I N A U G U R A L – D I S S E R T A T I O N

zur

Erlangung der Doktorwürde

der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

R u p r e c h t – K a r l s – U n i v e r s i t ä t

H e i d e l b e r g

vorgelegt von

Dipl.-Chem., B.Sc. (Math.) Marta Sauter

aus Zabrze, Polen

Tag der mündlichen Prüfung

.....

# Numerical Methods for Bilevel Optimal Control of Constrained Biomechanical Multbody Systems with Applications in Diagnosis of Cerebral Palsy

Betreuer

Prof. Dr. Dr. h. c. mult. Hans Georg Bock

Apl. Prof. Dr. rer. nat. Sebastian Wolf

## Zusammenfassung

Ziel dieser Arbeit ist die Entwicklung mathematischer Modelle und numerischer Methoden basierend auf Optimalsteuerung zur Erstellung eines Klassifizierungs- und Diagnoseschemas für den pathologischen Gang von Patienten mit Zerebralparese (CP). Der pathologische Gang von CP-Patienten ist noch immer nicht vollständig verstanden und Gegenstand aktueller Forschung. Methoden zur Diagnose oder Klassifizierung des Gangs von CP-Patienten sind essentiell für eine verlässliche Interventions- und Behandlungsplanung. Um dieses übergeordnete Ziel zu erreichen modellieren wir den Körper der CP-Patienten durch ein starres, biomechanisches Mehrkörpersystem. Die Dynamik dieses Mehrkörpersystems tritt auf als Beschränkung eines Optimalsteuerungsproblems (OCP), das wir zur Beschreibung des pathologischen Gangs aufsetzen. Dieses Vorgehen basiert auf der allgemeinen Annahme, dass Optimierung ein grundlegendes Prinzip der menschlichen Fortbewegung ist, so dass der Gang als optimal in Bezug auf eine Kombination verschiedener ausgewählter Optimierungskriterien interpretiert werden kann. Um ein kalibriertes personalisiertes Gangmodell zu erhalten, müssen unbekannte Modellparameter und Gewichte verschiedener Optimierungskriterien im OCP unter Berücksichtigung von gegebenen Bewegungsdaten aus dem HEIDELBERG MOTIONLAB identifiziert werden. Wir formulieren ein zweistufiges inverses OCP: Auf der oberen Ebene dieses zweistufigen Optimierungsproblems haben wir ein Parameterschätzproblem, das durch das parametrisierte OCP der unteren Ebene, das den Gang eines Patienten mit CP beschreibt, beschränkt wird. Bei gegebenen Bewegungserfassungsdaten können die Unbekannten so bestimmt werden, dass das entwickelte Gangmodell am besten zu den gegebenen Messungen passt.

Für die Lösung von zweistufigen inversen OCPsbetrachten wir einen direkten Lösungsansatz durch Anwendung der Mehrzielmethode und ersetzen das resultierende nichtlineare Programm (NLP) durch seine Optimalitätsbedingungen erster Ordnung. Daraus ergibt sich ein sogenanntes Mathematisches Programm mit Komplementaritätsbedingungen - eine anspruchsvolle Klasse von NLPs. Wir etablieren eine neue mathematische Methode unter Berücksichtigung bekannter spezifischer Eigenschaften der zugrundeliegenden Beschränkungen und schlagen effiziente numerische Algorithmen vor. In diesem sogenannten Direct Simultaneous Approach with Fixed Active Set (DISIMFAS) nutzen wir darüber hinaus gegebene Strukturen aus, die sich aus der Mehrzielmethode und dem zweistufigen Optimierungsproblem ergeben. In dieser Arbeit entwickeln wir das Softwarepaket PARDYNOPT mit einer effizienten Implementierung des vorgeschlagenen DISIMFAS. PARDYNOPT implementiert darüber hinaus numerische Methoden zur Lösung von OCPs im Allgemeinen, basierend auf der Mehrzielmethode und ist modular aufgebaut um zukünftige Erweiterungen und Untersuchungen zu ermöglichen.

Die Leistungsfähigkeit der entwickelten numerischen Methode wird anhand einer Reihe von unterschiedlichen Bilevel Inverse OCPs demonstriert, worunter wir auch ein neues Bilevel Inverse OCP für ein menschenähnliches Bewegungsmodell erarbeiten und analysieren, mit vielversprechenden Ergebnissen für eine zukünftige Anwendung der vorgeschlagenen Methode zur Identifizierung unbekannter Parameter im Gangmodell von Patienten mit CP. Als einen wesentlichen Schritt in diese Richtung entwickeln wir in dieser Arbeit ein personalisiertes Modell eines starren Mehrkörpersystems für einen Patienten mit CP, in einer Weise, dass dessen Dynamik die Hauptmerkmale des pathologischen Gangs erfassen kann. Wir schlagen ein OCP vor, das durch die Dynamik des starren Mehrkörpersystems beschränkt wird, mit einem Least-Squares Term als Optimierungskriterium, um die Rekonstruktion der zugrundeliegenden Dynamik des vorgeschlagenen CP-Modells für gegebene Bewegungserfassungsdaten zu untersuchen. Darüber hinaus leiten wir ein geeignetes parametrisiertes OCP mit einer ausgewählten Kombination von Optimierungskriterien für den Gang eines CP-Patienten ab. Dieses Gangmodell kann als untere Ebene in der zweistufigen inversen OCP-Formulierung dienen. Die Eignung des CP-Gangmodells wird durch die Analyse von Lösungen unterschiedlich gewichteter OCPs bewertet.

## Abstract

This thesis aims at developing mathematical models and numerical methods for establishing a classification and diagnosis scheme for the pathological gait of Cerebral Palsy (CP) patients based on Optimal Control (OC). The pathological gait of CP patients is still an ongoing field of research and not completely understood. Methods to diagnose or classify the gait of CP patients are important for intervention and treatment planning. For this purpose we model the patient's body by a biomechanical rigid multibody system. The dynamics of this multibody system appear as constraints in an Optimal Control Problem (OCP), which we use to describe the pathological gait. This is based on the common assumption that optimization serves as a fundamental principle of human locomotion, such that the gait can be interpreted as optimal with respect to a well-chosen combination of varying optimization criteria. To achieve a calibrated patient-specific gait model, unknown model parameters and objective weights of various optimization criteria in the OCP have to be identified under consideration of given motion capture data from the HEIDELBERG MOTIONLAB. We formulate a Bilevel Inverse OCP: on the upper level of this bilevel optimization problem we have a Parameter Estimation (PE) Problem, constrained by the lower level parametrized OCP describing the gait of a patient with CP. With given motion capture data the unknowns can be determined, such that the developed gait model fits the given measurements best.

For solving Bilevel Inverse OCPs we consider a direct solution approach by applying the Direct Multiple Shooting Method and replace the resulting lower level Nonlinear Programming Problem (NLP) by its first-order optimality conditions. This results in a so-called Mathematical Program with Complementarity Constraints (MPCC) - a challenging class of NLPs. We establish a novel mathematical method under consideration of known specific characteristics of the underlying constraints, and propose efficient numerical algorithms. In this so-called Direct Simultaneous Approach with Fixed Active Set (DISIMFAS), we furthermore exploit given structures, which arise as a result of the Direct Multiple Shooting Method and the bilevel optimization problem. In this thesis, we develop the software package PARDYNOPT with an efficient implementation of the proposed DISIMFAS. PARDYNOPT furthermore implements numerical methods for solving OCPs in general based on the Direct Multiple Shooting Method and is designed in a modular way for further extensions and investigations.

The performance of the developed numerical method is demonstrated on a set of different Bilevel Inverse OCPs. Among these, we derive and analyse a new Bilevel Inverse OCP for a human-like locomotion model with promising results for future application of the proposed method to identify unknown parameters in the gait model of patients with CP. As an essential step in this direction, in this thesis we develop a rigid multibody system model for a patient with CP, such that its dynamics can capture the main characteristics of the pathological gait. We propose an OCP constrained by the rigid multibody system dynamics with a least-squares objective to investigate the reconstruction of the underlying dynamics of the proposed CP model to given motion capture data. Furthermore, we derive an adequate parametrized OCP with a well-chosen combination of optimization criteria for the gait of a CP patient. This gait model can serve as a lower level in the Bilevel Inverse OCP formulation. The suitability of the CP gait model is evaluated by analyzing solutions of differently weighted OCPs.

## Acknowledgements

Without the support and help of others, such a thesis would hardly be possible. Therefore, I want to say thank you to all my supporters and appreciate their valuable contribution, which enabled me to grow in so many ways.

Thanks to:

My supervisors and mentors Ekaterina Kostina and Hans Georg Bock, for giving me the opportunity to perform this thesis on such an exciting topic and especially for the constant support and supervision, valuable discussions and the always warm atmosphere.

Johannes Schlöder for his supervision and support and all the interesting discussions and helpful input.

Sebastian Wolf from the HEIDELBERG MOTIONLAB for the great cooperation, supervision and his introduction into cerebral palsy and gait analysis. Also for the possibility to work with the gait analysis data, which enabled my work. Therefore, I would also like to thank everyone involved in deriving this data.

Katja Mombaur and the members of her group Optimization, Robotics & Biomechanics for the numerous fruitful meetings and discussions, which have always been a source of inspiration and motivation. Special thanks to Monika Harant for her help with the CP model.

All members of *SimOpt* and *NumOpt*, for the friendly atmosphere and many coffees, support with all kinds of problems and helpful discussions. Special thanks to Andreas Sommer for his great support, as well as valuable input to ParDynOpt and for proofreading this thesis. Also thanks to Andreas Meyer and Leo Wirsching for their support with ParDynOpt.

The doctoral office and Anastasia Valter, Jeannette Walsch, and Herta Fitzer for their help with administrative challenges.

My parents for their help in any situation and their constant and unconditional support throughout my life. Without them all of this would have not been possible.

My brother and family for being there and always making me smile.

And, finally, my family, Hanna, Felix, and Max for always believing in me and their unconditional love. I am so grateful that you were with me all these years.

# Contents

# Introduction

## Motivation and Overarching Goal

Cerebral Palsy (CP) is not a disease in the ordinary sense, but a collection of early childhood movement disorders with a prevalence of approximately 0.2% of live births [48]. These movement disorders are caused by abnormal development or damage in parts of the brain that control motor function. It is associated with impaired motor function, coordination, and abnormal muscle tone. Symptoms range from barely noticeable to severe and limiting movement disorders. These disorders can affect one or more limbs and may even lead to paralysis and joints that become so stiff that they can no longer be moved. Fortunately, although CP is not curable, there exist a variety of measures that can be taken to promote the patient's mobility and independence. One way is to use clinical gait analysis for therapy decision making by the physicians. This gait analysis provides kinematic information on joint angle positions during walking and the corresponding kinetic data, and gives further insight after electromyographic examination in activities of selected muscles. Although an objective and consistent diagnosis of CP gait disorders is necessary for the determination and verification of therapy strategies, it usually requires a high degree of clinical experience and, hence, strongly depends on the treating physicians. Our overarching goal is to develop a mathematical and numerical framework for inverse problems enabling a consistent diagnosis and support medical doctors with an automated diagnostic tool. As an important step in this direction we transfer investigated medical challenges to adequate mathematical tasks, and derive inverse problems to support physicians in diagnosis under consideration of provided motion capture data from the HEIDELBERG MOTIONLAB [173]. Consequently, for the purpose of classifying CP gaits and identifying patient-specific parameters, we are faced with highly non-smooth bilevel optimization problems, which we tackle on three different levels.

*Level 1:* To describe CP gaits mathematically, appropriate and personalized models of the patients' bodies have to be developed covering the main pathological physiology. This can be achieved by detailed rigid multibody system models, for which the underlying phasewise defined dynamics can be calculated by solving the resulting equations of motion.

*Level 2:* It is a common assumption that the human gait is a result of a person's decision guided by certain optimization criteria and constraints [10, 11]. This means that optimization serves as a guiding principle of bipedal locomotion of humans [129]. As a consequence human locomotion can be mathematically formulated as an Optimal Control Problem (OCP) with the underlying dynamics as described at level 1 and torques and forces as controls. An adequate Optimal Control (OC) model for the gait of a CP patient has to capture the main characteristics of the pathological gait with a well-chosen combination of various optimization criteria (such as stability, energy efficiency, or reduction of executed mechanical work), state and control inequality constraints, and given physiological parameters.

*Level 3:* For the purpose of classifying CP gaits and identifying patient-specific parameters, such as joint displacements and skeleton deformations, as well as a (parametrized) optimization criterion supporting medical diagnosis with the derived OCP of a constrained biomechanical multibody system from level 2, we formulate a bilevel optimization problem under consideration of provided motion capture data from the HEIDELBERG MOTIONLAB [173]. This results in a so-called *Bilevel Inverse OCP*: a parameter estimation problem constrained by the OCP derived on level 2 with the underlying dynamics from level 1. The essential role of this mathematical model as a so-called *digital twin* is to provide the physicians with a non-invasive diagnosis tool. However, this overarching goal can only be achieved by developing reliable mathematical methods to solve these kind of bilevel optimization problem, which is one key aspect in this thesis.

As basis for our work, we use the foundations laid by Hatz in [80].

## Scope of this Thesis

This work is part of the project "Numerical Methods for Diagnosis and Therapy Design of Cerebral Palsy by Bilevel Optimal Control of Constrained Biomechanical Multi-Body Systems" [27], guided by Prof. Dr. Ekaterina A. Kostina and Prof. Dr. Dr. h. c. mult. Hans Georg Bock, which emerged from the long-standing collaboration with Apl. Prof. Dr. rer. nat. Sebastian Wolf and the HEIDELBERG MOTIONLAB [173] of the Department of Orthopedics and Trauma Surgery of the Heidelberg University Hospital. It aims a detailed investigation of involved medical challenges, which arise from the morbid conditions of patients with CP with resulting pathological gaits. In close cooperation with Sebastian Wolf from the HEIDELBERG MOTIONLAB the key challenges could be identified and helpful inside in the medical background could be accomplished. This enables us to transfer these challenges and insights to adequate mathematical tasks. Within the overall project, our ultimate goal is "[...] to develop a reliable mathematical and numerical framework for

- inverse problems to support proper diagnosis and
- parameter optimization and OCP to improve the planning of interventions" [27],

and to provide a routinely applicable tool for the physicians. In this work we focus on diagnosis of CP gait, whereas in the side project [151] therapy design of CP is the main topic.

To meet the appearing challenges, especially in the field of biomechanical modeling of the locomotor system, and optimization and simulation of human gaits, our work was supported by close collaboration with Prof. Dr. Katja Mombaur - now at University Waterloo as holder of the "Canada Excellence Research Chair for Human-Centred Robotics & Machine Intelligence", and her former working group "Optimization in Robotics and Biomechanics" at Heidelberg University.

For a routinely applicable mathematical tool for the physicians and a sustainable software development, which enables future extentions by connecting other software packages through suitable interfaces, we cooperated with the "Scientific Computing Sustainable Software Collaboratory" under the supervision of Ekaterina Kostina at the Interdisciplinary Center for Scientific Computing.

## Contributions

### A Mathematical Method for Solving Bilevel Inverse Optimal Control Problems

To meet our ultimate goal, which is the identification of optimization criteria and model parameters in an OCP describing the gait of a patient with CP, we derive a direct simultaneous method for solving Bilevel Inverse OCPs. These kinds of bilevel problems comprise a Parameter Estimation (PE) Problem on the upper level and a parametrized OCP on the lower level, where parameters are determined, such that given measurements are met. In our direct simultaneous solution approach - the so-called *Direct Simultaneous Approach with Fixed Active Set (*DISIMFAS*)* - we start with a parametrization of the dynamics by applying the Direct Multiple Shooting Method [26] and an appropriate discretization of the controls, constraints and objective function. Before we reformulate the bilevel problem as a one-level problem, only those discretized inequality constraints are included and fixed that correspond to the *optimal active set* under consideration of given structural information on the underlying dynamic system. The lower level of the resulting bilevel Nonlinear Programming Problem (NLP) is then replaced by its first-order optimality conditions, which leads to a structured one-level NLP, which is solved by tailored numerical solution methods developed in this thesis. Generally, the optimal active set of the lower level cannot easily be determined under consideration of given measurements, therefore, the proposed method can be embedded in a sequential algorithm for the identification of the optimal active set.

#### Structure Exploitation in Numerical Algorithms for Bilevel Inverse Optimal Control Problems

A desired feature of the DISIMFAS developed in this thesis is its routine applicability by physicians for the identification of unknowns in a patient-specific CP gait model. To meet this desired feature and for reliability, we establish an efficient and structure exploiting algorithm to solve Bilevel Inverse OCPs by applying the DISIMFAS and solving the resulting highly structured NLP. We exploit given structures for several parts in the algorithm, which have to be provided in standard NLP solvers for an efficient calculation of essential quantities. These exploitations cover needed quantities by interior-point methods, Sequential Quadratic Programming (SQP) methods, and the Generalized Gauß-Newton Method.

#### Efficient Implementation of the Proposed Mathematical Method

In this work we developed a modular C++ software package PARDYNOPT within the Scientific Computing Sustainable Software Collaboratory under the supervision of Ekaterina Kostina at the Interdisciplinary Center for Scientific Computing. It was constructed in such a way that its modules can easily be exchanged and extended in the future for a sustainable implementation. In addition to general OCPs, our software package allows us to solve Bilevel Inverse OCPs efficiently. It realizes the proposed DISIMFAS and provides interfaces to the software package IPOPT [169] for the solution of the resulting structured one-level NLP with an efficient interior-point method. Furthermore, interfaces for SNOPT [71] and FILTERSQP [59] are prepared for implementations of SQP methods. For setting up a dynamic model in a comfortable way, an interface to the software package SOLVIND [5] is integrated. Furthermore, SOLVIND provides Ordinary Differential Equation (ODE) and Differential Algebraic Equation (DAE) solvers in an Internal Numerical Differentiation (IND) framework [23, 24] for exact derivative generation using Automatic Differentiation (AD) with an interface to the software package ADOL-C [170]. This powerful feature of the software suite SOLVIND is also supported by PARDYNOPT.

