the solver and only embed continuity constraints making sure that the iteration steps are connected. This would even allow using appropriate NLP solvers and not being restricted to a full-step exact-Hessian SQP method.

The former possible extension illuminates a drawback of the decomposed approach. In its current form, it makes use of a full-step exact-Hessian SQP algorithm. However, it is standard to consider a line search for the step sizes α_k in the update step $y_{k+1} = y_k + \alpha_k d_k$. It is usually designed to take a sufficient decrease in both the objective function and constraint violation into account. For this, a merit function is minimized, an inexact solution is often sufficient. Future studies should concentrate on incorporating such a line search into the problem. Additional variables representing the step sizes at the respective iteration steps would have to be introduced. They would be subject to conditions making sure that a sufficient decrease is made in each iteration. As exact line searches (for example via Wolfe conditions [126]) are often avoided due to performance reasons, they could easily be included here via suitable constraints. Hence, the step sizes would be identified simultaneously with the other variables. Another promising extension is to use approximations of the lower-level Hessian matrix instead of its analytic computation. In analogy to the line search, future studies should examine how to embed update formulae like BFGS into the decomposed problem.

A natural alternative to embedding SQP algorithms is using IP methods, in particular primal-dual approaches. The corresponding iteration steps would have to be included similarly. A comparison of both approaches using the problem library could lead to interesting insights. Besides, this could be the starting point for a generalization of the concept using arbitrary lower-level solution methods.

The decomposition for bilevel optimization is a heuristic approach. It is based on the idea of full discretization within ODE-constrained optimization. The numerical results demonstrate the strengths of this reformulation and provide more than only a proof of concept. Still, a theoretical analysis is needed to validate the observed effects. Constraint qualifications, necessary optimality conditions, or convergence proofs of the presented strategies will be important issues for future research.

Independent of the problem class, derivatives (apart from structural aspects) have found less attention in this work. They are approximated via finite differences within the chosen solver. However, an exact computation could bring several advantages, for example increased precision, but also reduced computation times, especially in the reduced approaches. In parameter identification, differentiating the numerical integration scheme would be an interesting feature. In bilevel programming, the use of parametric sensitivities for the reduced approach could be a useful extension. Providing automatic differentiation approaches could also be advantageous here.

Widening the Perspective

The idea of decomposition has been applied to specific use cases in this work: underlying iterative procedures have been decomposed and lifted into the problem formulation. The realization had to be done manually with a lot of attention to detail. Therefore, future studies should aim at an abstraction of this concept. The automatic embedding of any iterative procedure, independent of its actual appearance, would be an interesting next step. Steps in this direction have already been taken, for example in [90] with a background in bilevel optimization, or in [1] as generalizations of multiple shooting.

Not only iterative procedures, but also general nonlinear terms can be decomposed. This is also mentioned in [1] and the effects are demonstrated exemplarily. It raises the question under which circumstances this is advantageous, or whether a reduced version might be sufficient. Thus, it would be desirable to have a measure that evaluates the complexity of a given expression. Concepts for this exist, among others, in the theory of dynamical systems via Lyapunov exponents or linear approximation distances [60]. Conversely, it would also be interesting to measure the complexity of a given decomposition and make comparisons with its reduced counterpart. Based on these measures, one would then have to find criteria that predict when or even to what extent an expression can be decomposed.

The approach in this work has been to influence the computation of a desired solution, in our case the global one, by reformulating a problem and then applying an algorithm. In fact, the algorithm can play an important role in this scenario as well. This property is called “implicit bias” and is often mentioned in the field of deep learning (for example [22]) as well as in single-level or bilevel optimization (see [54] or [115]). It is shown that the gradient descent method, for instance, has a bias towards convergence to certain solutions. This encourages the investigation of the role that algorithms play in this work. It could be studied whether the algorithm — biased towards a specific solution — and the formulation — aimed at

increasing the global solution's region of attraction — work towards conflicting goals or whether they support each other.

This hands over to a more detailed look at those certain, specific, or desired solutions. All these terms aim at describing the quality of a solution, which is usually given by the corresponding objective function value as a primary criterion. However, other secondary criteria may also be of interest, and a desired solution is said to satisfy both. In such a setting, a global solution might turn out to be inferior to a local solution that performs better on the secondary criterion. Although in this work the global solution is the desired one, a secondary criterion has been implicitly taken into account: by applying decomposition, we increased the region of attraction for this solution. By changing the perspective, one could make this consideration explicit, based on the following idea. For a fixed problem formulation, a secondary criterion is defined and applied to all occurring solutions to measure their quality. Here, the works of Mykhailiuk et al. [84, 85] should be mentioned. The so-called "parametric stability score" is introduced as a secondary quality criterion for NLP solutions to quantify the influence of parameter perturbations. It is shown that a global solution is not necessarily the desired one, as other local solutions may have higher scores. For future research, it would be interesting to combine both approaches by explicitly considering secondary quality criteria in the formulation design.