#### Numerical Investigations of the Proposed Method in two Case Studies

Our derived solution approach for Bilevel Inverse OCPs in the implementation PARDYNOPT is investigated numerically in two case studies apart from human locomotion with different focuses. In the first case study, we consider the well-known structure of the resulting control functions in a rocket car example, which was, e.g., investigated in [81]. In our analysis, distinct multi-stage problem formulations lead to successful applications of our DISIMFAS in a Bilevel Inverse OCP setting with simulated measurements. In a second case study, we consider a polar robot example as derived by Steinbach [164] and discuss the performance of our solution approach under consideration of given structural information and compare it to results of Hatz [80] with the Direct All-at-Once Approach.

#### Bilevel Inverse Optimal Control Model for a Basic Human-Like Locomotion with Numerical Investigations of the Proposed Solution Approach

For numerical investigations of our DISIMFAS in the context of human locomotion, we derive a Bilevel Inverse OCP for the identification of unknown model parameters and objective weights in a basic walker gait model. This model, which we developed during the project, serves as a basic model for human locomotion. We provide a full description of the corresponding problem formulation. Furthermore, the performance of our proposed method in its implementation in PARDYNOPT is discussed in detail.

#### Parametrized Optimal Control Model for the Gait of a Patient with Cerebral Palsy and its Analysis for Deployment in a Bilevel Inverse OCP Set-Up

To meet our ultimate goal to support physicians with a diagnosis tool for evaluation of possible interventions on the locomotor system and therapy management of CP patients, we develop a CP gait model with 20 Degrees of Freedom (DOFs) based on the work of Hatz [80] under consideration of given measurements provided by

the HEIDELBERG MOTIONLAB [173]. With patient-specific knee axes, an integration of passive reset forces as restriction on the range of motion in the knees of CP patients, and a suitable combination of optimization criteria we are able to capture the typical pathological gait with an adequate incorporation of complexity into our gait model. This is demonstrated by numerical investigations, where a reconstruction of the derived dynamics of our newly developed rigid mulitbody system model of a CP patient is performed. Furthermore, with our derived multi-stage OCP formulation for the gait of a CP patient, varying gaits for differently weighted objective functions could be synthesized, which motivates us to use our OCP formulation in a Bilevel Inverse OCP set-up for the identification of unknowns for an appropriate calibration of the patient-specific gait model.

#### Supervised Learning Approach Based on Deep Neural Networks for the Identification of Optimal Weight in our CP Gait Model

As a pilot study, we develop a supervised learning approach, which is based on the application of Deep Neural Networks (DNNs). In a first *online phase* the DNN is trained on simulated data by solving the derived parametrized OCPs describing the gait of a patient with CP for varying objective weights. The appropriately trained DNN is then used in an *offline phase* for the estimation of optimal weights by propagation of newly obtained measurements through the DNN. The proposed supervised learning approach is investigated in a case study on simulated data for the developed CP gait model, where the objective weights could be identified reasonably well.

## Thesis Overview

This thesis comprises three main parts: *Foundations*, *Contributions*, and *Implementations and Numerical Results*. The first part presents foundations that the contributions of this thesis are build on. In the second part the contributions of this thesis, which are investigated numerically in the third part, are described in detail.

#### Part 1: Foundations

In chapter 1 we start with basic definitions for NLPs and introduce the concepts of selected numerical methods for solving these kinds of problems. We discuss a challenging class of NLPs: Mathematical Programs with Complementarity Constraints (MPCCs), as they appear in direct simultaneous solution approaches for Bilevel Inverse OCPs. In chapter 2 an overview on optimization of dynamic systems is given. It introduces a general problem formulation for OCPs and describes the special class of multi-stage OCPs with discontinuities. Selected numerical methods for these kinds of problems are presented. Furthermore, PE problems and a solution approach are discussed. As a main topic of this thesis, we state the general problem formulation for Bilevel Inverse OCPs used throughout this thesis and give a brief overview on selected solution approaches. In more detail we discuss the Direct All-at-Once Approach [80], as a basis for our DISIMFAS. In chapter 3 we introduce how human locomotion can be described using OC and give some background on the disease CP and the characteristic pathological gait.

#### Part 2: Contributions

The main contributions of this thesis in the field of mathematics and orthopedic biomechanics, as already introduced above, are presented in detail in the second part of this thesis. Therein, we start with the developed mathematical method for Bilevel Inverse OCPs and our motivation behind this approach. In section 4.2 we discuss each step of the numerical solution approach in detail and provide tailored numerical methods with structure exploitation to solve the resulting NLP. In section 4.4 we give an outlook on the embedding of the DISIMFAS in a sequential algorithm for the identification of the active set. In chapter 5, a Bilevel Inverse OCP formulation for the identification of unknowns in a basic model for human-like locomotion is developed. Therein, we start with the underlying dynamics of the rigid multibody system model of a basic walker model

[150], followed by an appropriate multi-stage OCP formulation describing its gait. This derived OCP serves as the lower level in the Bilevel Inverse OCP of a basic walker gait model. In the final chapter of the second part of this thesis the developed patient-specific CP gait model is presented. The derived rigid multibody system model as well as the corresponding underlying dynamics are discussed in detail. Furthermore, a least-squares multi-stage OCP formulation for the reconstruction of the dynamics to given motion capture data from the HEIDELBERG MOTIONLAB [173] is provided. In section 6.4 a multi-stage OCP formulation for the gait of a patient with CP is discussed with a combination of well-chosen optimization criteria. After a detailed comparison to the CP gait model by Hatz and other existing models, we provide a Bilevel Inverse OCP formulation for the identification of unknowns in the CP gait model as an outlook for further research. Part 2 is concluded with the development of a supervised learning approach for the identification of optimal weights in the derived CP gait model via DNNs.

**Part 3: Implementations and Numerical Results**

In the final part of this thesis we start with implementation details on the newly developed software package PARDYNOPT with the general software structure and the framework for setting up OCPs and Bilevel Inverse OCPs. The performance of the developed numerical method DISIMFAS is investigated in chapter 8 in two case studies, covering rocket car and polar robot examples, as, e.g., introduced in [81, 80, 164], and in chapter 9 for the Bilevel Inverse OCP for a basic walker gait model derived in chapter 5. Finally, in chapter 10 we analyze our derived CP gait model from chapter 6 numerically, and additionally perform a case study of the supervised learning approach for identification of optimal weights via DNNs, followed by a conclusion and outlook.

**Computational Environment** The numerical results presented in chapter 8, chapter 9 and chapter 10 have been achieved on an Ubuntu 20.04 LTS system powered by an Intel Core i7-8550U CPU with 15.5 GiB main memory available.

# Part I

# Foundations

# Chapter 1

# Nonlinear Programming

In this chapter we review some basic theory on nonlinear programming used in this thesis. We start with basic definitions, followed by a selection of numerical methods for nonlinear programming: interior-point methods, SQP, and the Generalized Gauß-Newton method. The presentations in the first and second sections mainly follow the book of Nocedal and Wright [132]. For the Generalized Gauß-Newton method the content is based on the thesis of Sommer [161] with references therein. The final section is dedicated to a challenging class of NLPs: MPCCs, and provides only a brief overview following Fletcher et al. [61]. The interested reader is referred to the references given in the last section.

## 1.1 Basic Definitions

Throughout this section we follow the book of Nocedal and Wright [132] for selected material on the theory of constrained nonlinear optimization and start with the definition for NLPs.

**Definition 1.1 (Nonlinear Programming Problem)**
A *Nonlinear Programming Problem* is a constrained nonlinear optimization problem of the form

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{1.1a}$$

$$\text{s.t.} \quad 0 = c_i(x), \quad i \in \mathcal{E}, \tag{1.1b}$$

$$0 \leq c_i(x), \quad i \in \mathcal{I}, \tag{1.1c}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is called the objective function and $c : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{E}\cup\mathcal{I}|}$ represent the equality constraints for components $c_i(x), i \in \mathcal{E}$ and the inequality constraints for $c_i(x), i \in \mathcal{I}$. All functions are supposed to be sufficiently smooth, real-valued, and $\mathcal{I}$ and $\mathcal{E}$ are two disjoint finite sets of indices. The cardinality of both sets is denoted by $|\mathcal{E} \cup \mathcal{I}|$. △

**Definition 1.2 (Feasible Set)**
The *feasible set* $\Omega$ is defined as

$$\Omega := \{x | c_i(x) = 0, i \in \mathcal{E}, c_i(x) \geq 0, i \in \mathcal{I}\}. \qquad \triangle$$

**Definition 1.3 (Active Set)**
Let $x \in \Omega$ be feasible. Then the *active set* $\mathcal{A}(x)$ at point $x$ consists of all indices $i \in \mathcal{E}$ of equality constraints together with the indices of inequality constraints $i$ for which $c_i(x) = 0$, such that

$$\mathcal{A}(x) := \mathcal{E} \cup \{i \in \mathcal{I} | c_i(x) = 0\}.$$

Furthermore, we define the active set which belongs to inequality constraints as

$$\mathcal{A}^{\text{ic}}(x) := \mathcal{A}(x) \cap \mathcal{I} = \{i \in \mathcal{I} | c_i(x) = 0\}. \tag{1.2}$$

△

The inequality constraint $i \in \mathcal{I}$ is called *active* at a feasible point $x$, if $c_i(x) = 0$, and *inactive* if the strict inequality $c_i(x) > 0$ is satisfied.

**Definition 1.4 (Local and Global Solution)**

- A feasible point $x^* \in \Omega$ is a *local solution* of problem (1.1) if there is a neighborhood $\mathcal{N}$ of $x^*$ such that

  $$f(x) \geq f(x^*), \forall x \in \mathcal{N} \cap \Omega.$$

  It is called a *strict local solution* if the inequality is strictly fulfilled.

- A feasible point $x^* \in \Omega$ is a *global solution* of problem (1.1) if

  $$f(x) \geq f(x^*), \forall x \in \Omega.$$

  It is called a *strict global solution* if the inequality is strictly fulfilled. △

**Definition 1.5 (Lagrangian Function)**
The *Lagrangian function* of the general problem (1.1) is $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{I}|} \to \mathbb{R}$ and defined by

$$\mathcal{L}(x, \lambda, \mu) := f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) - \sum_{i \in \mathcal{I}} \mu_i c_i(x),$$

with *Lagrange multipliers* $\lambda \in \mathbb{R}^{|\mathcal{E}|}$ and $\mu \in \mathbb{R}^{|\mathcal{I}|}$. The cardinality of each set is denoted by $|\cdot|$. △

**Definition 1.6 (Linear Independence Constraint Qualification (LICQ),[83])**
Given the feasible point $x \in \Omega$ and the active set from definition 1.3, the *Linear Independence Constraint Qualification* holds at $x$ if the set of active constraint gradients

$$\{\nabla c_i(x), i \in \mathcal{A}(x)\},$$

is linearly independent. △

A generalization of LICQ is the Mangasarian-Fromowitz Constraint Qualification (MFCQ), which is a weaker condition than the LICQ and defined in the following.

**Definition 1.7 (Mangasarian-Fromowitz Constraint Qualification,[120])**
Given the feasible point $x \in \Omega$ and the active set from definition 1.3, the *Mangasarian-Fromowitz Constraint Qualification* holds at $x$ if the following conditions are satisfied:

- The set of equality constraint gradients

  $$\{\nabla c_i(x), i \in \mathcal{E}\},$$

  is linearly independent.

- There exists a vector $w \in \mathbb{R}^n$ such that

  $$\nabla c_i(x)^T w > 0, \forall i \in \mathcal{A}(x) \cap \mathcal{I} \quad \text{and} \quad \nabla c_i(x)^T w = 0, \forall i \in \mathcal{E}.$$ △

### 1.1.1 First-Order Necessary Optimality Conditions

In the following we state the first-order necessary optimality conditions which are also known as Karush-Kuhn-Tucker (KKT) conditions.

**Theorem 1.8 (Karush-Kuhn-Tucker Conditions, [99, 107])**
Suppose $x^*$ is a local solution of problem (1.1). Furthermore, let functions $f$ and $c_i$ be continuously differentiable, and the LICQ or MFCQ be fulfilled at $x^*$. Then there exist Lagrange multipliers $\lambda^* \in \mathbb{R}^{|\mathcal{E}|}$ and $\mu^* \in \mathbb{R}^{|\mathcal{I}|}$

such that

$$0 = \nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*), \qquad \text{(Stationarity)} \tag{1.3a}$$
$$0 = c_i(x^*), \qquad \forall i \in \mathcal{E}, \quad \text{(Primal Feasibility)} \tag{1.3b}$$
$$0 \leq c_i(x^*), \qquad \forall i \in \mathcal{I}, \quad \text{(Primal Feasibility)} \tag{1.3c}$$
$$0 \leq \mu^*, \qquad \text{(Dual Feasibility)} \tag{1.3d}$$
$$0 = \mu_i^* c_i(x^*), \qquad \forall i \in \mathcal{I}. \quad \text{(Complementarity)} \tag{1.3e}$$

For a given solution $x^*$, the Lagrangian multipliers are unique, if the LICQ holds. △

**Proof** See Nocedal and Wright [132], chapter 12.3. □

The complementarity conditions (1.3e) imply that inequality constraint $c_i(x^*)$ is active or that the corresponding Lagrange multiplier $\mu_i^* = 0$ is zero, or both. Together with the definition of the active set $\mathcal{A}^{\text{ic}}$ in (1.2), that belongs to inequality constraints the stationarity condition (1.3a) can be rewritten as

$$0 = \nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = \nabla f(x^*) - \sum_{i \in \mathcal{E}} \lambda_i^* \nabla c_i(x^*) - \sum_{i \in \mathcal{A}^{\text{ic}}(x^*)} \mu_i^* \nabla c_i(x^*).$$

## 1.2 Numerical Methods for Nonlinear Programming

In this section we give a brief overview on selected numerical methods for nonlinear programming and start with interior-point methods, followed by SQP and the Generalized Gauß-Newton Method. Numerical methods for nonlinear programming are addressed by many textbooks, e.g., Fletcher [58], Geiger and Kanzow [67], Gill et al. [72], and Nocedal and Wright [132]. Here we summarize some essential characteristics of each method following the latter reference and refer the interested reader to the corresponding chapters in the book by Nocedal and Wright [132, Chapters 10,18,19].

### 1.2.1 Interior-Point Methods

Interior-point methods - also known as barrier methods - for nonlinear optimization were already developed in the 1950s, but gained more interest by the late 1990s. An efficient implementation of an interior-point method in the software package IPOPT [169] is also integrated as NLP solver in PARDYNOPT developed in this work.

We start by considering NLP problem (1.1) of definition 1.1 and reformulate it as

$$\min_{x,s} \quad f(x) \tag{1.4a}$$
$$\text{s.t.} \quad 0 = c_i(x), \qquad i \in \mathcal{E}, \tag{1.4b}$$
$$0 = c_i(x) - s_i, \quad i \in \mathcal{I}, \tag{1.4c}$$
$$0 \leq s_i, \qquad i \in \mathcal{I}, \tag{1.4d}$$

by the introduction of a slack variables $s \in \mathbb{R}^{|\mathcal{I}|}$, such that the inequality constraints can be written as equalities. Let $\mathcal{L}(x^*, s^*, \lambda^*, \mu^*)$ denote the Lagrangian of the reformulated NLP (1.4). Then the KKT conditions (1.3) are

$$0 = \nabla_{x,s} \mathcal{L}(x^*, s^*, \lambda^*, \mu^*) \tag{1.5a}$$
$$0 = c_i(x), \qquad i \in \mathcal{E}, \tag{1.5b}$$
$$0 = c_i(x) - s, \qquad i \in \mathcal{I}, \tag{1.5c}$$
$$0 \leq s_i, \qquad i \in \mathcal{I}, \tag{1.5d}$$
$$0 \leq \mu_i, \qquad i \in \mathcal{I}, \tag{1.5e}$$
$$\nu = s_i \mu_i, \qquad i \in \mathcal{I}, \tag{1.5f}$$

if parameter $\nu = 0, \nu \in \mathbb{R}$. Equations (1.5f) with $\nu = 0$ and (1.5d), (1.5e) introduce a combinatorial aspect into solving the equation system (1.5) in determining the optimal active set. This difficulty can be circumvented by enforcing $\nu$ to be strictly positive. In the homotopy approach, a sequence of so-called *perturbed KKT conditions* of the form (1.5) are solved, where the positive parameters $\nu^k$ are iteratively driven to zero, while $s, \mu > 0$ and, hence, remain in the *interior*.

### 1.2.2 Sequential Quadratic Programming

SQP methods are frequently used for non-linearly constrained optimization, and are appropriate for small and large problem sizes. The basic concept of SQP methods is to solve a sequence of quadratic subproblems. We start with an equality constrained NLP of the following form

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{1.6a}$$

$$\text{s.t.} \quad 0 = c(x), \tag{1.6b}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{E}|}$ are smooth functions. In the SQP approach at interate $x^k$ NLP (1.6) is modeled by a quadratic subproblem. This subproblem is then minimized to produce a new iterate to solve the original problem.