The potential future research directions described above cover the aspects mentioned at the beginning of this thesis (Figure 1.1): "formulation", "algorithm", and "solution". In the long run, the combination of these aspects could lead to a holistic approach to solving application-based nonlinear optimization problems.

Bibliography

[1] J. Albersmeyer and M. Diehl. "The Lifted Newton Method and Its Application in Optimization". In: *SIAM Journal on Optimization* 20.3 (2010), pp. 1655–1684. DOI: 10.1137/080724885.

[2] S. Albrecht, K. Ramírez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz. "Imitating human reaching motions using physically inspired optimization principles". In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. 2011, pp. 602–607. DOI: 10.1109/Humanoids.2011.6100856.

[3] G. B. Allende and G. Still. "Solving bilevel programs with the KKT-approach". In: *Mathematical Programming* 138 (2013), pp. 309–332. DOI: 10.1007/s10107-012-0535-x.

[4] E. L. Allgower and K. Georg. *Numerical Continuation Methods. An Introduction*. Springer Berlin, Heidelberg, 1990.

[5] M. Amouzegar and K. Moshirvaziri. "Determining Optimal Pollution Control Policies: An Application of Bilevel Programming". In: *European Journal of Operational Research* 119.1 (1999), pp. 100–120. DOI: 10.1016/S0377-2217(98)00336-1.

[6] J. F. Bard. "Convex two-level optimization". In: *Mathematical Programming* 40 (1988), pp. 15–27. DOI: 10.1007/bf01580720.

[7] I. Bauer, H. G. Bock, S. Körkel, and J. P. Schlöder. "Numerical methods for optimum experimental design in DAE systems". In: *Journal of Computational and Applied Mathematics* 120.1 (2000), pp. 1–25. DOI: 10.1016/S0377-0427(00)00300-9.

[8] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2nd ed. Society for Industrial and Applied Mathematics, 2010. DOI: 10.1137/1.9780898718577.

[9] L. T. Biegler. "An overview of simultaneous strategies for dynamic optimization". In: *Chemical Engineering and Processing: Process Intensification* 46.11 (2007). Special Issue on Process Optimization and Control in Chemical Engineering and Processing, pp. 1043–1053. DOI: 10.1016/j.cep.2006.06.021.

[10] L. T. Biegler, J. Nocedal, and C. Schmid. "A Reduced Hessian Method for Large-Scale Constrained Optimization". In: *SIAM Journal on Optimization* 5.2 (1995), pp. 314–347. DOI: 10.1137/0805017.

[11] H. G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*. Vol. 183. Bonner Mathematische Schriften. Universität Bonn, 1987.

[12] H. G. Bock. "Recent Advances in Parameteridentification Techniques for O.D.E." In: *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Ed. by P. Deuflhard and E. Hairer. Vol. 2. Progress in Scientific Computing. Birkhäuser Boston, 1983, pp. 95–121. DOI: 10.1007/978-1-4684-7324-7_7.

[13] H. G. Bock, E. Kostina, and J. P. Schlöder. "Direct Multiple Shooting and Generalized Gauss–Newton Method for Parameter Estimation Problems in ODE Models". In: *Multiple Shooting and Time Domain Decomposition Methods*. Ed. by T. Carraro, M. Geiger, S. Körkel, and R. Rannacher. Springer International Publishing, 2015, pp. 1–34. DOI: 10.1007/978-3-319-23321-5_1.

[14] H. G. Bock, E. Kostina, and J. P. Schlöder. "Numerical Methods for Parameter Estimation in Nonlinear Differential Algebraic Equations". In: *GAMM-Mitteilungen* 30.2 (2007), pp. 376–408. DOI: 10.1002/gamm.200790024.

[15] M. Boddy and T. Dean. "Solving Time-Dependent Planning Problems". In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Vol. 2. IJCAI'89. Detroit, Michigan: Morgan Kaufmann Publishers Inc., 1989, pp. 979–984.

[16] P. T. Boggs and J. W. Tolle. "Sequential Quadratic Programming". In: *Acta Numerica* 4 (1995), pp. 1–51. DOI: 10.1017/S0962492900002518.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends® in Machine Learning* 3 (2011), pp. 1–122. DOI: 10.1561/2200000016.

[18] C. Büskens. "Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen". PhD thesis. Universität Münster, 1985.

[19] C. Büskens and H. Maurer. "Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems". In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. Springer Berlin Heidelberg, 2001, pp. 3–16. DOI: 10.1007/978-3-662-04331-8_1.

[20] C. Büskens and D. Wassel. "The ESA NLP Solver WORHP". In: *Modeling and Optimization in Space Engineering*. Ed. by G. Fasano and J. D. Pintér. Springer New York, 2013, pp. 85–110. DOI: 10.1007/978-1-4614-4469-5_4.

[21] R. H. Byrd and J. Nocedal. "An analysis of reduced Hessian methods for constrained optimization". In: *Mathematical Programming* 49.1 (1990), pp. 285–323. DOI: 10.1007/BF01588794.

[22] H.-H. Chou, C. Gieshoff, J. Maly, and H. Rauhut. *Gradient Descent for Deep Matrix Factorization: Dynamics and Implicit Bias towards Low Rank*. 2021. DOI: 10.48550/arXiv.2011.13772.

[23] P. A. Clark and A. W. Westerberg. "Bilevel programming for steady-state chemical process design—I. Fundamentals and algorithms". In: *Computers & Chemical Engineering* 14 (1990), pp. 87–97. DOI: 10.1016/0098-1354(90)87007-C.

[24] D. Colton and R. Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer New York, 2013. DOI: 10.1007/978-1-4614-4942-3.

[25] W. P. Coutinho, J. Fliege, and M. Battarra. "Glider Routing and Trajectory Optimisation in disaster assessment". In: *European Journal of Operational Research* 274.3 (2019), pp. 1138–1154. DOI: 10.1016/j.ejor.2018.10.057.

[26] J. M. A. Danby. *Fundamentals of Celestial Mechanics*. 2nd ed. Richmond, Va.: Willmann-Bell, 2003.

[27] S. Dempe and J. Dutta. "Is bilevel programming a special case of a mathematical program with complementarity constraints?" In: *Mathematical Programming* 131.1 (2012), pp. 37–48. DOI: 10.1007/s10107-010-0342-1.

[28] S. Dempe and A. B. Zemkoho. "The bilevel programming problem: reformulations, constraint qualifications and optimality conditions". In: *Mathematical Programming* 138.1 (2013), pp. 447–473. DOI: 10.1007/s10107-011-0508-5.

[29] S. Dempe. "An Implicit Function Approach to Bilevel Programming Problems". In: *Multilevel Optimization: Algorithms and Applications*. Ed. by A. Migdalas, P. M. Pardalos, and P. Värbrand. Boston, MA: Springer US, 1998, pp. 273–294. DOI: 10.1007/978-1-4613-0307-7_12.

[30] S. Dempe. *Foundations of Bilevel Programming*. Nonconvex Optimization and Its Applications. Springer New York, NY, 2002. DOI: 10.1007/b101970.

[31] E. Dierkes, C. Meerpohl, K. Flaßkamp, and C. Büskens. "Estimation and Mapping of System-Surface Interaction by Combining Nonlinear Optimization and Machine Learning". In: *IFAC-PapersOnLine* 54.14 (2021). 3rd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2021, pp. 138–143. DOI: 10.1016/j.ifacol.2021.10.342.

[32] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical Programming* 91.2 (2002), pp. 201–213. DOI: 10.1007/s101070100263.

[33] D. M. Dunlavy and D. P. O'Leary. *Homotopy optimization methods for global optimization.* Tech. rep. 2005. DOI: 10.2172/876373.

[34] J. Dutta, L. Lafhim, A. B. Zemkoho, and S. Zhou. *Nonconvex quasi-variational inequalities: stability analysis and application to numerical optimization*. 2022. DOI: 10.48550/arXiv.2210.02531.

[35] M. Echim. "Modellbasierte optimale Mehrgrößenregelung eines aufgeladenen Dieselmotors mittels Methoden der nichtlinearen Optimierung". PhD thesis. University of Bremen, 2014.

[36] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. "qpOASES: A parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363. DOI: 10.1007/s12532-014-0071-1.

[37] A. Fischer. "A special newton-type optimization method". In: *Optimization* 24.3-4 (1992), pp. 269–284. DOI: 10.1080/02331939208843795.