The KKT conditions (1.3) for the equality-constrained NLP (1.6) can be summarized in

$$F(x, \lambda) = \begin{pmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ c(x) \end{pmatrix} = 0, \tag{1.7}$$

where $\lambda \in \mathbb{R}^{|\mathcal{E}|}$ are the Lagrange multipliers related to the equality constraints and

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) - \nabla c(x)^T \lambda,$$

the gradient of the Lagrange function with respect to $x$. $\nabla c(x)$ denotes the constraint Jacobian, where the rows are the gradients $\nabla c_i(x), i \in \mathcal{E}$. The equation system (1.7) can now be solved by applying Newton's method, see e.g. [132, Chapter 11]. Therefore, we define the Jacobian of (1.7) as

$$J(x, \lambda) = \begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x, \lambda) & -\nabla c(x)^T \\ \nabla c(x) & 0 \end{pmatrix}, \tag{1.8}$$

and obtain an update of the current iterate $(x^k, \lambda^k)$

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ \lambda^k \end{pmatrix} + \begin{pmatrix} p^k \\ p^\lambda \end{pmatrix}, \tag{1.9}$$

where $p^k$ and $p^\lambda$ solve the following Newton-KKT system

$$\begin{pmatrix} \nabla^2_{xx} \mathcal{L}(x^k, \lambda^k) & -\nabla c(x^k)^T \\ \nabla c(x^k) & 0 \end{pmatrix} \begin{pmatrix} p^k \\ p^\lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x^k) + \nabla c(x^k)^T \lambda^k \\ -c(x^k) \end{pmatrix}. \tag{1.10}$$

If matrix (1.8) is regular, the solution of the equation system in (1.10) is well-defined. This is the case for the KKT matrix, if the following assumption holds at $(x, \lambda) = (x^k, \lambda^k)$.

**Assumption 1.1**

- The constraint Jacobian $\nabla c(x)$ has full rank,

- The matrix $\nabla^2_{xx}\mathcal{L}(x,\lambda)$ is positive definite on the null space of the constraint gradients, that is

$$d^T\nabla^2_{xx}\mathcal{L}(x,\lambda)d > 0 \forall d \neq 0 \quad \text{such that} \quad \nabla c(x)d = 0. \qquad \triangle$$

The steps $p^k$ and $p^\lambda$ determined by (1.9) and (1.10) can also be interpreted as the solution of a Quadratic Program (QP) of the following form:

$$\min_p \quad \frac{1}{2}p^T\nabla^2_{xx}\mathcal{L}(x^k,\lambda^k)p + \nabla f(x^k)^T p \tag{1.11a}$$

$$\text{s.t.} \quad 0 = \nabla c(x^k)p + c(x^k). \tag{1.11b}$$

If the assumption 1.1 holds, QP (1.11) has a unique solution $(p^k, v^k)$, which satisfies the following KKT conditions:

$$0 = \nabla^2_{xx}\mathcal{L}(x^k,v^k)p^k + \nabla f(x^k) - \nabla c(x^k)^T v^k \tag{1.12}$$

$$0 = \nabla c(x^k)p^k + c(x^k), \tag{1.13}$$

where $v^k \in \mathbb{R}^{|\mathcal{E}|}$ define the Lagrange multipliers. We have that $v^k = \lambda^k + p^\lambda = \lambda^{k+1}$, if we subtract $\nabla c(x^k)^T\lambda^k$ from both sides of the first equation in the Newton-KKT system (1.10). Hence, $(p^k, v^k)$ can be identified with the solution of (1.10). Both interpretations - the new iterate $(x^{k+1}, \lambda^{k+1})$ as the solution of QP (1.11) or as an iterate generated by Newton's method in (1.9) and (1.10) - have their justification. The first interpretation is often used in the analysis of convergence properties, whereas the latter interpretation serves as a starting point for the derivation of practical algorithms. In the following, we state a local SQP algorithm in a basic form.

**Algorithm 1** Scheme of the Local SQP Method

1: Choose an initial pair $(x^0, \lambda^0)$, $k \leftarrow 0$
2: **while** a convergence test is not satisfied at $(x^k, \lambda^k)$ **do**
3: Evaluate $f(x^k), \nabla f(x^k), \nabla^2_{xx}\mathcal{L}(x^k,\lambda^k), c(x^k)$ and $\nabla c(x^k)$
4: Compute solution $(p^k, \lambda^{k+1})$ of QP (1.11)
5: $x^{k+1} \leftarrow x^k + p^k$
6: $k \leftarrow k+1$
7: **end while**

Algorithm 1 can be extended for solving the general NLP (1.1) with inequality constraints from definition 1.1 by linearization of both the equality and inequality constraints to obtain the following QP

$$\min_p \quad \frac{1}{2}p^T\nabla^2_{xx}\mathcal{L}(x^k,\lambda^k)p + \nabla f(x^k)^T p \tag{1.14a}$$

$$\text{s.t.} \quad 0 = \nabla c_i(x^k)p + c_i(x^k), \quad i \in \mathcal{E}, \tag{1.14b}$$

$$0 \leq \nabla c_i(x^k)p + c_i(x^k), \quad i \in \mathcal{I}. \tag{1.14c}$$

Replacing QP (1.11) by QP (1.14) in algorithm 1 leads to a local SQP method for the general case. The solution of the inequality constrained QP can be achieved by applying algorithms for quadratic programming as, e.g. the ones described in the textbook of Nocedal and Wright [132, Chapter 16].

### 1.2.3 The Generalized Gauß-Newton Method

The Generalized Gauß-Newton method was introduced by Bock [25] for solving equality and inequality constrained nonlinear least-squares problems, which generalizes the classical Gauß-Newton method for unconstrained nonlinear least-squares problems as described, e.g., in [132]. The presentation of this subsection mainly follows the theses of Bock [25], Hatz [80], and Sommer [161], and the textbook of Nocedal and

Wright [132]. We start with an equality-constrained nonlinear least-squares problem of the following form

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|r(x)\|_2^2 \tag{1.15a}$$

$$\text{s.t.} \quad 0 = c(x), \tag{1.15b}$$

where the residual function $r:\mathbb{R}^n \to \mathbb{R}^{n_{lsq}}$ and the equality constraint function $c:\mathbb{R}^n \to \mathbb{R}^{|\mathcal{E}|}$ are smooth functions. The KKT conditions (1.3) for NLP (1.15) are given by

$$\begin{pmatrix} \nabla_x \mathcal{L}(x,\lambda) \\ c(x) \end{pmatrix} = 0, \tag{1.16}$$

where $\lambda \in \mathbb{R}^{|\mathcal{E}|}$ are the Lagrange multipliers related to the equality constraints and

$$\nabla_x \mathcal{L}(x,\lambda) = \nabla r(x)^T r(x) - \nabla c(x)^T \lambda, \tag{1.17}$$

the gradient of the Lagrange function with respect to $x$. Similar to the above case for the derivation of the SQP method, the equation system (1.16) can now be solved by applying Newton's method. Therein, we obtain an update of the current iterate $(x^k, \lambda^k)$

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ \lambda^k \end{pmatrix} + \begin{pmatrix} p^k \\ p^\lambda \end{pmatrix}, \tag{1.18}$$

where $p^k$ and $p^\lambda$ solve the following Newton-KKT system

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x^k,\lambda^k) & -\nabla c(x^k)^T \\ \nabla c(x^k) & 0 \end{pmatrix} \begin{pmatrix} p^k \\ p^\lambda \end{pmatrix} = \begin{pmatrix} -\nabla r(x^k)^T r(x^k) + \nabla c(x^k)^T \lambda^k \\ -c(x^k) \end{pmatrix}. \tag{1.19}$$

The Hessian of the Lagrangian in (1.19) can be approximated by

$$\nabla^2_{xx}\mathcal{L}(x,\lambda) = \nabla r(x)^T \nabla r(x) + \sum_{i=1}^{n_{lsq}} r_i(x)\nabla^2 r_i(x) - \sum_{i=0}^{n_m} \lambda_i \nabla^2 c_i(x) \tag{1.20a}$$

$$\approx \nabla r(x)^T \nabla r(x). \tag{1.20b}$$

This approximation can be made if (see [25]):

- The residuals $r_i(x)$ are small.
- The multipliers $\lambda$ become small close to the solution.

The latter statement can be expected, because the gradient of the Lagrangian (1.17) should become small close to the solution. Consequentially, because of the first statement expecting that the residuals are small, the multipliers $\lambda$ become small close to the solution as well. With approximation (1.20b) the Newton-KKT system can be written as

$$\begin{pmatrix} \nabla r(x^k)^T \nabla r(x^k) & -\nabla c(x^k)^T \\ \nabla c(x^k) & 0 \end{pmatrix} \begin{pmatrix} p^k \\ p^\lambda \end{pmatrix} = \begin{pmatrix} -\nabla r(x^k)^T r(x^k) + \nabla c(x^k)^T \lambda^k \\ -c(x^k) \end{pmatrix}. \tag{1.21}$$

The solution $(p^k, p^\lambda)$ of (1.21) can be furthermore interpreted as the solution of a linearized nonlinear least-squares problem as

$$\min_{p\in\mathbb{R}^n} \quad \frac{1}{2}\|r(x) + \nabla r(x)p\|_2^2 \tag{1.22a}$$

$$\text{s.t.} \quad 0 = c(x) + \nabla c(x)p, \tag{1.22b}$$

under the following assumptions:

**Assumption 1.2**

- The constraint Jacobian $\nabla c(x)$ has full row rank,
- The approximation $\nabla r(x)^T \nabla r(x)$ of the Hessian of the Lagrangian is positive definite on the null space of the constraint gradients, that is

$$d^T \nabla r(x)^T \nabla r(x) d > 0 \forall d \neq 0 \quad \text{such that} \quad \nabla c(x) d = 0. \qquad \triangle$$

If we identify the Lagrange multipliers $v^k$ with $\lambda^{k+1}$ in the equation system (1.21) both views are equivalent and, hence, a step in each iteration of a Generalized Gauß-Newton method can be computed by solving the Newton-KKT system (1.21) or by solving the linearized nonlinear least-squares problem (1.22). A sketch of the basic algorithm of the Generalized Gauß-Newton method is illustrated in the following.

**Algorithm 2** Scheme of the Generalized Gauß-Newton Method

1: Choose an initial pair $(x^0, \lambda^0)$, $k \leftarrow 0$
2: **while** a convergence test is not satisfied at $(x^k, \lambda^k)$ **do**
3: Evaluate $r(x^k), \nabla r(x^k), c(x^k)$ and $\nabla c(x^k)$
4: Compute solution $(p^k, \lambda^{k+1})$ of linearized nonlinear least-squares problem (1.22)
5: $x^{k+1} \leftarrow x^k + p^k$
6: $k \leftarrow k+1$
7: **end while**

The Generalized Gauß-Newton can also be extended to inequality-constrained nonlinear least-squares problems. Further details on the Generalized Gauß-Newton method, such as, e.g., the existence of a generalized inverse as a solution operator for the linearized problem (1.22) can be found in the work of Bock [25].

## 1.3 Mathematical Programs with Complementarity Constraints

Mathematical Programs with Complementarity Constraints (MPCCs) are optimization problems, where the so-called *complementarity constraints* appear within constraints. MPCCs are considered as generalizations of bilevel programs in which the solution of the underlying problem is implicitly characterized. An excellent overview can be found, e.g., in Luo et al. [119]. The constraints of MPCCs fail to satisfy standard Constraint Qualifications (CQs) at any feasible point, including the LICQ and the MFCQ, see e.g. Chen and Florian [33].

Due to their challenging treatment and their wide applicability to practical problems, MPCCs have attracted considerable research efforts during the last decades, see Scheel and Scholtes [149], Scholtes [154, 155], Outrata [133, 134], Flegel and Kanzow [56], Flegel [55], Hoheisel et al. [88], Kanzow and Schwartz [97, 98], Luo et al. [119], Pang and Fukushima [136], Ye [176], Raghunathan and Biegler [142], Steffensen and Ulbrich [162], Gfrerer [68], Guo and Ye [76], and Gfrerer and Ye [69].

Recent focus has been put on generalizations to infinite dimension by Herzog et al. [82], Hintermüller and Kopacka [84], Hintermüller and Surowiec [85], Hintermüller et al. [86], and Mehlitz and Wachsmuth [122].

### 1.3.1 Problem Formulation

MPCCs are NLPs with a special structure and can be defined as follows:

**Definition 1.9 (Mathematical Programs with Complementarity Constraints)**
A *Mathematical Program with Complementarity Constraints* is a constrained nonlinear optimization problem of the form

$$\min_{x,s,t} \quad f(x,s,t) \tag{1.23a}$$
$$\text{s.t.} \quad 0 = c_i(x,s,t), \quad i \in \mathcal{E}, \tag{1.23b}$$
$$0 \le c_i(x,s,t), \quad i \in \mathcal{I}, \tag{1.23c}$$
$$0 \le s \perp t \ge 0, \tag{1.23d}$$

where $x \in \mathbb{R}^n$, $s \in \mathbb{R}^{n_s}$, $t \in \mathbb{R}^{n_s}$ are the optimization variables, $f : \mathbb{R}^n \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \to \mathbb{R}$ is called the objective function and $c : \mathbb{R}^n \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \to \mathbb{R}^{|\mathcal{E} \cup \mathcal{I}|}$ represent the equality constraints for components $c_i(x), i \in \mathcal{E}$ and the inequality constraints for $c_i(x), i \in \mathcal{I}$. All functions are supposed to be smooth, real-valued functions, and $\mathcal{I}$ and $\mathcal{E}$ are two disjoint finite sets of indices. The so-called *complementarity constraint* $0 \le s \perp t \ge 0$ denotes $0 \le s$, $0 \le t$, $s^T t = 0$. △

The combinatorial structure of the complementarity constraint (1.23d) results in the violation of standard CQs, e.g. LICQ and MFCQ, at any feasible point of MPCCs:

**Lemma 1.10 (Violation of Standard CQs for MPCCs)**
Let $(x,s,t)$ be a feasible point for (1.23). Then MFCQ is violated for the NLP formulation of (1.23):

$$\min_{x,s,t} \quad f(x,s,t) \tag{1.24a}$$
$$\text{s.t.} \quad 0 = c_i(x,s,t), \quad i \in \mathcal{E}, \tag{1.24b}$$
$$0 \le c_i(x,s,t), \quad i \in \mathcal{I}, \tag{1.24c}$$
$$0 \le s, 0 \le t, \tag{1.24d}$$
$$0 = s^T t. \tag{1.24e}$$

△

**Proof** See Chen et al. [32], Scheel and Scholtes [149]. □

Because standard solution methods for NLPs rely on the satisfaction of CQ, the special problem class of MPCCs is challenging and needs appropriate treatment.

### 1.3.2 Towards Constraint Qualifications and Stationarity for MPCCs

In this section we present a tailored CQ for MPCCs and two selected concepts for stationarity, where we mainly follow the publication of Fletcher et al. [61] based on the development of Scheel and Scholtes [149]. Beside the last-mentioned articles, the interested reader is advised to the thesis of Flegel [55] and references therein for a comprehensive discussion on CQs and stationarity for MPCCs.

We start with two index sets $\mathcal{I}_s, \mathcal{I}_t \subset \{1,\dots,n_s\}$ with

$$\mathcal{I}_s \cup \mathcal{I}_t = \{1,\dots,n_s\},$$

and denote their respective complements in $\{1,\dots,n_s\}$ by $\mathcal{I}_s^c$, $\mathcal{I}_t^c$. Note that the finite index sets $\{1,\dots,n_s\}$, $\mathcal{E}$, and $\mathcal{I}$ are chosen to be disjoint. We define:

**Definition 1.11 (Relaxed Nonlinear Programming Problem (RNLP))**
The Relaxed Nonlinear Programming Problem (RNLP) corresponding to the MPCC (1.23) is defined as

$$\min_{x,s,t} \quad f(x,s,t) \tag{1.25a}$$
$$\text{s.t.} \quad 0 = c_i(x,s,t), \quad i \in \mathcal{E}, \tag{1.25b}$$
$$0 \leq c_i(x,s,t), \quad i \in \mathcal{I}, \tag{1.25c}$$
$$0 = s_i, \quad i \in \mathcal{I}_t^c, \tag{1.25d}$$
$$0 \leq s_i, \quad i \in \mathcal{I}_t, \tag{1.25e}$$
$$0 = t_i, \quad i \in \mathcal{I}_s^c, \tag{1.25f}$$
$$0 \leq t_i, \quad i \in \mathcal{I}_s. \tag{1.25g}$$

△

Note that if $(x^*, s^*, t^*)$ is a local solution of the Relaxed Nonlinear Programming Problem (RNLP) (1.25) and satisfies the complementarity constraint (1.23d), then it is also a local solution of the original MPCC (1.23) of definition 1.9.