[38] A. Fischer, A. B. Zemkoho, and S. Zhou. "Semismooth Newton-type method for bilevel optimization: global convergence and extensive numerical experiments". In: *Optimization Methods and Software* 37.5 (2022), pp. 1770–1804. DOI: 10.1080/10556788.2021.1977810.

[39] K. Flaßkamp, K. Schäfer, and C. Büskens. "Variational Integrators for Parameter Identification of Mechanical Systems". In: *Proceedings in Applied Mathematics and Mechanics* 18.1 (2018), e201800284. DOI: 10.1002/pamm.201800284.

[40] R. Fletcher. *Practical methods of optimization*. 2nd ed. Chichester: John Wiley & Sons, Ltd, 1987.

[41] R. Fletcher. "Restricted Step Methods". In: *Practical Methods of Optimization*. John Wiley & Sons, Ltd, 2000. Chap. 5, pp. 95–109. DOI: 10.1002/9781118723203.ch5.

[42] R. Fletcher and M. J. D. Powell. "A Rapidly Convergent Descent Method for Minimization". In: *The Computer Journal* 6.2 (1963), pp. 163–168. DOI: 10.1093/comjnl/6.2.163.

[43] J. Fliege, A. Tin, and A. B. Zemkoho. "Gauss–Newton-type methods for bilevel optimization". In: *Computational Optimization and Applications* 78.3 (2021), pp. 793–824. DOI: 10.1007/s10589-020-00254-3.

[44] A. Forsgren, P. E. Gill, and M. H. Wright. "Interior Methods for Nonlinear Optimization". In: *SIAM Review* 44.4 (2002), pp. 525–597. DOI: 10.1137/S0036144502414942.

[45] A. Frantsev, A. Sinha, and P. Malo. "Finding Optimal Strategies in Multi-period Stackelberg Games Using an Evolutionary Framework". In: *IFAC Proceedings Volumes* 45.25 (2012). 15th IFAC Workshop on Control Applications of Optimization, pp. 33–38. DOI: 10.3182/20120913-4-IT-4027.00038.

[46] G. Frison and M. Diehl. "HPIPM: a high-performance quadratic programming framework for model predictive control". In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 6563–6569. DOI: 10.1016/j.ifacol.2020.12.073.

[47] A. Gábor and J. R. Banga. "Robust and efficient parameter estimation in dynamic models of biological systems". In: *BMC Systems Biology* 9.1 (2015). DOI: 10.1186/s12918-015-0219-2.

[48] S. Geffken. "Effizienzsteigerung numerischer Verfahren der nichtlinearen Optimierung". PhD thesis. University of Bremen, 2017.

[49] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer Berlin, Heidelberg, 2002. DOI: 10.1007/978-3-642-56004-0.

[50] M. E. Geiger. "Adaptive Multiple Shooting for Boundary Value Problems and Constrained Parabolic Optimization Problems". PhD thesis. University of Heidelberg, 2015.

[51] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press Inc., 1981.

[52] C. Göttlicher, M. Gnoth, M. Bittner, and F. Holzapfel. "Aircraft Parameter Estimation Using Optimal Control Methods". In: *AIAA Atmospheric Flight Mechanics Conference*. 2016. DOI: 10.2514/6.2016-1534.

[53] A. Griewank and A. Walther. *Evaluating Derivatives*. 2nd ed. Society for Industrial and Applied Mathematics, 2008. DOI: 10.1137/1.9780898717761.

[54] S. Gunasekar, J. Lee, D. Soudry, and N. Srebro. "Characterizing Implicit Bias in Terms of Optimization Geometry". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1832–1841.

[55] J. Hadamard. "Sur les problèmes aux dérivés partielles et leur signification physique". In: *Princeton University Bulletin* 13 (1902), pp. 49–52.

[56] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations I*. Springer Berlin, Heidelberg, 1993. DOI: 10.1007/978-3-540-78862-1.

[57] F. Hamilton. "Parameter Estimation in Differential Equations: A Numerical Study of Shooting Methods". In: *SIAM Undergraduate Research Online* 4 (2011). DOI: 10.1137/10S010739.

[58] M. Heinkenschloss. *Numerical Solution of Implicitly Constrained Optimization Problems*. Tech. rep. CAAM Technical Report TR08-05, 2008.

[59] M. Heinkenschloss and L. N. Vicente. "An Interface Optimization and Application for the Numerical Solution of Optimal Control Problems". In: *ACM Transactions on Mathematical Software* 25.2 (1999), pp. 157–190. DOI: 10.1145/317275.317278.

[60] A. Helbig, W. Marquardt, and F. Allgöwer. "Nonlinearity measures: definition, computation and applications". In: *Journal of Process Control* 10.2 (2000), pp. 113–123. DOI: 10.1016/S0959-1524(99)00033-5.

[61] R. J., K. Gupta, H. S. Kusumakar, V. K. Jayaraman, and B. D. Kulkarni. "A Tabu Search Based Approach for Solving a Class of Bilevel Programming Problems in Chemical Engineering". In: *Journal of Heuristics* 9.4 (2003), pp. 307–319. DOI: 10.1023/a:1025699819419.

[62] M. Jacobse and C. Büskens. "Revisiting Design Aspects of a QP Solver for WORHP". In: *7th International Conference on Astrodynamics Tools and Techniques (ICATT)*. 2018.

[63] B. Kaltenbacher. "All-at-once versus reduced iterative methods for time dependent inverse problems". In: *Inverse Problems* 33.6 (2017). DOI: 10.1088/1361-6420/aa6f34.

[64] C. Kanzow. "Some Noninterior Continuation Methods for Linear Complementarity Problems". In: *SIAM Journal on Matrix Analysis and Applications* 17.4 (1996), pp. 851–868. DOI: 10.1137/s0895479894273134.

[65] H. B. Keller. *Numerical Methods for Two Point Boundary Value Problems*. New York: Blaisdell, 1968.

[66] Y. Kim, S. Leyffer, and T. Munson. "MPEC Methods for Bilevel Optimization Problems". In: *Bilevel Optimization: Advances and Next Challenges*. Ed. by S. Dempe and A. B. Zemkoho. Springer International Publishing, 2020, pp. 335–360. DOI: 10.1007/978-3-030-52119-6_12.

[67] P.-M. Kleniati and C. S. Adjiman. "Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: Convergence analysis and numerical results". In: *Journal of Global Optimization* 60.03 (2014), pp. 459–481. DOI: 10.1007/s10898-013-0120-8.

[68] P.-M. Kleniati and C. S. Adjiman. "Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: Theoretical development". In: *Journal of Global Optimization* 60.03 (2014), pp. 425–458. DOI: 10.1007/s10898-013-0121-7.

[69] M. Knauer and C. Büskens. "Real-Time Optimal Control Using TransWORHP and WORHP Zen". In: *Modeling and Optimization in Space Engineering: State of the Art and New Challenges*. Ed. by G. Fasano and J. D. Pintér. Springer International Publishing, 2019, pp. 211–232. DOI: 10.1007/978-3-030-10501-3_9.

[70] E. Kostina. "Robust Parameter Estimation in Dynamic Systems". In: *Optimization and Engineering* 5.4 (2004), pp. 461–484. DOI: 10.1023/B:OPTE.0000042035.67293.92.

[71] R. Kuhlmann and C. Büskens. "A primal-dual augmented Lagrangian penalty-interior-point filter line search algorithm". In: *Mathematical Methods of Operations Research* 87.3 (2018), pp. 451–483. DOI: 10.1007/s00186-017-0625-x.

[72] J. Leander, T. Lundh, and M. Jirstrand. "Stochastic differential equations as a tool to regularize the parameter estimation problem for continuous time dynamical systems given discrete time measurements". In: *Mathematical Biosciences* 251 (2014), pp. 54–62. DOI: 10.1016/j.mbs.2014.03.001.

[73] H. Li and Y. Wang. "A Hybrid Genetic Algorithm for Solving Nonlinear Bilevel Programming Problems Based on the Simplex Method". In: *Third International Conference on Natural Computation (ICNC 2007)*. Vol. 4. 2007, pp. 91–95. DOI: 10.1109/ICNC.2007.48.

[74] T.-W. Liu. "A reduced Hessian SQP method for inequality constrained optimization". In: *Computational Optimization and Applications* 49.1 (2011), pp. 31–59. DOI: 10.1007/s10589-009-9285-y.

[75] R. Manikantan, S. Chakraborty, T. K. Uchida, and C. P. Vyasarayani. "Parameter Identification in Nonlinear Mechanical Systems with Noisy Partial State Measurement Using PID-Controller Penalty Functions". In: *Mathematics* 8.7 (2020). DOI: 10.3390/math8071084.

[76] D. W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[77] C. Michalik, R. Hannemann, and W. Marquardt. "Incremental single shooting — A robust method for the estimation of parameters in dynamical systems". In: *Computers & Chemical Engineering* 33.7 (2009), pp. 1298–1305. DOI: 10.1016/j.compchemeng.2009.02.002.