**Definition 1.12 (Set of Degenerate Indices)**
Let $(x, s, t)$ be feasible for MPCC (1.23). Then the *set of degenerate indices* is defined by

$$\mathcal{D}(x,s,t) := \{i | s_i = t_i = 0\} \text{ or } \mathcal{D} := \mathcal{I}_s \cap \mathcal{I}_t. \tag{1.26}$$

△

**Definition 1.13 (MPCC Lagrangian)**
The *MPCC Lagrangian* $\mathcal{L}^{cc} : \mathbb{R}^n \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \to \mathbb{R}$ is given by

$$\mathcal{L}^{cc}(x,s,t,\lambda,\mu,\nu,\sigma) := f(x,s,t) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x) - \sum_{i\in\mathcal{I}} \mu_i c_i(x) - \nu^T s - \sigma^T t,$$

with *Lagrange multipliers* $\lambda \in \mathbb{R}^{|\mathcal{E}|}$, $\mu \in \mathbb{R}^{|\mathcal{I}|}$, $\nu \in \mathbb{R}^{n_s}$ and $\sigma \in \mathbb{R}^{n_s}$. △

In the following we define a CQ tailored to the MPCC (1.23) - the MPCC-LICQ:

**Definition 1.14 (MPCC-LICQ)**
Let $s, t \geq 0, c_i(x,s,t) = 0, \forall i \in \mathcal{E}, c_i(x,s,t) \geq 0, \forall i \in \mathcal{I}$ and define index sets

$$\mathcal{I}_s := \{i \in \{1,\dots,n_s\} | s_i = 0\} \quad \text{and} \quad \mathcal{I}_t := \{i \in \{1,\dots,n_s\} | t_i = 0\}.$$

Then the MPCC (1.23) is said to satisfy the *MPCC-LICQ* at $(x, s, t)$, if the corresponding RNLP (1.25) satisfies LICQ at this point. △

In the remainder of this section we give the definitions of two different and closely related stationarity concepts: *Bouligand-stationarity* and *strong stationarity*. In [55], [114] and [176], e.g., an overview on more stationarity concepts for MPCCs can be found.

**Definition 1.15 (B-Stationarity)**
A point $(x^*, s^*, t^*)$ is called Bouligand, or *B-stationary*, if $d = 0$ solves the following linear program with complementarity constraints:

$$\min_{d} \quad \nabla f(x^*,s^*,t^*)^T d \tag{1.27a}$$
$$\text{s.t.} \quad 0 = c_i(x^*,s^*,t^*) + \nabla c_i(x^*,s^*,t^*) d, \quad i \in \mathcal{E}, \tag{1.27b}$$

$$0 \le c_i(x^*, s^*, t^*) + \nabla c_i(x^*, s^*, t^*) d, \quad i \in \mathcal{I}, \tag{1.27c}$$

$$0 \le (s^* + d_s) \perp (t^* + d_t) \ge 0. \tag{1.27d}$$

△

**Definition 1.16 (Strong Stationarity)**
A point $(x^*, s^*, t^*)$ is called strong stationary, if there exist multipliers $\lambda, \mu, \nu, \sigma$ such that:

$$0 = \nabla_{x,s,t} \mathcal{L}^{cc}(x^*, s^*, t^*, \lambda, \mu, \nu, \sigma), \tag{1.28a}$$

$$0 = c_i(x^*, s^*, t^*), \qquad i \in \mathcal{E}, \tag{1.28b}$$

$$0 \le c_i(x^*, s^*, t^*), \qquad i \in \mathcal{I}, \tag{1.28c}$$

$$0 \le s^*, \tag{1.28d}$$

$$0 \le t^*, \tag{1.28e}$$

$$0 = s_i^* \quad \text{or} \quad 0 = t_i^*, \qquad i \in \{1, \dots, n_s\} \tag{1.28f}$$

$$0 \le \mu, \tag{1.28g}$$

$$0 = \mu_i c_i(x^*, s^*, t^*), \qquad i \in \mathcal{I}, \tag{1.28h}$$

$$0 = \nu \circ s^*, \tag{1.28i}$$

$$0 = \sigma \circ t^*, \tag{1.28j}$$

$$0 \le \nu_i \quad \text{and} \quad 0 \le \sigma_i, \quad \text{if} \quad i \in \mathcal{D}(x^*, s^*, t^*)(1.26), \tag{1.28k}$$

where $\circ$ denotes the element-wise vector multiplication (Hadamard product). △

The conditions for strong stationarity in (1.28) of an MPCC (1.23) are the stationarity conditions of the relaxed NLP (1.25) from definition 1.11 at $(x^*, s^*, t^*)$. Scheel and Scholtes [149] have shown that B-stationarity is equivalent to strong stationarity if MPCC-LICQ holds.

### 1.3.3 Numerical Methods for MPCCs

Several approaches to MPCCs have been investigated where the complementarity constraint is expressed as a nonlinear equation. These primarily include nonlinear equation, smoothing and regularization, penalization, and structural approaches which are explained briefly in the following. We mainly rely on the comprehensive overview in the thesis of Lenders [111], where the interested reader is advised to.

#### Nonlinear Equation Approaches

Treating the equilibrium or complementarity constraint as a nonlinear equation and solving the arising problem with an algorithm for general nonlinear programs, especially SQP methods, showed success on a wide range of practical problems, see e.g. Leyffer [113], Fletcher and Leyffer [60], Fletcher et al. [61]. However, this methodology still shows deficits such as the fact that convergence to points with trivial descents cannot be excluded without strong assumptions. Recently, Andreani et al. [14] could improve the results of Izmailov et al. [93] for the case with unbounded multipliers by showing that a second order augmented Lagrangian method converges to M-stationary points under MPCC-LICQ.

#### Smoothing and Regularization Approaches

Smoothing and regularization approaches in which the complementarity constraint is expressed as a nonlinear complementarity constraint function have also been extensively studied by Facchinei et al. [50], Scholtes [153], Ralph and Wright [143], Pang and Fukushima [136], and several approaches were suggested, e.g. by Fukushima and Tseng [63], Raghunathan and Biegler [142], Liu and Sun [118], Lin and Fukushima [116, 117], DeMiguel

et al. [42], Kadrani et al. [96], Steffensen and Ulbrich [162], Kanzow and Schwartz [98], Hatz [80], Stein [163], Chen and Wan [34]. A review of these approaches can be found in Hoheisel et al. [88].

#### Penalization Approach

Another approach is to reformulate the problem by penalizing the objective function with the complementarity constraint, e.g. performed by Fukushima et al. [64], Hu and Ralph [91], Luo et al. [119], Leyffer et al. [115], Jiang and Ralph [95], Scholtes and Stöhr [156], Stöhr [165], Benson et al. [22], Anitescu [15, 16], Anitescu et al. [17], Clason et al. [36].

#### Structural Function Approach

Further methods were developed where the complementarity constraint is formulated as a structural constraint in a subproblem by Giallombardo and Ralph [70], Kirches et al. [102], Benko and Gfrerer [21]. One notable example is the combination of SQP with Sequential Linear Programming performed in Leyffer and Munson [114]. With this approach convergence to points without trivial descents under realizable conditions could be achieved. Lenders [111] developed a similar algorithm to this approach which is an extension of the Sequential Linear Equality Constraint Quadratic Programming Method (SLEQP) of Nocedal and Waltz in [30, 31] for nonlinear programs to MPCCs. A more recent approach which falls in this category was suggested by Kirches et al. [103]: the Sequential Linearization Method for Bound-Constrained MPCCs, where in each iteration of the proposed algorithm a linear program with complementarity constraints is solved to obtain an estimate of the active set.

# Chapter 2

# Optimization of Dynamic Systems

This chapter focuses on optimization of dynamic systems described by ODEs, and in particular on OCPs, PE Problems and Bilevel Inverse OCPs. In the first section various problem formulations for OCPs are given and their transformations are discussed following the thesis of Meyer [124]. The second section is concerned with the Direct Multiple Shooting Method developed by Bock and Plitt [26, 138] as a tailored numerical method to treat OCPs and gives a brief overview on possible methods for derivative generation. In the third section PE Problems are discussed with their problem formulation and an approach to solve these kinds of problems. For more details on PE Problems we refer to the works of Bock [25] and Schlöder [150]. The last section introduces a special class of hierarchical optimization of dynamics systems: a Bilevel Inverse OCP. The corresponding problem formulation for this PE Problem constrained by an OCP is provided and various solution approaches are discussed with a more detailed discussion on the Direct All-at-Once Approach [80]. For more details on Bilevel Inverse OCPs, the interested reader is referred to the references given in the last section.

## 2.1 Optimal Control of Dynamic Systems

In this section we start with a general problem formulation for OCPs and give possible transformation techniques to end up with a special multi-stage OCP with discontinuities, which later can be used to model the human gait as described in section 3.1.

### 2.1.1 General Problem Formulation

We start with a general OCP for dynamic processes described by an ODE system on time horizon $\mathcal{T} = [t_\mathrm{s}, t_\mathrm{f}] \subset \mathbb{R}$ with fixed initial time $t_\mathrm{s}$ and final time $t_\mathrm{f}$, and $t_\mathrm{s} < t_\mathrm{f}$. It can be formulated as

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \boldsymbol{\phi}^\mathrm{M}(t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f})) + \int_{t_\mathrm{s}}^{t_\mathrm{f}} \boldsymbol{\phi}^\mathrm{L}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t \tag{2.1a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad t \in \mathcal{T}, \tag{2.1b}$$

$$0 \le \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad t \in \mathcal{T}, \tag{2.1c}$$

$$0 = \boldsymbol{r}^\mathrm{ec}(t_\mathrm{s}, \boldsymbol{x}(t_\mathrm{s}), t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f})), \tag{2.1d}$$

$$0 \le \boldsymbol{r}^\mathrm{ic}(t_\mathrm{s}, \boldsymbol{x}(t_\mathrm{s}), t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f})). \tag{2.1e}$$

In the following the arising variables and sufficiently smooth functions in OCP (2.1) are described:

- Bolza type objective (2.1a) incorporates a Mayer term $\boldsymbol{\phi}^\mathrm{M} : \mathbb{R} \times \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}$ and a Lagrange term where $\boldsymbol{\phi}^\mathrm{L} : \mathcal{T} \times \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}$,
- differential states denoted by $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n_\mathrm{x}}$,
- control functions denoted by $\boldsymbol{u} : \mathcal{T} \to \mathbb{R}^{n_\mathrm{u}}$,
- dynamics described by a system of ODEs (2.1b) with the right-hand side $\boldsymbol{f} : \mathcal{T} \times \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$,
- boundary equality constraints $\boldsymbol{r}^\mathrm{ec} : \mathcal{T} \times \mathbb{R}^{n_\mathrm{x}} \times \mathcal{T} \times \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{ec}}$ in (2.1d) and
- inequality constraints $\boldsymbol{r}^\mathrm{ic} : \mathcal{T} \times \mathbb{R}^{n_\mathrm{x}} \times \mathcal{T} \times \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{ic}}$ constraints in (2.1d),

- mixed control-state constraints combined in $\boldsymbol{c} : \mathcal{T} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ (2.1c).

The OCP of general form described here is a so-called *non-autonomous* problem. In problems of this kind, the arising functions depend explicitly on the time of the process, whereas in *autonomous* problems this is not the case. Every non-autonomous problem (2.1) can be transformed into an autonomous problem. This transformation technique together with other selected transformations as described in the thesis of Meyer [124] are introduced in the following. As a result, we can then formulate a multi-stage OCP (2.7) in subsection 2.1.2 which is equivalent to the general problem formulation in (2.1), and, hence, general numerical approaches can be applied to the special class as well.

#### Transformation to Autonomous Problem

With the introduction of an additional differential state $\boldsymbol{t}(\cdot)$ with $\dot{\boldsymbol{t}}(t) = 1,\ \boldsymbol{t}(t_s) = t_s$, a non-autonomous problem can be easily transformed to an autonomous problem.

#### Embedding of Global Parameters

Global parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$ can be a part of the dynamic model and may enter objective functions $\boldsymbol{\phi}^M(\cdot)$ and $\boldsymbol{\phi}^L(\cdot)$, right-hand side functions $\boldsymbol{f}(\cdot)$ or equality and inequality constraint functions $\boldsymbol{r}^{ic}(\cdot)$ and $\boldsymbol{r}^{ec}(\cdot)$, respectively, and mixed control-state constraints $\boldsymbol{c}(\cdot)$. Such model parameters can be treated as additional controls, which are constant over $\mathcal{T}$. In the following we call them constant control parameters.

#### Transformation to Multi-Stage Problems

Many processes in nature cannot easily be represented by a single ODE system because the dynamics may change. To overcome this drawback the dynamics can be modeled in phases and be embedded in a multi-stage OCP. Therefore, $n_S$ consecutive so-called *model stages* with index $j = 1, \dots, n_S$ are defined on fixed possibly empty time horizons $\mathcal{T}^j := [t_{j-1}, t_j]$ with ordered time points

$$t_s = t_0 \le t_1 \le \ldots \le t_{n_S} = t_f. \tag{2.2}$$

This results in differential states $\boldsymbol{x}_j(t)$ and control functions $\boldsymbol{u}_j(t)$ on each model stage. Together with phasewise defined Mayer type and Langrange type objective functionals $\boldsymbol{\phi}_j^M(\cdot)$ and $\boldsymbol{\phi}_j^L(\cdot)$, respectively, right-hand side functions $\boldsymbol{f}_j(\cdot)$ and mixed control-state constraints $\boldsymbol{c}_j(\cdot)$ the multi-stage OCP can be formulated as

$$\min_{\substack{\boldsymbol{x}_1,\dots,\boldsymbol{x}_{n_S},\\ \boldsymbol{u}_1,\dots,\boldsymbol{u}_{n_S}}} \quad \sum_{j=1}^{n_S} \boldsymbol{\phi}_j^M(t_j, \boldsymbol{x}_j(t_j)) + \int_{t_{j-1}}^{t_j} \boldsymbol{\phi}_j^L(t, \boldsymbol{x}_j(t), \boldsymbol{u}_j(t))\,\mathrm{d}t \tag{2.3a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_j(t) = \boldsymbol{f}_j(t_j, \boldsymbol{x}_j(t), \boldsymbol{u}_j(t)), \qquad t \in \mathcal{T}^j,\ j = 1, \dots, n_S, \tag{2.3b}$$

$$0 \le \boldsymbol{c}_j(t_j, \boldsymbol{x}_j(t), \boldsymbol{u}_j(t)), \qquad t \in \mathcal{T}^j,\ j = 1, \dots, n_S, \tag{2.3c}$$

$$0 = \boldsymbol{r}^{ec}(t_0, \boldsymbol{x}_1(t_0), t_1, \boldsymbol{x}_1(t_1), \dots, \boldsymbol{x}_{n_S}(t_{n_S})), \tag{2.3d}$$

$$0 \le \boldsymbol{r}^{ic}(t_0, \boldsymbol{x}_1(t_0), t_1, \boldsymbol{x}_1(t_1), \dots, \boldsymbol{x}_{n_S}(t_{n_S})). \tag{2.3e}$$

Boundary constraints are included in (2.3d) and (2.3d).

#### Transformation to Fixed Time Intervals

The general problem formulation in (2.1) and the multi-stage OCP (2.3) are defined on time horizons with fixed initial time $t_s$ and fixed final time $t_f$. However, in many applications, processes with a priori unknown final time have to be considered. In this paragraph we show how the resulting OCP with free final time can be transformed onto a fixed and normalized time interval $[0, 1]$ such that it is equivalent to problem formulations (2.1) and (2.3).

In this work we focus on multi-stage OCPs with discontinuities with free final time, later introduced in subsection 2.1.2. Hence, at this point we also consider phasewise defined dynamic processes as in the previous paragraph on possibly empty time horizons $\mathcal{T}^{\mathrm{j}} := [t_{\mathrm{j}-1}, t_{\mathrm{j}}]$ for each model stage $\mathrm{j} = 1, \dots, \mathrm{n_S}$ with free time points ordered as in (2.2). However, for $\mathrm{n_S} = 1$ the general case of only one model stage is included. To perform a linear time transformation, we fix initial time $t_{\mathrm{s}} = 0$ of the first model stage without loss of generality and consider a mapping $\boldsymbol{t}_{\mathrm{j}} : [0,1] \to \mathcal{T}^{\mathrm{j}}$ on each model stage $\mathrm{j} = 1, \dots, \mathrm{n_S}$ defined by

$$\boldsymbol{t}_{\mathrm{j}}(\tilde{t}) := \sum_{\mathrm{j}=1}^{\mathrm{n_S}-1} d_{\mathrm{j}} + \tilde{t} \cdot d_{\mathrm{j}},$$

where we introduce the duration parameters $d_{\mathrm{j}} = t_{\mathrm{j}} - t_{\mathrm{j}-1}$ and combine them in the vector $\boldsymbol{d} := \begin{pmatrix} d_1 & \dots & d_{\mathrm{n_S}} \end{pmatrix}^T$. This parameter vector can be embedded in an OCP of the form (2.1) or (2.3) according to the previous paragraph "Embedding of Global Parameters" on page 22. With the definition of the differential states and control functions as

$$\begin{aligned} \tilde{\boldsymbol{x}}_{\mathrm{j}} &: [0,1] \to \mathbb{R}^{\mathrm{n_{x_j}}}, \quad \tilde{\boldsymbol{x}}_{\mathrm{j}}(\tilde{t}) := \boldsymbol{x}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t})), \\ \tilde{\boldsymbol{u}}_{\mathrm{j}} &: [0,1] \to \mathbb{R}^{\mathrm{n_{u_j}}}, \quad \tilde{\boldsymbol{u}}_{\mathrm{j}}(\tilde{t}) := \boldsymbol{u}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t})), \end{aligned}$$

where $\boldsymbol{t}_{\mathrm{j}}(\tilde{t}) \in \mathcal{T}^{\mathrm{j}}$, the ODE system for all model stages is given by

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}}_{\mathrm{j}}(\tilde{t}) &= \frac{\partial \boldsymbol{x}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t}))}{\partial \boldsymbol{t}_{\mathrm{j}}} \cdot \frac{\partial \boldsymbol{t}_{\mathrm{j}}(\tilde{t})}{\partial \tilde{t}} \\ &= d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t}), \boldsymbol{x}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t})), \boldsymbol{u}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t})))) \\ &= d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{t}_{\mathrm{j}}(\tilde{t}), \tilde{\boldsymbol{x}}_{\mathrm{j}}(\tilde{t}), \tilde{\boldsymbol{u}}_{\mathrm{j}}(\tilde{t})). \end{aligned}$$