[78] A. Mitsos and P. I. Barton. *A Test Set for Bilevel Programs*. Tech. rep. MIT, 2006.

[79] A. Mitsos, P. Lemonidis, and P. I. Barton. "Global solution of bilevel programs with a nonconvex inner program". In: *Journal of Global Optimization* 42.04 (2008), pp. 475–513. DOI: 10.1007/s10898-007-9260-z.

[80] C. G. Moles, P. Mendes, and J. R. Banga. "Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods". In: *Genome Research* 13.11 (2003), pp. 2467–2474. DOI: 10.1101/gr.1262503.

[81] T. G. Müller and J. Timmer. "Parameter Identification Techniques For Partial Differential Equations". In: *International Journal of Bifurcation and Chaos* 14.06 (2004), pp. 2053–2060. DOI: 10.1142/S0218127404010424.

[82] G. Muñoz, D. Salas, and A. Svensson. "Exploiting the Polyhedral Geometry of Stochastic Linear Bilevel Programming". In: *Integer Programming and Combinatorial Optimization*. Ed. by A. Del Pia and V. Kaibel. Springer International Publishing, 2023, pp. 363–377. DOI: 10.1007/978-3-031-32726-1_26.

[83] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. 1st ed. Boca Raton, FL, USA: CRC Press, 1994. DOI: 10.1201/9781315136370.

[84] I. Mykhailiuk, K. Schäfer, and C. Büskens. "Parametric Stability Score and Its Application in Optimal Control". In: *IFAC-PapersOnLine* 55.16 (2022). 18th IFAC Workshop on Control Applications of Optimization CAO 2022, pp. 172–177. DOI: 10.1016/j.ifacol.2022.09.019.

[85] I. Mykhailiuk, K. Schäfer, and C. Büskens. "Stability Score for Local Solutions of Unconstrained Parametric Nonlinear Programs". In: *Proceedings in Applied Mathematics and Mechanics* 21.1 (2021), e202100215. DOI: 10.1002/pamm.202100215.

[86] I. Mykhailiuk, K. Schäfer, K. Flaßkamp, and C. Büskens. "On the Computation of Convergence Regions for Sequential Nonlinear Programming Problems". In: *Proceedings in Applied Mathematics and Mechanics* 20.1 (2021), e202000281. DOI: 10.1002/pamm.202000281.

[87] A. Neumaier. "Complete search in continuous global optimization and constraint satisfaction". In: *Acta Numerica* 13 (2004), pp. 271–369. DOI: 10.1017/S0962492904000194.

[88] J. Nocedal and M. L. Overton. "Projected Hessian Updating Algorithms for Nonlinearly Constrained Optimization". In: *SIAM Journal on Numerical Analysis* 22.5 (1985), pp. 821–850. DOI: 10.1137/0722050.

[89] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd ed. Springer New York, NY, 2006. DOI: 10.1007/978-0-387-40065-5.

[90] P. Ochs, R. Ranftl, T. Brox, and T. Pock. "Techniques for Gradient-Based Bilevel Optimization with Non-smooth Lower Level Problems". In: *Journal of Mathematical Imaging and Vision* 56.2 (2016), pp. 175–194. DOI: 10.1007/s10851-016-0663-7.

[91] M. Peifer and J. Timmer. "Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting". In: *IET Systems Biology* 1.2 (2007), pp. 78–88. DOI: 10.1049/iet-syb:20060067.

[92] A. A. Poyton, M. S. Varziri, K. B. McAuley, P. J. McLellan, and J. O. Ramsay. "Parameter estimation in continuous-time dynamic models using principal differential analysis". In: *Computers & Chemical Engineering* 30.4 (2006), pp. 698–708. DOI: 10.1016/j.compchemeng.2005.11.008.

[93] A. H. Ribeiro and L. A. Aguirre. "Shooting Methods for Parameter Estimation of Output Error Models". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 13998–14003. DOI: 10.1016/j.ifacol.2017.08.2421.

[94] N. V. Sahinidis. "BARON: A general purpose global optimization software package". In: *Journal of Global Optimization* 8.2 (1996), pp. 201–205. DOI: 10.1007/BF00138693.

[95] K. Schäfer, M. Runge, K. Flaßkamp, and C. Büskens. "Parameter Identification for Dynamical Systems Using Optimal Control Techniques". In: *2018 European Control Conference (ECC)*. 2018, pp. 137–142. DOI: 10.23919/ECC.2018.8550045.