Note that in general, the dimensions of the differential states and control functions, $\mathrm{n_{x_j}}$ and $\mathrm{n_{u_j}}$, respectively, may differ on each model stage. However, in our applications of chapter 8, chapter 9, and chapter 10 the dimensions of the differential states and control functions, $\mathrm{n_x}$ and $\mathrm{n_u}$, respectively, stay the same on each model stage. For all remaining functions in OCP (2.1) and (2.3) - the objective function and the constraints - we can proceed in a similar way. In total, this results in an OCP of the form

$$\min_{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \boldsymbol{d}} \quad \boldsymbol{\phi}^{\mathrm{M}}(\boldsymbol{t}(1), \tilde{\boldsymbol{x}}(1)) + \int_0^1 \boldsymbol{\phi}^{\mathrm{L}}(\boldsymbol{t}(\tilde{t}), \tilde{\boldsymbol{x}}(\tilde{t}), \tilde{\boldsymbol{u}}(\tilde{t}), \boldsymbol{d}) \mathrm{d}\tilde{t} \tag{2.6a}$$

$$\text{s.t.} \quad \dot{\tilde{\boldsymbol{x}}}(\tilde{t}) = \boldsymbol{d} \circ \boldsymbol{f}(\boldsymbol{t}(\tilde{t}), \tilde{\boldsymbol{x}}(\tilde{t}), \tilde{\boldsymbol{u}}(\tilde{t})), \qquad \tilde{t} \in [0,1], \tag{2.6b}$$

$$0 \le \boldsymbol{c}(\boldsymbol{t}(\tilde{t}), \tilde{\boldsymbol{x}}(\tilde{t}), \tilde{\boldsymbol{u}}(\tilde{t})), \qquad \tilde{t} \in [0,1], \tag{2.6c}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{t}(0), \tilde{\boldsymbol{x}}(0), \boldsymbol{t}(1), \tilde{\boldsymbol{x}}(1)), \tag{2.6d}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{t}(0), \tilde{\boldsymbol{x}}(0), \boldsymbol{t}(1), \tilde{\boldsymbol{x}}(1)), \tag{2.6e}$$

where $\circ$ denotes the element-wise vector multiplication (Hadamard product). Problem (2.6) is defined on fixed normalized time horizons $\tilde{t} \in [0,1]$ for each model stage with duration parameters $\boldsymbol{d} \in \mathbb{R}^{\mathrm{n_S}}$ as time-independent optimization variables. The other variables are represented in their compact form

$$\tilde{\boldsymbol{x}}(\tilde{t}) := \begin{pmatrix} \tilde{\boldsymbol{x}}_1(\tilde{t}) \\ \vdots \\ \tilde{\boldsymbol{x}}_{\mathrm{n_S}}(\tilde{t}) \end{pmatrix} \quad \text{and} \quad \tilde{\boldsymbol{u}}(\tilde{t}) := \begin{pmatrix} \tilde{\boldsymbol{u}}_1(\tilde{t}) \\ \vdots \\ \tilde{\boldsymbol{u}}_{\mathrm{n_S}}(\tilde{t}) \end{pmatrix}.$$

All remaining phasewise defined functions which appear in OCP (2.6) are treated similarly to the right hand side function

$$\boldsymbol{f}(\boldsymbol{t}(\tilde{t}),\tilde{\boldsymbol{x}}(\tilde{t}),\tilde{\boldsymbol{u}}(\tilde{t})) := \begin{pmatrix} \boldsymbol{f}_1(\boldsymbol{t}_1(\tilde{t}),\tilde{\boldsymbol{x}}_1(\tilde{t}),\tilde{\boldsymbol{u}}_1(\tilde{t})) \\ \vdots \\ \boldsymbol{f}_{\mathrm{n_S}}(\boldsymbol{t}_{\mathrm{n_S}}(\tilde{t}),\tilde{\boldsymbol{x}}_{\mathrm{n_S}}(\tilde{t}),\tilde{\boldsymbol{u}}_{\mathrm{n_S}}(\tilde{t})) \end{pmatrix}.$$

In the following we omit the tilde which appears in the quantities after time transformation for autonomous problems for a clearer presentation.

#### Transformations of Objective Functions

The objective function in (2.1a) is of Bolza type and combines a Mayer type objective $\boldsymbol{\phi}^{\mathrm{M}}(\cdot)$ and a Lagrange type objective $\boldsymbol{\phi}^{\mathrm{L}}(\cdot)$. With the introduction of additional differential states, differentiation, and integration these types can be easily transformed into each other.

### 2.1.2 Multi-Stage Problem Formulation with Discontinuities

In this thesis we consider human locomotion of rigid multi-body system models which can be interpreted as an OCP, see section 3.1 for more details. Its dynamics are often phasewise defined with discontinuities between phases. If the sequence of the dynamics with its transitions is not known in advance, it results in a *switched* OCP as, e.g., in [124, 151]. Contrary to this, if the consecutive order of all phases is predefined human locomotion can be formulated by multi-stage OCPs with discontinuities. These kinds of problems are numerically easier to solve as the challenging switched OCPs.

In this thesis we are concerned with Bilevel Inverse OCPs: PE problems constrained by an OCP as introduced later in section 2.4 with the ultimate goal to describe the individual gait of a CP patient. Hence, in our case with given measurement data of the dynamic process, the sequence of the dynamics is already predefined. Therefore, on the lower hierarchical level we choose a multi-stage OCP formulation with discontinuities, which incorporates the multi-stage structure in consecutive order on fixed *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [0,1]$ for each model stage $\mathrm{j} = 1,\dots,\mathrm{n_S}$ with free duration parameters $d_{\mathrm{j}} \in \mathbb{R}$. For autonomous systems with model parameters $\boldsymbol{p} \in \mathbb{R}^{\mathrm{n_p}}$ this OCP can be formulated as

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \sum_{\mathrm{j}=1}^{\mathrm{n_S}} \boldsymbol{\Phi}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),d_{\mathrm{j}},\boldsymbol{\alpha},\boldsymbol{p}) \tag{2.7a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_{\mathrm{j}}(t) = d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j} = 1,\dots,\mathrm{n_S}, \tag{2.7b}$$

$$\boldsymbol{x}_{\mathrm{j}+1}(0) = \Delta_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(1),\boldsymbol{p}), \qquad \mathrm{j} = 1,\dots,\mathrm{n_S}-1, \tag{2.7c}$$

$$0 \le \boldsymbol{c}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j} = 1,\dots,\mathrm{n_S}, \tag{2.7d}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{x}_1(0),\boldsymbol{x}_1(1),\dots,\boldsymbol{x}_{\mathrm{n_S}}(1),\boldsymbol{p}), \tag{2.7e}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{x}_1(0),\boldsymbol{x}_1(1),\dots,\boldsymbol{x}_{\mathrm{n_S}}(1),\boldsymbol{p}), \tag{2.7f}$$

where the phasewise defined quantities and functions $\boldsymbol{x}_{\mathrm{j}}(\cdot)$, $\boldsymbol{u}_{\mathrm{j}}(\cdot)$, $\boldsymbol{f}_{\mathrm{j}}(\cdot)$, $\boldsymbol{c}_{\mathrm{j}}(\cdot)$, and $\boldsymbol{\Phi}_{\mathrm{j}}(\cdot)$ correspond to the ones in the general problem formulation (2.1) after time transformation as in (2.6) combined with a transformation to autonomous systems and an embedding of global parameters from subsection 2.1.1. The objective function on each model stage is of Bolza type and is represented by a weighted sum of $\mathrm{n_M}$ Mayer type optimization criteria $\boldsymbol{\phi}^{\mathrm{M}}_{\mathrm{jk}}$ and $\mathrm{n_L}$ Lagrange type optimization criteria $\boldsymbol{\phi}^{\mathrm{L}}_{\mathrm{jk}}$ as follows

$$\boldsymbol{\Phi}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),d_{\mathrm{j}},\boldsymbol{\alpha},\boldsymbol{p}) = \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} \cdot \boldsymbol{\phi}^{\mathrm{M}}_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(1),d_{\mathrm{j}},\boldsymbol{p}) + \sum_{\mathrm{k}=\mathrm{n_M}+1}^{\mathrm{n_M}+\mathrm{n_L}} \alpha_{\mathrm{k}} \cdot \int_0^1 d_{\mathrm{j}} \cdot \boldsymbol{\phi}^{\mathrm{L}}_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p})\mathrm{d}t.$$

The corresponding ojective weights are denoted by $\boldsymbol{\alpha} \in \mathbb{R}^{n_M+n_L}$. Discontinuities between phases can be modeled by transitions of the values of differential states $\boldsymbol{x}_j$ of two consecutive model stages at phase transition. This transition is described by equation (2.7c). If no discontinuity between two phases occurs, only an identity mapping has to be considered.

If additional knowledge of the process is available further multi-point equality and inequality conditions can be included in equations (2.7e) and (2.7f), respectively, and then have to be fulfilled on each model stage $j = 1,\dots,n_S$ on specific *normalized* time points

$$0 = t_{j,0}^{ec} < t_{j,1}^{ec} < \dots < t_{j,n_{ec,j}}^{ec} = 1 \quad \text{and} \quad 0 = t_{j,0}^{ic} < t_{j,1}^{ic} < \dots < t_{j,n_{ec,j}}^{ic} = 1,$$

respectively. However, in order to ensure a clear representation, equations (2.7e) and (2.7f) only comprise boundary conditions on each phase which are related to (2.1d) and (2.1e) in the general formulation. By applying the transformations described in subsection 2.1.1 and adding equation (2.7c) to the boundary condition OCP (2.7) can be represented in the general form of (2.1).

## 2.2 Numerical Methods for Optimal Control Problems

In general, we distinguish between indirect and direct solution approaches for OCPs. In indirect solution approaches the OCP is considered as an infinite dimensional optimization problem. For this continuous OCP optimality conditions are established. This results in a nonlinear multipoint boundary value problem that has to be solved numerically. In contrast to indirect solution approaches, in direct solution approaches the OCP is first discretized and transformed to a finite-dimensional NLP. This resulting NLP can then be solved by applying the methods described in section 1.2. In these numerical methods derivatives have to be generated, hence, we also give a brief overview on this topic in the last section. In this thesis we choose the direct solution approach and, in particular, the so-called Direct Multiple Shooting Method that is described in the following.

### 2.2.1 The Direct Multiple Shooting Method

The Direct Multiple Shooting Method was developed by Bock and Plitt [26, 138]. An efficient implementation of the Direct Multiple Shooting Method can be found, e.g., in the optimal control package MUSCOD-II by Leineweber [108], with a comprehensive overview of the method itself and some advanced extensions in [109, 110]. The software package ParDynOpt tailored for the DiSimFAS developed in this thesis implements the Direct Multiple Shooting Method as well.

In the following, we give a brief overview of the approach and consider a single-stage OCP similar to the multi-stage formulation from previous subsection 2.1.2 on a fixed normalized time horizon $[0,1]$ with free duration parameter $d \in \mathbb{R}$ and given model parameters $\boldsymbol{p}$ and objective weights $\boldsymbol{\alpha}$ as follows

$$\min_{\boldsymbol{x},\boldsymbol{u},d} \quad \sum_{k=1}^{n_M} \alpha_k \cdot \boldsymbol{\phi}_k^M(\boldsymbol{x}(1), d, \boldsymbol{p}) \tag{2.8a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = d \cdot \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}), \qquad t \in [0,1], \tag{2.8b}$$

$$0 \le \boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}), \qquad t \in [0,1], \tag{2.8c}$$

$$0 = \boldsymbol{r}^{ec}(\boldsymbol{x}(0), \boldsymbol{x}(1), \boldsymbol{p}), \tag{2.8d}$$

$$0 \le \boldsymbol{r}^{ic}(\boldsymbol{x}(0), \boldsymbol{x}(1), \boldsymbol{p}), \tag{2.8e}$$

where $n_S$ is set to 1 and the stage index j as well as the transition condition (2.7c) are omitted. For a clearer representation, only Mayer terms are considered in the problem formulation of (2.8). As described earlier in subsection 2.1.1 by introducing an additional state and an ODE, the Lagrange type objective can be transformed to a Mayer type objective so this is an adequate formulation and is used in the following to describe the Direct Multiple Shooting Method.

### Discretization of Control Function

In the direct approach the control functions $\boldsymbol{u}(t)$ on infinite dimensional space are approximated by a finite set of piecewise defined functions with local support. Therefore, we introduce the discretization grid, also called *multiple shooting grid*

$$0 = t_0^{\mathrm{ms}} < t_1^{\mathrm{ms}} < \ldots < t_{\mathrm{N}}^{\mathrm{ms}} = 1, \tag{2.9}$$

and replace the control functions $\boldsymbol{u}(t)$ on each *multiple shooting interval* $\mathcal{T}^{\mathrm{i}} = [t_{\mathrm{i}}^{\mathrm{ms}}, t_{\mathrm{i+1}}^{\mathrm{ms}}]$ for $\mathrm{i} = 0, \ldots, \mathrm{N}-1$ and for each component $\mathrm{j} = 1, \ldots, \mathrm{n_u}$ by some given (typically polynomial) basis functions $\boldsymbol{\xi}_{\mathrm{ij}} : \mathcal{T}^{\mathrm{i}} \times \mathbb{R}^{\mathrm{n_{q_{ij}}}} \to \mathbb{R}$ such that

$$\hat{\boldsymbol{u}}_{\mathrm{ij}}(t) := \boldsymbol{\xi}_{\mathrm{ij}}(t, \boldsymbol{q}_{\mathrm{ij}}), \quad t \in \mathcal{T}^{\mathrm{i}}, \ \mathrm{i} = 0, \ldots, \mathrm{N}-1. \tag{2.10}$$

The *locally* defined control parameters are denoted by $\boldsymbol{q}_{\mathrm{ij}} \in \mathbb{R}^{\mathrm{n_{q_{ij}}}}$. For $\mathrm{j} = 1, \ldots, \mathrm{n_u}$ the components are combined in the vectors $\boldsymbol{q}_{\mathrm{i}} := \begin{pmatrix} \boldsymbol{q}_{\mathrm{i}1}^T & \ldots & \boldsymbol{q}_{\mathrm{in_u}}^T \end{pmatrix}^T$. For completeness, for the final node $t_{\mathrm{N}}^{\mathrm{ms}}$ of the chosen time grid we introduce additional control parameters such that $\boldsymbol{q}_{\mathrm{N}} := \boldsymbol{q}_{\mathrm{N}-1}$ and

$$\hat{\boldsymbol{u}}_{\mathrm{N}}(t_{\mathrm{N}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{N}}) := \hat{\boldsymbol{u}}_{\mathrm{N}-1}(t_{\mathrm{N}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{N}-1}).$$

In total, the control functions $\boldsymbol{u}(\cdot)$ can be approximated by

$$\hat{\boldsymbol{u}}(t, \boldsymbol{q}) = \begin{cases} \hat{\boldsymbol{u}}_{\mathrm{i}}(t, \boldsymbol{q}_{\mathrm{i}}), & \text{if } t \in [t_{\mathrm{i}}^{\mathrm{ms}}, t_{\mathrm{i+1}}^{\mathrm{ms}}), \ \text{for } \mathrm{i} = 0, \ldots, \mathrm{N}-1, \\ \hat{\boldsymbol{u}}_{\mathrm{N}}(t, \boldsymbol{q}_{\mathrm{N}}), & \text{if } t = t_{\mathrm{N}}^{\mathrm{ms}}, \end{cases} \tag{2.11}$$

with $\hat{\boldsymbol{u}}_{\mathrm{i}}(\cdot) = \begin{pmatrix} \hat{\boldsymbol{u}}_{\mathrm{i}1}(\cdot) & \ldots & \hat{\boldsymbol{u}}_{\mathrm{in_u}}(\cdot) \end{pmatrix}^T$ where each component is defined in (2.10) and control parameters combined in $\boldsymbol{q} := \begin{pmatrix} \boldsymbol{q}_0^T & \ldots & \boldsymbol{q}_{\mathrm{N}}^T \end{pmatrix}^T$. For instance, piecewise constant approximations for all $\mathrm{n_u}$ components of control functions $\boldsymbol{u}(t)$ can be written as

$$\hat{\boldsymbol{u}}_{\mathrm{i}}(t, \boldsymbol{q}_{\mathrm{i}}) = \boldsymbol{q}_{\mathrm{i}}, \quad t \in \mathcal{T}^{\mathrm{i}}, \ \mathrm{i} = 0, \ldots, \mathrm{N}-1.$$

If a piecewise linear approximation is preferred on each multiple shooting interval for $\mathrm{i} = 0, \ldots, \mathrm{N}-1$ and for each component $\mathrm{j} = 1, \ldots, \mathrm{n_u}$ we have

$$\hat{\boldsymbol{u}}_{\mathrm{ij}}(t, \boldsymbol{q}_{\mathrm{ij}}) = q_{\mathrm{ij}}^1 + \frac{t - t_{\mathrm{i-1}}^{\mathrm{ms}}}{t_{\mathrm{i}}^{\mathrm{ms}} - t_{\mathrm{i-1}}^{\mathrm{ms}}} \left( q_{\mathrm{ij}}^2 - q_{\mathrm{ij}}^1 \right), \quad t \in \mathcal{T}^{\mathrm{i}}, \ \mathrm{i} = 0, \ldots, \mathrm{N}-1,$$

with $\boldsymbol{q}_{\mathrm{ij}} := \begin{pmatrix} q_{\mathrm{ij}}^1 & q_{\mathrm{ij}}^2 \end{pmatrix}^T \in \mathbb{R}^2$. If desired, continuity can be ensured at the *multiple shooting nodes* by additional constraints such as

$$\hat{\boldsymbol{u}}_{\mathrm{i}}(t_{\mathrm{i+1}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{i}}) = \hat{\boldsymbol{u}}_{\mathrm{i+1}}(t_{\mathrm{i+1}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{i+1}}), \quad \mathrm{i} = 0, \ldots, \mathrm{N}-2,$$

e.g., in the piecewise linear case. Similarly, approximations with higher polynomial order can be used. However, for many applications piecewise constant or piecewise linear control approximations are sufficient. Although basis functions with higher order represent the true control functions more accurately from the mathematical point of view, they often cannot be realized in the application itself.

After control discretization, the control functions in OCP (2.8) are replaced by control parameters $\boldsymbol{q} \in \mathbb{R}^{\mathrm{n_q}}$ as optimization variables which enter the arising functions, e.g. the ODE in (2.8b): $\boldsymbol{f}(\boldsymbol{x}(t), \hat{\boldsymbol{u}}(t, \boldsymbol{q}), \boldsymbol{p})$. In the following we mostly use piecewise constant approximations of all control functions.

### Parametrization of Dynamics

Suppose we have the same multiple shooting grid (2.9) as in the previous section

$$0 = t_0^{\mathrm{ms}} \leq t_1^{\mathrm{ms}} \leq \ldots \leq t_{\mathrm{N}}^{\mathrm{ms}} = 1,$$

and choose an approximated control function from (2.11). Then the ODE model is parametrized by the introduction of variables $\boldsymbol{s} = \begin{pmatrix} \boldsymbol{s}_0^T & \ldots & \boldsymbol{s}_{\mathrm{N}}^T \end{pmatrix}^T$ with $\boldsymbol{s}_{\mathrm{i}} \in \mathbb{R}^{\mathrm{n_x}}$ such that on each multiple shooting interval $\mathcal{T}^{\mathrm{i}} = [t_{\mathrm{i}}^{\mathrm{ms}}, t_{\mathrm{i+1}}^{\mathrm{ms}}]$ for $\mathrm{i} = 0, \ldots, \mathrm{N}-1$ the following Initial Value Problems (IVPs)

$$\dot{\boldsymbol{x}}(t) = d \cdot \boldsymbol{f}(\boldsymbol{x}(t), \hat{\boldsymbol{u}}_{\mathrm{i}}(t, \boldsymbol{q}_{\mathrm{i}}), \boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{i}} \text{ for } \mathrm{i} = 0, \ldots, \mathrm{N}-1, \tag{2.12a}$$

$$\boldsymbol{x}(t_{\mathrm{i}}^{\mathrm{ms}}) = \boldsymbol{s}_{\mathrm{i}}, \tag{2.12b}$$

are solved. The interval solutions are denoted by $\boldsymbol{x}(t; \boldsymbol{s}_{\mathrm{i}}, \boldsymbol{q}_{\mathrm{i}}, d, \boldsymbol{p})$ to emphasize the dependence on the variables $\boldsymbol{s}_{\mathrm{i}}$, control parameters $\boldsymbol{q}_{\mathrm{i}}$, duration parameter $d$, and model parameters $\boldsymbol{p}$ at multiple shooting node $t_{\mathrm{i}}^{\mathrm{ms}}$. To ensure a continuous solution, additional continuity conditions or *matching conditions* have to be fulfilled at the multiple shooting nodes:

$$0 = \boldsymbol{x}(t_{\mathrm{i+1}}^{\mathrm{ms}}; \boldsymbol{s}_{\mathrm{i}}, \boldsymbol{q}_{\mathrm{i}}, d, \boldsymbol{p}) - \boldsymbol{s}_{\mathrm{i+1}}, \qquad \mathrm{i} = 0, \ldots, \mathrm{N}-1.$$

Similarly, by setting $\boldsymbol{x}_0(t_0^{\mathrm{ms}}) = \boldsymbol{x}_{\mathrm{s}}$ the initial condition at the first multiple shooting node

$$0 = \boldsymbol{x}_{\mathrm{s}} - \boldsymbol{s}_0, \tag{2.13}$$

has to hold and can be incorporated in the boundary conditions of OCP (2.8).

In the multi-stage setting in subsection 2.1.2 the model stage index j enters the discretized quantities for distinction between different phases, which are defined on multiple shooting grids

$$0 = t_{\mathrm{j},0}^{\mathrm{ms}} \leq t_{\mathrm{j},1}^{\mathrm{ms}} \leq \ldots \leq t_{\mathrm{j},\mathrm{N_j}}^{\mathrm{ms}} = 1, \quad \mathrm{j} = 1, \ldots, \mathrm{n_S} - 1.$$

Then the transition condition (2.7c) between two model stages $j$ and $j+1$ can be represented by

$$0 = \Delta_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{j},\mathrm{N_j}}^{\mathrm{ms}}; \boldsymbol{s}_{\mathrm{j},\mathrm{N_j}-1}, \boldsymbol{q}_{\mathrm{j},\mathrm{N_j}-1}, d_{\mathrm{j}}, \boldsymbol{p}), \boldsymbol{p}) - \boldsymbol{s}_{\mathrm{j+1},0}, \quad \mathrm{j} = 1, \ldots, \mathrm{n_S} - 1. \tag{2.14}$$

It describes the transition between the last multiple shooting node of model stage $j$ and the first node of model stage $j+1$.

### Discretization of Constraints

After discretization of the control functions and parametrization of the dynamics on the multiple shooting grid, the mixed control-state constraints in OCP (2.8) are enforced to hold only at the multiple shooting nodes such that

$$0 \leq \boldsymbol{c}(\boldsymbol{s}_{\mathrm{i}}, \hat{\boldsymbol{u}}_{\mathrm{i}}(t_{\mathrm{i}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{i}}), \boldsymbol{p}), \qquad \mathrm{i} = 0, \ldots, \mathrm{N}.$$

In many applications this assumption is sufficient. However, if this is not the case and the constraints are violated significantly between the multiple shooting nodes, one can refine the multiple shooting grid or use an approach as described in [140, 141], which leads to more computational effort.

Furthermore, the boundary equality and inequality constraints, (2.8d) and (2.8e), respectively, are written as

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}),$$
$$0 \leq \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}),$$

which also include initial condition (2.13) at the first multiple shooting node.

#### Resulting NLP

With discretization of the control functions, parametrization of the dynamics and discretization of the constraints on the fixed and normalized multiple shooting grid (2.9) together with an appropriate Mayer type objective of the form

$$\boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{\alpha}, \boldsymbol{p}) := \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} \cdot \boldsymbol{\phi}_{\mathrm{k}}^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{p}),$$

and given model parameters $\boldsymbol{p}$ and objective weights $\boldsymbol{\alpha}$, the OCP (2.8) defined on the fixed and normalized time horizon $t \in [0,1] = [t_0^{\mathrm{ms}}, t_{\mathrm{N}}^{\mathrm{ms}}]$ results in a Multiple Shooting NLP as follows:

$$\min_{\boldsymbol{s},\boldsymbol{q},d} \quad \boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{\alpha}, \boldsymbol{p}) \tag{2.15a}$$
$$\text{s.t.} \quad 0 = \boldsymbol{x}(t_{\mathrm{i+1}}^{\mathrm{ms}}; \boldsymbol{s}_{\mathrm{i}}, \boldsymbol{q}_{\mathrm{i}}, d, \boldsymbol{p}) - \boldsymbol{s}_{\mathrm{i+1}}, \qquad \mathrm{i} = 0, \ldots, \mathrm{N}-1, \tag{2.15b}$$
$$0 \leq \boldsymbol{c}(\boldsymbol{s}_{\mathrm{i}}, \hat{\boldsymbol{u}}_{\mathrm{i}}(t_{\mathrm{i}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{i}}), \boldsymbol{p}), \qquad \mathrm{i} = 0, \ldots, \mathrm{N}, \tag{2.15c}$$
$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}), \tag{2.15d}$$
$$0 \leq \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}). \tag{2.15e}$$

NLP (2.15) can be solved by the methods described in section 1.2. For the interested reader more details on structure exploiting SQP methods tailored to multi-stage OCPs can be found in [108] and [109, 110]. Interior point methods can be found, e.g., in [123, 169].

For a more compact representation of NLP (2.15) the multiple shooting variables are combined in the vectors $\boldsymbol{v}_{\mathrm{i}} := \begin{pmatrix} \boldsymbol{s}_{\mathrm{i}}^T & \boldsymbol{q}_{\mathrm{i}}^T \end{pmatrix}^T$ with $\boldsymbol{s}_{\mathrm{i}} \in \mathbb{R}^{\mathrm{n_x}}$ and $\boldsymbol{q}_{\mathrm{i}} \in \mathbb{R}^{\mathrm{n_{q_i}}}$, where the dimension of the control parameters on each node depends on the chosen control approximation. These shooting vectors are combined in the total multiple shooting variable vector $\boldsymbol{v} := \begin{pmatrix} \boldsymbol{v}_0^T & \ldots & \boldsymbol{v}_{\mathrm{N}}^T \end{pmatrix}^T \in \mathbb{R}^{\mathrm{n_v}}$, with the dimension $\mathrm{n_v} := (\mathrm{N}+1) \cdot (\mathrm{n_x} + \mathrm{n_{q_i}})$ and the corresponding simple upper and lower bounds, which are denoted by $\overline{\boldsymbol{v}} := \begin{pmatrix} \overline{\boldsymbol{v}}_0{}^T & \ldots & \overline{\boldsymbol{v}}_{\mathrm{N}}{}^T \end{pmatrix}^T$ and $\underline{\boldsymbol{v}} := \begin{pmatrix} \underline{\boldsymbol{v}}_0{}^T & \ldots & \underline{\boldsymbol{v}}_{\mathrm{N}}{}^T \end{pmatrix}^T$, respectively. Without loss of generality we assume that all simple bounds are excluded from inequality constraints $\boldsymbol{c}(\cdot)$ from (2.15c) and $\boldsymbol{r}^{\mathrm{ic}}(\cdot)$ from (2.15e), and use notations $\tilde{\boldsymbol{c}}(\cdot)$ and $\tilde{\boldsymbol{r}}_{\mathrm{ic}}(\cdot)$ in the following. Together with this assumption all equality and inequality constraints without simple bounds can be combined in constraint functions

$$\boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v}, d, \boldsymbol{p}) := \begin{pmatrix} \boldsymbol{x}(t_1^{\mathrm{ms}}; \boldsymbol{s}_0, \boldsymbol{q}_0, d, \boldsymbol{p}) - \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{x}(t_{\mathrm{N}}^{\mathrm{ms}}; \boldsymbol{s}_{\mathrm{N}-1}, \boldsymbol{q}_{\mathrm{N}-1}, d, \boldsymbol{p}) - \boldsymbol{s}_{\mathrm{N}} \\ \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}) \end{pmatrix} \quad \text{and} \quad \boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v}, \boldsymbol{p}) := \begin{pmatrix} \tilde{\boldsymbol{c}}(\boldsymbol{s}_0, \hat{\boldsymbol{u}}_0(t_0^{\mathrm{ms}}, \boldsymbol{q}_0), \boldsymbol{p}) \\ \vdots \\ \tilde{\boldsymbol{c}}(\boldsymbol{s}_{\mathrm{N}}, \hat{\boldsymbol{u}}_{\mathrm{N}}(t_{\mathrm{N}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{N}}), \boldsymbol{p}) \\ \tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}) \end{pmatrix},$$

with dimensions $\mathrm{n_{C_{ec}}}$ and $\mathrm{n_{C_{ic}}}$, respectively. Additionally, we introduce slack variables $\boldsymbol{w} \in \mathbb{R}^{\mathrm{n_{C_{ic}}}}$ and collect both constraint functions into one equality constraint function such that

$$\boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}) := \begin{pmatrix} \boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v}, d, \boldsymbol{p}) \\ \boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v}, \boldsymbol{p}) - \boldsymbol{w} \end{pmatrix} = 0 \quad \text{with} \quad \boldsymbol{w} \geq 0,$$

where $\boldsymbol{z} := \begin{pmatrix} \boldsymbol{v}^T & d & \boldsymbol{w}^T \end{pmatrix}^T \in \mathbb{R}^{n_z}$ combines the decision variables in the NLP with dimension $n_z := n_v + 1 + n_{C_{ic}}$. In total, NLP (2.15) can be rewritten in compact form as

$$\min_{\boldsymbol{z}} \quad \boldsymbol{\Phi}(\boldsymbol{s}_N, d, \boldsymbol{\alpha}, \boldsymbol{p}) \tag{2.17a}$$

$$\text{s.t.} \quad 0 = \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}), \tag{2.17b}$$

$$\underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \tag{2.17c}$$

where model parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$ and objective weights $\boldsymbol{\alpha} \in \mathbb{R}^{n_M}$ have fixed values. The upper and lower bounds are set to $\overline{\boldsymbol{z}} := \begin{pmatrix} \overline{\boldsymbol{v}}^T & \overline{d} & \infty \end{pmatrix}^T$ and $\underline{\boldsymbol{z}} := \begin{pmatrix} \underline{\boldsymbol{v}}^T & \underline{d} & 0 \end{pmatrix}^T$, respectively. If no simple bounds on the variables appear, the values of the bounds can be set to $+\infty$ or $-\infty$.

The resulting NLP can then be solved with the methods described in section 1.2. This compact NLP formulation is later used for the lower level in Bilevel Inverse OCPs when we introduce the Direct All-at-Once Approach by Hatz [80] and when we describe our DISIMFAS in chapter 4. Within these methods the model parameters $\boldsymbol{p}$ and objective weights $\boldsymbol{\alpha}$ are identified.

### 2.2.2 Derivative Generation

The NLP (2.15) of the previous subsection, as well as other later in this thesis introduced NLPs as a result of direct approaches applied onto PE Problems and Bilevel Invers OCPs, can be solved by the methods described in section 1.2. All these methods are derivative-based and, hence, accurate and efficient derivative generation is required.

There are various ways to compute derivatives of model functions, e.g. analytical differentiation by hand, symbolic differentiation by computer algebra systems, finite difference approximation, or AD. The latter two variants are briefly discussed in the following. The one-sided finite difference approximation for a function $g(y)$ can be computed by

$$\frac{\mathrm{d}}{\mathrm{d}y} g(y) = \frac{g(y+h) - g(y)}{h} + \mathcal{O}(h),$$

with perturbation $h > 0$. It results in an error of $\mathcal{O}(h)$. For central differences approximations computed as

$$\frac{\mathrm{d}}{\mathrm{d}y} g(y) = \frac{g(y+h) - g(y-h)}{2h} + \mathcal{O}(h^2),$$

an error of $\mathcal{O}(h^2)$ is obtained. The advantage of the finite differences approach is its simple implementation. The function $g(y)$ at hand can be treated as a "black box" and, hence, no further information on the structure is needed. However, the approach might suffer from high computational effort for function evaluations and lack of accuracy. A more sophisticated approach for derivative generation is AD, which can produce exact derivatives up to machine precision. Therein, functions are represented as compositions of elementary operations. Their derivatives are then computed by applying the chain rule systematically. Various variants are available such as *forward* and *reverse modes*, as well as the *propagation of Taylor coefficients* for higher order derivatives. For a comprehensive discussion on AD we refer the reader to the textbook of Griewank and Walther [74].

As a result of the Direct Multiple Shooting Method we also need derivatives of the states $\boldsymbol{x}(t_{i+1}^{ms}; \boldsymbol{s}_i, \boldsymbol{q}_i, d, \boldsymbol{p})$ as a result of the solution of an IVP with respect to multiple shooting variables $\boldsymbol{s}_i$, control parameters $\boldsymbol{q}_i$, duration parameters $\boldsymbol{d}$, and eventually model parameters $\boldsymbol{p}$. These derivatives are also called *sensitivities* as they describe the sensitivity of the trajectory with respect to the initial values and parameters. In the following we give a brief overview on selected approaches and refer to the work of Albersmeyer [5] for a comprehensive introduction.

#### External Numerical Differentiation (END)

The END approach is a method of perturbed trajectories and can be realized by applying the finite differences approach described above after solving the nominal IVP for the original values and once again for slightly perturbed values. Despite its advantage that END is easy to implement, it suffers from a huge drawback: the inconsistency of the computed derivatives. The finite differences approach assumes that the functions at hand are differentiable, which is not the case for general integration procedures because of adaptive components and iterative solvers. Especially if second or higher order derivatives are needed, this approach does not necessarily approximates the exact derivatives to a satisfactory accuracy.

#### Internal Numerical Differentiation (IND)

In the concept of IND invented by Bock [23, 24] the lack of non-smoothness can be overcome. The basic idea is to freeze the adaptive components during integration procedure for the approximation of the nominal trajectory and the perturbed trajectory, and differentiate the adaptively generated discretization scheme. This approach results in the generation of consistent derivatives and in combination with AD the derivatives computed together with the nominal trajectory are exact up to machine precision. In the thesis of Albersmeyer [5] adjoint based algorithms and numerical methods for sensitivity generation based on AD are developed. These methods are implemented in the software package SOLVIND [5] with an interface to ADOL-C by Walther and Griewank [171]. In our work SOLVIND is integrated in the software package PARDYNOPT, see chapter 7, and used for the generation of first- and second-order derivatives of model functions. Additionally, its implementation of an adaptive Backward Differentiation Formulas (BDF) method with an interface to DAESOL-II [5] and the corresponding sensitivity generations using IND is applied up to second-order derivatives.