[96] K. Schäfer, K. Flaßkamp, and C. Büskens. "A Numerical Study of the Robustness of Transcription Methods for Parameter Identification Problems". In: *Proceedings in Applied Mathematics and Mechanics* 18.1 (2018), e201800101. DOI: 10.1002/pamm.201800101.

[97] K. Schäfer, K. Flaßkamp, J. Fliege, and C. Büskens. "A Combined Homotopy-Optimization Approach to Parameter Identification for Dynamical Systems". In: *Proceedings in Applied Mathematics and Mechanics* 19.1 (2019), e201900266. DOI: 10.1002/pamm.201900266.

[98] K. Schäfer, J. Fliege, K. Flaßkamp, and C. Büskens. "Reformulating Bilevel Problems by SQP Embedding". In: *Proceedings in Applied Mathematics and Mechanics* 20.1 (2021), e202000302. DOI: 10.1002/pamm.202000302.

[99] K. Schittkowski. "Solving Constrained Nonlinear Least Squares Problems by a General Purpose SQP-Method". In: *Trends in Mathematical Optimization*. Ed. by K.-H. Hoffmann, J. Zowe, J.-B. Hiriart-Urruty, and C. Lemarechal. Vol. 84. International Series of Numerical Mathematics. Birkhäuser Basel, 1988, pp. 295–309. DOI: 10.1007/978-3-0348-9297-1_19.

[100] V. H. Schulz. "Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots". PhD thesis. Universität Heidelberg, 1996.

[101] V. Schulz and H. G. Bock. "Partially reduced sqp methods for large-scale nonlinear optimization problems". In: *Nonlinear Analysis: Theory, Methods & Applications* 30.8 (1997). Proceedings of the Second World Congress of Nonlinear Analysts, pp. 4723–4734. DOI: 10.1016/S0362-546X(97)00198-3.

[102] K. Shimizu, Y. Ishizuka, and J. F. Bard. *Nondifferentiable and Two-Level Mathematical Programming*. Springer New York, NY, 1997. DOI: 10.1007/978-1-4615-6305-1.

[103] A. Sinha, P. Malo, and K. Deb. "An improved bilevel evolutionary algorithm based on Quadratic Approximations". In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. 2014, pp. 1870–1877. DOI: 10.1109/CEC.2014.6900391.

[104] A. Sinha, P. Malo, and K. Deb. "A review on bilevel optimization: from classical to evolutionary approaches and applications". In: *IEEE Transactions on Evolutionary Computation* 22.2 (2018), pp. 276–295. DOI: 10.1109/TEVC.2017.2712906.

[105] A. Sommer. "Numerical methods for parameter estimation in dynamical systems with noise. with applications in systems biology". PhD thesis. Universität Heidelberg, 2016. DOI: 10.11588/heidok.00022589.

[106] Steinbeis-Forschungszentrum Optimierung, Steuerung und Regelung. *Users' Guide to WORHP 1.15*. 2022. URL: https://worhp.de/latest/download/user_manual.pdf (visited on 07/08/2023).

[107] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer New York, NY, 1980. DOI: 10.1007/978-1-4757-5592-3.

[108] A. S. Strekalovsky, A. V. Orlov, and A. V. Malyshev. "On computational search for optimistic solutions in bilevel problems". In: *Journal of Global Optimization* 48.1 (2010), pp. 159–172. DOI: 10.1007/s10898-009-9514-z.

[109] I. Swameye, T. G. Müller, J. Timmer, O. Sandra, and U. Klingmüller. "Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by databased modeling". In: *Proceedings of the National Academy of Sciences* 100.3 (2003), pp. 1028–1033. DOI: 10.1073/pnas.0237333100.

[110] C.-J. Thore. "Implicitly and explicitly constrained optimization problems for training of recurrent neural networks". In: *22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2014.

[111] A. Tin and A. B. Zemkoho. "Levenberg-Marquardt method and partial exact penalty parameter selection in bilevel optimization". In: *Optimization and Engineering* 24 (2022), pp. 1343–1385. DOI: 10.1007/s11081-022-09736-1.

[112] S.-M. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark. "AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 4860–4871.

[113] J. L. G. Velarde, J. F. C. Vallejo, and G. P. Serrano. "A Scatter Search Algorithm for Solving a Bilevel Optimization Model for Determining Highway Tolls". In: *Computación y Sistemas* 19 (2015). DOI: 10.13053/CyS-19-1-1916.