#### Variational Differential Equation (VDE)

Another approach for the generation of sensitivities is the solution of a VDE system defined on each multiple shooting interval. For the generation of sesitivities with respect to the initial states the equations system is as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{G}(t;\boldsymbol{s}_\mathrm{i},\boldsymbol{q}_\mathrm{i},d,\boldsymbol{p}) = \frac{\partial}{\partial \boldsymbol{x}}\left(d\cdot\boldsymbol{f}(\boldsymbol{x}(t),\hat{\boldsymbol{u}}_\mathrm{i}(t,\boldsymbol{q}_\mathrm{i}),\boldsymbol{p})\right)\cdot\boldsymbol{G}(t;\boldsymbol{s}_\mathrm{i},\boldsymbol{q}_\mathrm{i},d,\boldsymbol{p}),\quad t\in\mathcal{T}^\mathrm{i}, \tag{2.18a}$$

$$\boldsymbol{G}(t_\mathrm{i}^\mathrm{ms};\boldsymbol{s}_\mathrm{i},\boldsymbol{q}_\mathrm{i},d,\boldsymbol{p}) = \boldsymbol{I}_{\mathrm{n_x}}, \tag{2.18b}$$

for $\mathrm{i}=0,\ldots,\mathrm{N}-1$ as it can be shown that

$$\boldsymbol{G}(t;\boldsymbol{s}_\mathrm{i},\boldsymbol{q}_\mathrm{i},d,\boldsymbol{p}) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{s}_\mathrm{i}}\boldsymbol{x}(t^\mathrm{ms};\boldsymbol{s}_\mathrm{i},\boldsymbol{q}_\mathrm{i},d,\boldsymbol{p}).$$

The VDE approach can be realized by applying IND to overcome the inconsistency of the derivative approximations. The VDE system can be integrated together with the initial value problem (2.12). Similar to (2.18) the sesitivities with respect to parameters $\boldsymbol{q}_\mathrm{i}, \boldsymbol{d}$ and $\boldsymbol{p}$ can be computed as well.

## 2.3 Parameter Estimation in Dynamic Systems

In this section a constrained PE problem is considered, where a parameter-dependent dynamic model is fitted to given measurements. We start with the problem formulation, where the arising variables and functions are defined, and then describe a direct solution approach based on the Direct Multiple-Shooting Method introduced earlier in subsection 2.2.1.

### 2.3.1 Problem Formulation

We consider a PE problem, where a weighted least-squares objective is minimized constrained by an underlying parameter-dependent ODE model with boundary conditions and additional state constraints. The objective function comprises measurements $\boldsymbol{\eta} \in \mathbb{R}^{n_m \times n_h}$ observed at specific time points

$$t_s \le t_0^m < t_1^m < \ldots < t_{n_m}^m \le t_f.$$

and the corresponding model response function $\boldsymbol{h} : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_m \times n_h}$ evaluated at these time points. We assume that the measurements $\eta_{nk}$ are afflicted with independent, additive, and normally distributed errors $\varepsilon_{nk}$ with zero mean and variances $\sigma_{nk}^2$. Then for a correct model with true parameters $\boldsymbol{p}^*$ we get

$$\eta_{nk} = h_k(t_n^m, \boldsymbol{x}(t_n^m), \boldsymbol{p}^*) + \varepsilon_{nk}, \qquad n = 0, \ldots, n_m - 1,\ k = 0, \ldots, n_h - 1.$$

Hence, for finding an optimal trajectory $\boldsymbol{x}(t)$ with the corresponding optimal model parameters $\boldsymbol{p}$ which meet the measurement data best, a PE problem can be formulated as follows

$$\min_{\boldsymbol{x},\boldsymbol{p}} \quad \frac{1}{2} \sum_{n=0}^{n_m-1} \sum_{k=0}^{n_h-1} \frac{\left(h_k(t_n^m, \boldsymbol{x}(t_n^m), \boldsymbol{p}) - \eta_{nk}\right)^2}{\sigma_{nk}^2} \tag{2.19a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}, \tag{2.19b}$$

$$0 \le \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}, \tag{2.19c}$$

$$0 = \boldsymbol{r}^{ec}(t_s, \boldsymbol{x}(t_s), t_f, \boldsymbol{x}(t_f), \boldsymbol{p}), \tag{2.19d}$$

$$0 \le \boldsymbol{r}^{ic}(t_s, \boldsymbol{x}(t_s), t_f, \boldsymbol{x}(t_f), \boldsymbol{p}), \tag{2.19e}$$

defined on the fixed time horizon $\mathcal{T} = [t_s, t_f] \subset \mathbb{R}$ with initial time $t_s$ and final time $t_f$, $t_s < t_f$. In the following the remaining variables and sufficiently smooth functions in (2.19) are described:

- differential states denoted by $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n_x}$,
- model parameters denoted by $\boldsymbol{p} \in \mathbb{R}^{n_p}$,
- dynamics described by a system of ODEs (2.19b) with the right-hand side $\boldsymbol{f} : \mathcal{T} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$,
- boundary equality constraints $\boldsymbol{r}^{ec} : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{ec}}$ in (2.19d),
- inequality constraints $\boldsymbol{r}^{ic} : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{ic}}$ in (2.19e),
- state constraints $\boldsymbol{c} : \mathcal{T} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ in (2.19c).

### 2.3.2 An Approach to Solve Parameter Estimation Problems in Dynamic Systems

One approach to solve PE Problems of the form (2.19) is to apply the Direct Multiple Shooting Method described in subsection 2.2.1 for parametrization of the dynamics and discretization of all arising constraints. This results in an NLP as follows:

$$\min_{\boldsymbol{s},\boldsymbol{p}} \quad \frac{1}{2} \sum_{n=0}^{n_m-1} \sum_{k=0}^{n_h-1} \frac{\left(h_k(t_n^m, \boldsymbol{x}(t_n^m; \boldsymbol{s}, \boldsymbol{p}), \boldsymbol{p}) - \eta_{nk}\right)^2}{\sigma_{nk}^2} \tag{2.20a}$$

$$\text{s.t.} \quad 0 = \boldsymbol{x}(t_{i+1}^{ms}; t_i^{ms}, \boldsymbol{s}_i, \boldsymbol{p}) - \boldsymbol{s}_{i+1}, \qquad i = 0, \ldots, N-1, \tag{2.20b}$$

$$0 \le \boldsymbol{c}(t_i^{ms}, \boldsymbol{s}_i, \boldsymbol{p}), \qquad i = 0, \ldots, N, \tag{2.20c}$$

$$0 = \boldsymbol{r}^{ec}(t_0^{ms}, \boldsymbol{s}_0, t_N^{ms}, \boldsymbol{s}_N, \boldsymbol{p}), \tag{2.20d}$$

$$0 \le \boldsymbol{r}^{ic}(t_0^{ms}, \boldsymbol{s}_0, t_N^{ms}, \boldsymbol{s}_N, \boldsymbol{p}). \tag{2.20e}$$

which can then be solved with a tailored Generalized Gauß-Newton method as introduced in subsection 1.2.3. For more details we refer to the works of Bock [25] and Schlöder [150].

## 2.4 Bilevel Inverse Optimal Control Problems

Bilevel optimization of problems comprising an OCP on the lower hierarchy level is related to differential games. Vajda and Isaacs [166] investigated this in their ground-breaking work during the 1950s, which can also be considered as a generalization of bilevel optimization to a function space in the field of finite-dimensional nonlinear mathematical optimization. Detailed introduction to bilevel optimal control can be found in Bard [19], Dempe et al. [43, 46], Shimizu et al. [160], and Lewis et al. [112]. Constraint qualifications and necessary optimality conditions of bilevel problems are discussed in Ye [174], Ye [175], Mehlitz [121], Benita and Mehlitz [20], and Mehlitz and Wachsmuth [122].

In this thesis we deal with a special class of Bilevel OCPs: PE Problems constrained by OCPs, and refer to it as Bilevel Inverse OCPs. In the following, the problem formulation we mainly consider in this thesis is introduced, which incorporates the multi-stage structure from subsection 2.1.2 on the lower level and a least-squares term as objective function on the upper level. Further, an overview of various solution approaches and a brief introduction to the Direct All-at-Once Approach, see [81, 80], are given.

### 2.4.1 Problem Formulation

In this thesis we mostly deal with a hierarchical problem, where the upper level PE problem is constrained by the lower level multi-stage OCP (2.7) with discontinuities from subsection 2.1.2. This problem class differs from the classical PE Problem as introduced in section 2.3, where the constraints model a dynamic process with unknown parameters. In contrast to the classical PE Problem in Bilevel Inverse OCPs the PE Problem is constrained by a dynamic optimization problem. This lower level OCP describes a dynamic process with unknown parameters that optimizes given criteria. The Bilevel Inverse OCP on fixed *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [0,1]$ for each model stage $\mathrm{j} = 1,\dots,\mathrm{n_S}$ can be formulated as follows:

$$\min_{\substack{\boldsymbol{\alpha},\boldsymbol{p},\\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \frac{1}{2}\sum_{\mathrm{j}=1}^{\mathrm{n_S}}\sum_{\mathrm{n}=0}^{\mathrm{n_m^j}-1}\sum_{\mathrm{k}=0}^{\mathrm{n_h^j}-1}\frac{\left(h_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{jn}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{jnk}}\right)^2}{\sigma_{\mathrm{jnk}}^2} \tag{2.21a}$$

s.t.

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \sum_{\mathrm{j}=1}^{\mathrm{n_S}}\boldsymbol{\Phi}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),d_{\mathrm{j}},\boldsymbol{\alpha},\boldsymbol{p}) \tag{2.21b}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_{\mathrm{j}}(t) = d_{\mathrm{j}}\cdot\boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t\in\mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,\dots,\mathrm{n_S}, \tag{2.21c}$$

$$\boldsymbol{x}_{\mathrm{j}+1}(0) = \Delta_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(1),\boldsymbol{p}), \qquad \mathrm{j}=1,\dots,\mathrm{n_S}-1, \tag{2.21d}$$

$$0 \le \boldsymbol{c}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t\in\mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,\dots,\mathrm{n_S}, \tag{2.21e}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{x}_1(0),\boldsymbol{x}_1(1),\dots,\boldsymbol{x}_{\mathrm{n_S}}(1),\boldsymbol{p}), \tag{2.21f}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{x}_1(0),\boldsymbol{x}_1(1),\dots,\boldsymbol{x}_{\mathrm{n_S}}(1),\boldsymbol{p}), \tag{2.21g}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n_M}+\mathrm{n_L}}\alpha_{\mathrm{k}} = 1,\ \boldsymbol{\alpha}\ge 0, \tag{2.21h}$$

$$\underline{\boldsymbol{p}} \le \boldsymbol{p} \le \overline{\boldsymbol{p}}, \tag{2.21i}$$

where the arising model parameters $\boldsymbol{p}\in\mathbb{R}^{\mathrm{n_p}}$ and objective weights $\boldsymbol{\alpha}\in\mathbb{R}^{\mathrm{n_M}+\mathrm{n_L}}$ are determined by fitting the model to given measurement data $\boldsymbol{\eta}\in\mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}}}$ with the dimension $\mathrm{n}_{\boldsymbol{\eta}} := \sum_{\mathrm{j}=1}^{\mathrm{n_S}}\left(\mathrm{n_m^j}\cdot\mathrm{n_h^j}\right)$. To avoid redundant solutions, the constraint (2.21h) is added to the Bilevel Inverse OCP. Further, bounds on the model parameters are included in (2.21i). In the following we describe the upper level objective function and refer the reader to subsection 2.1.2 for a detailed introduction to the lower level multi-stage OCP and all defined quantities.

The least-squares objective function (2.21a) comprises measurements $\boldsymbol{\eta} \in \mathbb{R}^{n_\eta}$ observed at specific time points. After time transformations to fixed *normalized* time horizons $\mathcal{T}^j := [0,1]$ for each model stage these time points are defined as

$$0 \le t^m_{j,0} < t^m_{j,1} < \ldots < t^m_{j,n^j_m-1} \le 1, \quad j = 1,\ldots,n_S.$$

The corresponding model response $\boldsymbol{h}_j : \mathbb{R}^{n_{xj}} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n^j_m \times n^j_h}$ is evaluated at these time points. Similar to the classical PE Problem we assume that the measurements $\eta_{jnk}$ are afflicted with independent, additive, and normally distributed errors $\varepsilon_{jnk}$ with zero mean and variances $\sigma^2_{jnk}$. Then for a correct model with true parameters $\boldsymbol{p}^*$ we get

$$\eta_{jnk} = h_{jk}(\boldsymbol{x}_j(t^m_{jn}), \boldsymbol{p}^*) + \varepsilon_{jnk}, \qquad j = 1,\ldots,n_S,\ n = 0,\ldots,n^j_m - 1,\ k = 0,\ldots,n^j_h - 1.$$

### 2.4.2 Solution Approaches for Bilevel Inverse Optimal Control Problems

The bilevel optimization problem we deal with can be solved in various ways. Here, we distinguish between *simultaneous* and *bilevel approaches*. Clever and Mombaur [37] presented a direct inverse optimal control framework which was first investigated by Mombaur et al. [130] using the latter of these two approaches where the bilevel structure of the problem was exploited. In this approach in each iteration of the upper level parameter estimation problem, the lower level OCP has to be solved. A derivative free method was used for the upper level and a direct structure exploiting SQP method solved the lower level OCP in each iteration using MUSCOD-II, a software package developed by Leineweber [108]. With this inverse optimal control framework, optimality criteria in whole-body human walking were identified for seven different subjects.

On the other side, Bilevel Inverse OCPs can be solved using simultaneous approaches. These methods mainly rely on the substitution of the lower level OCP by its optimality conditions. The term *simultaneous* refers to the fact that in these approaches the upper and lower level problems are solved together at the same time, where the optimality conditions of the lower level OCP serve as a constraint in the upper level PE problem. Knauer [104], and Knauer and Büskens [105] proposed a simultaneous approach for handling an infinite-dimensional bilevel optimization problem by replacing the lower level OCP by its first order optimality conditions based on Pontryagin's Maximum Principle [139]. In their work, inequality constraints were not considered. The resulting one-level problem was then solved by a direct method using special SQP. For discretization of the nonlinear discontinuous boundary value problem collocation was considered. Alternatively, the one-level problem was solved - again based on Pontryagin's maximum principle - using an indirect method. However, both approaches based on Pontryagin's Maximum Principle on the upper level are not easily applicable for modeling human locomotion where a highly non-smooth multi-level OCP with mixed control-state constraints has to be solved.

Another possible approach, which again falls into the category of simultaneous approaches, is to first discretize the problem before reformulation of the resulting lower level problem by its KKT optimality conditions. Albrecht et al. [8, 9, 6] developed methods for solving parameter estimation problems comprising OCPs as constraints. They also used a simultaneous approach and reformulated the OCP, after a collocation-based discretization, by its KKT conditions. Due to the omission of inequality constraints the resulting nonlinear problem can be solved with an interior-point method. This approach has been successfully used to identify optimization criteria for arm motions [9] or human navigation [8]. In contrast, if inequality constraints are present in the discretized lower level problem the replacement by its KKT conditions results in an MPCC - a challenging class of NLPs, see discussion and solution approaches in section 1.3. In the context of human locomotion Albrecht and Ulbrich [7] compared various regularization and lifting strategies for the treatment of arising complementarity constraints. In this work a simple dynamic model was used, which navigates from start to final position. A more complex dynamic model of a CP patient's gait was considered in the work of

Hatz [80], where the Direct All-at-Once Approach was successfully applied in a first analysis. Therein, the lower level OCP is reformulated by its KKT optimality conditions, after a multiple shooting based parametrization of the states and an adequate discretization of the controls via the Direct Multiple Shooting Method developed by Bock and Plitt [26, 138]. In general, when applying simultaneous approaches, the resulting one-level NLP and the original problem are not necessarily mathematically equivalent, see, e.g., [44, 45] for a detailed discussion on this topic.

In this thesis, we use the Direct All-at-Once Approach as a basis for the development of our DISIMFAS in chapter 4 due to its lower computational effort compared to bilevel approaches and its promising results for a collection of benchmark problems in [80].