[114] L. Vicente, G. Savard, and J. Júdice. "Descent approaches for quadratic bilevel programming". In: *Journal of Optimization Theory and Applications* 81.02 (1994), pp. 379–399. DOI: 10.1007/BF02191670.

[115] P. Vicol, J. P. Lorraine, F. Pedregosa, D. Duvenaud, and R. B. Grosse. "On Implicit Bias in Overparameterized Bilevel Optimization". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 22234–22259.

[116] V. Visweswaran, C. A. Floudas, M. G. Ierapetritou, and E. N. Pistikopoulos. "A Decomposition-Based Global Optimization Approach for Solving Bilevel Linear and Quadratic Programs". In: *State of the Art in Global Optimization: Computational Methods and Applications*. Ed. by C. A. Floudas and P. M. Pardalos. Vol. 7. Boston, MA: Springer US, 1996, pp. 139–162.

[117] C. P. Vyasarayani, T. Uchida, and J. McPhee. "Single-shooting homotopy method for parameter identification in dynamical systems". In: *Physical Review E* 85 (3 2012). DOI: 10.1103/PhysRevE.85.036201.

[118] C. P. Vyasarayani, T. Uchida, A. Carvalho, and J. McPhee. "Parameter identification in dynamic systems using the homotopy optimization approach". In: *Multibody System Dynamics* 26.4 (2011), pp. 411–424. DOI: 10.1007/s11044-011-9260-0.

[119] C. P. Vyasarayani, T. Uchida, and J. McPhee. "Nonlinear Parameter Identification in Multibody Systems Using Homotopy Continuation". In: *Journal of Computational and Nonlinear Dynamics* 7 (2011). DOI: 10.1115/1.4004885.

[120] A. Wächter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[121] Y. Wang, W. Yin, and J. Zeng. "Global Convergence of ADMM in Nonconvex Nonsmooth Optimization". In: *Journal of Scientific Computing* 78.1 (2019), pp. 29–63. DOI: 10.1007/s10915-018-0757-z.

[122] L. T. Watson. "Theory of Globally Convergent Probability-One Homotopies for Nonlinear Programming". In: *SIAM Journal on Optimization* 11.3 (2001), pp. 761–780. DOI: 10.1137/S105262349936121X.

[123] L. T. Watson and R. T. Haftka. "Modern homotopy methods in optimization". In: *Computer Methods in Applied Mechanics and Engineering* 74.3 (1989), pp. 289–305. DOI: 10.1016/0045-7825(89)90053-4.

[124] H. Wernsing and C. Büskens. "Parameter Identification for Finite Element Based Models in Dry Machining Applications". In: *Procedia CIRP* 31 (2015). 15th CIRP Conference on Modelling of Machining Operations (15th CMMO), pp. 328–333. DOI: 10.1016/j.procir.2015.03.037.

[125] H. Wernsing. "PDE-restringierte Optimierung in Anwendungen der spanenden Trockenbearbeitung". PhD thesis. University of Bremen, 2018.

[126] P. Wolfe. "Convergence Conditions for Ascent Methods". In: *SIAM Review* 11.2 (1969), pp. 226–235. DOI: 10.1137/1011036.

[127] J. J. Ye and D. L. Zhu. "Optimality conditions for bilevel programming problems". In: *Optimization* 33.1 (1995), pp. 9–27. DOI: 10.1080/02331939508844060.

[128] J. J. Ye. "Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints". In: *Journal of Mathematical Analysis and Applications* 307.1 (2005), pp. 350–369. DOI: 10.1016/j.jmaa.2004.10.032.

[129] C. Zach and G. Bourmaud. "Descending, Lifting or Smoothing: Secrets of Robust Cost Optimization". In: *Computer Vision — ECCV 2018*. Ed. by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss. Springer International Publishing, 2018, pp. 558–574. DOI: 10.1007/978-3-030-01258-8_34.

[130] S. Zhou, A. B. Zemkoho, and A. Tin. "BOLIB: Bilevel Optimization LIBrary of Test Problems". In: *Bilevel Optimization*. Ed. by S. Dempe and A. B. Zemkoho. Vol. 161. Springer International Publishing, 2020, pp. 563–580. DOI: 10.1007/978-3-030-52119-6_19.

[131] C. Zimmer, F. T. Bergmann, and S. Sahle. *Reducing local minima in fitness landscapes of parameter estimation by using piecewise evaluation and state estimation*. 2016. DOI: 10.48550/arXiv.1601.04458.