### 2.4.3 The Direct All-at-Once Approach

The DISIMFAS developed in this thesis mainly relies on the Direct All-at-Once Approach with a direct method for the treatment of the lower level OCP by Hatz et al. [81] and Hatz [80]. In this section we give a brief overview of this efficient and simultaneous approach to solve Bilevel Inverse OCPs of the form (2.21) as stated in subsection 2.4.1 and discuss the challenges which appear. Although, according to [81] the lower level OCP can be treated with an indirect method in the Direct All-at-Once Approach as well, in [80] the *Indirect All-at-Once Approach* relates to this particular case. When we use the phrase *Direct All-at-Once Approach*, we only refer to the simultaneous approach with a direct method on the lower level. The main steps of the Direct All-at-Once Approach are as follows. First the lower level multi-stage OCP is treated with a direct method according to the Direct Multiple Shooting Method described earlier in subsection 2.2.1. This results in a finite-dimensional nonlinear bilevel program with a structured NLP on the lower level similar to the single-stage counterpart in equation (2.15). In a second step, this lower level NLP is then replaced by its necessary optimality conditions which transforms the bilevel program to a one-level NLP of special class: an MPCC. Finally, the resulting structured NLP is solved with a tailored Generalized Gauss Newton method. In the following we briefly introduce the steps described above for a Bilevel Inverse OCP with a single-stage OCP on the lower level as the one in the problem formulation (2.8) used in subsection 2.2.1. We formulate the Bilevel Inverse OCP on a fixed normalized time horizon $t \in [0,1]$ with free duration parameter $d \in \mathbb{R}$ as follows

$$\min_{\substack{\boldsymbol{\alpha},\boldsymbol{p},\\ \boldsymbol{x},\boldsymbol{u},d}} \quad \frac{1}{2}\sum_{n=0}^{n_m-1}\sum_{k=0}^{n_h-1}\frac{\left(h_k(\boldsymbol{x}(t_n^m),\boldsymbol{p})-\eta_{nk}\right)^2}{\sigma_{nk}^2} \tag{2.22a}$$

s.t.

$$\min_{\boldsymbol{x},\boldsymbol{u},d} \quad \sum_{k=1}^{n_M}\alpha_k\cdot\boldsymbol{\phi}_k^M(\boldsymbol{x}(1),d,\boldsymbol{p}) \tag{2.22b}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = d\cdot\boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \qquad t\in[0,1], \tag{2.22c}$$

$$0 \le \boldsymbol{c}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \qquad t\in[0,1], \tag{2.22d}$$

$$0 = \boldsymbol{r}^{ec}(\boldsymbol{x}(0),\boldsymbol{x}(1),\boldsymbol{p}), \tag{2.22e}$$

$$0 \le \boldsymbol{r}^{ic}(\boldsymbol{x}(0),\boldsymbol{x}(1),\boldsymbol{p}), \tag{2.22f}$$

$$\sum_{k=1}^{n_M}\alpha_k = 1, \boldsymbol{\alpha}\ge 0, \tag{2.22g}$$

$$\underline{\boldsymbol{p}} \le \boldsymbol{p} \le \overline{\boldsymbol{p}}. \tag{2.22h}$$

All arising variables and functions are introduced in subsection 2.4.1 where $n_S$ is set to 1 and the stage index j as well as the transition condition (2.21d) are omitted. Furthermore, only Mayer terms are considered. Similar to subsection 2.2.1 the single-stage formulation facilitates the notation, on the one hand, and captures the main characteristics of the mathematical method, on the other hand. Situations where the multi-stage formulation may affect the described method are rare and mostly covered by incorporating the transition condition (2.21d)

in our discussions. We briefly introduce the Direct All-at-Once Approach and follow the presentation in [80]. For a more elaborate, detailed description the interested reader is advised to the contributions of Hatz et al. [81, 80].

#### Step 1: Apply the Direct Multiple Shooting Method on the Lower Level OCP

The lower level single-stage OCP is treated with the Direct Multiple Shooting Method according to subsection 2.2.1, where discretization of control functions, parametrization of differential states and discretization of constraints lead to NLP (2.15). This structured NLP replaces the lower level OCP in Bilevel Inverse OCP (2.22) such that we can formulate a finite-dimensional bilevel nonlinear problem of the following form

$$\min_{\substack{\boldsymbol{\alpha},\boldsymbol{p},\\ \boldsymbol{s},\boldsymbol{q},d}} \quad \frac{1}{2}\sum_{\mathrm{n}=0}^{\mathrm{n_m}-1}\sum_{\mathrm{k}=0}^{\mathrm{n_h}-1}\frac{\left(h_{\mathrm{k}}(\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{nk}}^2} \tag{2.23a}$$

s.t.

$$\min_{\boldsymbol{s},\boldsymbol{q},d} \quad \boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{\alpha},\boldsymbol{p}) \tag{2.23b}$$

$$\text{s.t.} \quad 0=\boldsymbol{x}(t_{\mathrm{i+1}}^{\mathrm{ms}};\boldsymbol{s}_{\mathrm{i}},\boldsymbol{q}_{\mathrm{i}},d,\boldsymbol{p})-\boldsymbol{s}_{\mathrm{i+1}}, \qquad \mathrm{i}=0,\dots,\mathrm{N}-1, \tag{2.23c}$$

$$0\le\boldsymbol{c}(\boldsymbol{s}_{\mathrm{i}},\hat{\boldsymbol{u}}_{\mathrm{i}}(t_{\mathrm{i}}^{\mathrm{ms}},\boldsymbol{q}_{\mathrm{i}}),\boldsymbol{p}), \qquad \mathrm{i}=0,\dots,\mathrm{N}, \tag{2.23d}$$

$$0=\boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0,\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p}), \tag{2.23e}$$

$$0\le\boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{s}_0,\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p}). \tag{2.23f}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n_M}}\alpha_{\mathrm{k}}=1,\boldsymbol{\alpha}\ge 0, \tag{2.23g}$$

$$\underline{\boldsymbol{p}}\le\boldsymbol{p}\le\overline{\boldsymbol{p}}. \tag{2.23h}$$

The least-squares objective (2.23a) depends on the term $\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}})$, which we shortly explain in the following. Let the trajectory $\boldsymbol{x}(t;\boldsymbol{s}_{\mathrm{i}},\boldsymbol{q}_{\mathrm{i}},d,\boldsymbol{p})$ be the solution of an IVP (2.12) on multiple shooting interval $\mathcal{T}^{\mathrm{i}}=[t_{\mathrm{i}}^{\mathrm{ms}},t_{\mathrm{i+1}}^{\mathrm{ms}}]$ as described in subsection 2.2.1. Then the term $\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}}):=\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}};\boldsymbol{s}_{\mathrm{i}},\boldsymbol{q}_{\mathrm{i}},d,\boldsymbol{p})$ means that this trajectory is evaluated at a specific time point $t_{\mathrm{n}}^{\mathrm{m}}\in\mathcal{T}^{\mathrm{i}}$ which lies in this particular multiple shooting interval. At this point, please note that in a multi-stage environment in addition to matching conditions (2.23c) also transition conditions between model stages would enter NLP (2.23) with equation (2.14). These transition constraints are enforced between the last multiple shooting node of one model stage and the first node of its successor model stage as described in subsection 2.2.1.

In the following we use the more compact representation of NLP (2.17) on the lower level (with an equivalent reformulation of the bounds), where we combine all constraints into one constraint function $\boldsymbol{C}(\cdot)$ by introducing slack variables $\boldsymbol{w}\in\mathbb{R}^{\mathrm{n_{Cic}}}$ such that the NLP can be written in the form

$$\min_{\boldsymbol{\alpha},\boldsymbol{p},\boldsymbol{z}} \quad \frac{1}{2}\sum_{\mathrm{n}=0}^{\mathrm{n_m}-1}\sum_{\mathrm{k}=0}^{\mathrm{n_h}-1}\frac{\left(h_{\mathrm{k}}(\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{nk}}^2} \tag{2.24a}$$

s.t.

$$\min_{\boldsymbol{z}} \quad \boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{\alpha},\boldsymbol{p}) \tag{2.24b}$$

$$\text{s.t.} \quad 0=\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p}), \tag{2.24c}$$

$$0\le\overline{\boldsymbol{z}}-\boldsymbol{z}, \tag{2.24d}$$

$$0\le\boldsymbol{z}-\underline{\boldsymbol{z}}, \tag{2.24e}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n_M}}\alpha_{\mathrm{k}}=1,\boldsymbol{\alpha}\ge 0, \tag{2.24f}$$

$$\underline{\boldsymbol{p}}\le\boldsymbol{p}\le\overline{\boldsymbol{p}}, \tag{2.24g}$$

with simple bounds on the decision variables $\boldsymbol{z} := \begin{pmatrix} \boldsymbol{v}^T & d & \boldsymbol{w}^T \end{pmatrix}^T$ as the only appearing inequality constraints on the lower level, where the variables are combined in the vectors $\boldsymbol{v}_\mathrm{i} := \begin{pmatrix} \boldsymbol{s}_\mathrm{i}^T & \boldsymbol{q}_\mathrm{i}^T \end{pmatrix}^T$ and $\boldsymbol{v} := \begin{pmatrix} \boldsymbol{v}_0^T & \dots & \boldsymbol{v}_\mathrm{N}^T \end{pmatrix}^T$.

#### Step 2: Replace the Lower Level NLP by its KKT Conditions

When replacing the parametrized and discretized lower level optimal control problem by its optimality conditions one should be aware of the fact that the original problem and the resulting one-level NLP, which is an MPCC as introduced in the following, are not necessarily mathematically equivalent, see, e.g. [44, 45], for a detailed discussion on this topic. We start with the Lagrange function in compact notation such that we have

$$\mathcal{L}(\boldsymbol{Z}) := \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \boldsymbol{\lambda}^T \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}) - \boldsymbol{\mu}_{\overline{z}}^T \left(\overline{\boldsymbol{z}} - \boldsymbol{z}\right) - \boldsymbol{\mu}_{\underline{z}}^T \left(\boldsymbol{z} - \underline{\boldsymbol{z}}\right),$$

where all variables are combined into vector $\boldsymbol{Z} := \begin{pmatrix} \boldsymbol{z}^T & \boldsymbol{\lambda}^T & \boldsymbol{\mu}^T & \boldsymbol{\alpha}^T & \boldsymbol{p}^T \end{pmatrix}^T$ with the newly introduced Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{\mathrm{n_{Cec}+C_{ic}}}$ related to equality constraints (2.24c) and $\boldsymbol{\mu} := \begin{pmatrix} \boldsymbol{\mu}_{\overline{z}}^T & \boldsymbol{\mu}_{\underline{z}}^T \end{pmatrix}^T$ with $\boldsymbol{\mu}_{\overline{z}} \in \mathbb{R}^{\mathrm{n_z}}$ and $\boldsymbol{\mu}_{\underline{z}} \in \mathbb{R}^{\mathrm{n_z}}$ related to simple bounds (2.24d) and (2.24e), respectively.

According to theorem 1.8 and the assumption that LICQ holds at the solution, the KKT conditions of the lower level problem can be formulated as

Stationarity:

$$0 = \nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{Z}), \tag{2.25a}$$

Primal Feasibility:

$$0 = \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}), \tag{2.25b}$$

$$0 \leq \overline{\boldsymbol{z}} - \boldsymbol{z}, \tag{2.25c}$$

$$0 \leq \boldsymbol{z} - \underline{\boldsymbol{z}}, \tag{2.25d}$$

Dual Feasibility:

$$0 \leq \boldsymbol{\mu}, \tag{2.25e}$$

Complementarity:

$$0 = \boldsymbol{\mu}_{\overline{z}} \circ \left(\overline{\boldsymbol{z}} - \boldsymbol{z}\right), \tag{2.25f}$$

$$0 = \boldsymbol{\mu}_{\underline{z}} \circ \left(\boldsymbol{z} - \underline{\boldsymbol{z}}\right), \tag{2.25g}$$

where $\circ$ denotes the element-wise vector multiplication (Hadamard product), and the gradient of the Lagrangian with respect to the lower level variables is defined by

$$\nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{Z}) := \nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p})^T \boldsymbol{\lambda} - \nabla_{\boldsymbol{z}} \left(\overline{\boldsymbol{z}} - \boldsymbol{z}\right)^T \boldsymbol{\mu}_{\overline{z}} - \nabla_{\boldsymbol{z}} \left(\boldsymbol{z} - \underline{\boldsymbol{z}}\right)^T \boldsymbol{\mu}_{\underline{z}} \tag{2.26a}$$

$$= \nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p})^T \boldsymbol{\lambda} + \boldsymbol{\mu}_{\overline{z}} - \boldsymbol{\mu}_{\underline{z}}. \tag{2.26b}$$

Replacing the discretized and parametrized lower level problem by conditions (2.25) leads to the following structured one-level NLP

$$\min_{Z} \quad \frac{1}{2}\sum_{n=0}^{n_m-1}\sum_{k=0}^{n_h-1}\frac{\left(h_k(\boldsymbol{x}(t_n^m),\boldsymbol{p})-\eta_{nk}\right)^2}{\sigma_{nk}^2} \tag{2.27a}$$

$$\text{s.t.} \quad 0=\nabla_{\boldsymbol{z}}\mathcal{L}(\boldsymbol{Z}), \tag{2.27b}$$

$$0=\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p}), \tag{2.27c}$$

$$0\le\overline{\boldsymbol{z}}-\boldsymbol{z}, \tag{2.27d}$$

$$0\le\boldsymbol{z}-\underline{\boldsymbol{z}}, \tag{2.27e}$$

$$0\le\boldsymbol{\mu}, \tag{2.27f}$$

$$0=\boldsymbol{\mu}_{\overline{z}}\circ\left(\overline{\boldsymbol{z}}-\boldsymbol{z}\right), \tag{2.27g}$$

$$0=\boldsymbol{\mu}_{\underline{z}}\circ\left(\boldsymbol{z}-\underline{\boldsymbol{z}}\right), \tag{2.27h}$$

$$\sum_{k=1}^{n_M}\alpha_k=1,\boldsymbol{\alpha}\ge 0, \tag{2.27i}$$

$$\underline{\boldsymbol{p}}\le\boldsymbol{p}\le\overline{\boldsymbol{p}}, \tag{2.27j}$$

where vector $\boldsymbol{Z}:=\begin{pmatrix}\boldsymbol{z}^T & \boldsymbol{\lambda}^T & \boldsymbol{\mu}^T & \boldsymbol{\alpha}^T & \boldsymbol{p}^T\end{pmatrix}^T$ includes all decision variables. $\nabla_{\boldsymbol{z}}\boldsymbol{\Phi}(\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{p})$ and $\nabla_{\boldsymbol{z}}\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p})$ enter the gradient of the Lagrangian in (2.26). Its definitions and the specific structure are given in chapter 4 when we describe our DISIMFAS in detail, where these quantities are also needed and the occuring structures have to be exploited for an efficient implementation.

Problem (2.27) belongs to the special class of MPCCs. In section 1.3 we give a brief overview on general MPCCs and its solution approaches. As mentioned there, the numerical treatment of these kinds of problems is challenging and using general NLP solvers may lead to degenerated quadratic programs due to the lack of valid constraint qualifications, e.g. MFCQ and LICQ. Hence, an appropriate handling of complementarity constraints (2.27g) and (2.27h) is necessary as, e.g., the lifting approach developed by Hatz [80]. However, an important observation in a first analysis of a CP patient's gait in [80, Chapter 15] is that this regularization approach, in which the complementarity constraint is expressed as a nonlinear complementarity constraint function proved inappropriate for solving this highly non-smooth large-scale NLP. Because of this an exact penalty approach as described in Benson et al. [22] is used.

In chapter 4 we present the DISIMFAS as a solution approach for Bilevel Inverse OCPs to overcome the difficulty of arising MPCCs.

#### Step 3: Solve the One-Level NLP with the Generalized Gauß-Newton Method

The one-level NLP (2.27) can be solved with the Generalized Gauß-Newton Method as introduced in subsection 1.2.3 and in subsection 2.3.2 for PE problems. We refer the interested reader to the contribution of Hatz in [80, Chapter 5] for a more detailed description and a presentation of an efficient structure exploitation. An implementation of the proposed method with several approaches for handling the complementarity constraints, e.g. the Lifting approach, can be found in the software package PARAOCP, [80]. It is realized within a SQP framework based on the NLP solver FILTERSQP[59], where the Hessian of the Lagrangian of NLP (2.27) is approximated appropriately.

# Chapter 3

# Human Locomotion and Cerebral Palsy

In this chapter we introduce how human locomotion can be described using optimal control and start the first section with dynamics of rigid multibody systems before we give a general multi-stage OCP formulation for the human gait. Furthermore, as this thesis is concerned with gaits of CP patients in the second section the disease *Cerebral Palsy* and its classification are described. The chapter is finalized with a brief overview on typical characteristics of CP gait.

## 3.1 Human Locomotion as an OCP

The human body and its locomotion have been intensively investigated throughout various disciplines, such as biology, physiology, neuroscience, robotics, biomechanics, and others. In this section we focus on rigid multibody systems and its dynamics, and provide an appropriate OCP formulation to describe the human gait. We follow the work of Mombaur [128], Felis [52], and Hatz [80], where the first part is mainly based on theoretical foundations as described in the book of Featherstone [51]. Various other textbooks covering the dynamics of rigid multibody systems are available, such as the ones by Jain [94], Shabana [157], Khalil [100], and Craig [39].

### 3.1.1 Dynamics of Rigid Multibody Systems

For studying human locomotion there are various ways to model the body of a specific subject. One can use very complex skeletal muscle models as in the work of Millard et al. [125] and Delp et al. [41], or stick to basic inverted pendulum models as in the simplest walking model of Garcia et al. [66]. The choice of the model depends on the specific application. In this work we focus on rigid multibody systems as we use the results of Hatz [80] as a basis. We assume that single segments of the body are connected via joints, where the torques are acting directly at the joints. The motion of such a multibody model can be described by considering a minimal number of coordinates or *DOFs*, denoted with $\mathrm{n_{dofs}}$. A typical choice are position and orientation of one base, such as for example the pelvis, and internal DOFs at the joints. In the following, these minimal coordinates or *generalized coordinates* are defined by $\boldsymbol{q}(t) \in \mathbb{R}^{\mathrm{n_{dofs}}}$. The corresponding *generalized velocities, accelerations* and *forces* are denoted in the following by $\dot{\boldsymbol{q}}(t)$, $\ddot{\boldsymbol{q}}(t)$ and $\boldsymbol{\tau}(t) \in \mathbb{R}^{\mathrm{n_{dofs}}}$, respectively. All quantities together with given model parameters combined in the vector $\boldsymbol{p} \in \mathbb{R}^{\mathrm{n_p}}$ describe the system and its movement at a specific time point $t \in \mathbb{R}$. Multibody systems with one base segment belong to so-called *underactuated* systems. In these systems the number of actuated DOFs is smaller than the number of DOFs, $\mathrm{n_{act}} < \mathrm{n_{dofs}}$. The generalized forces of these types of systems can be written as

$$\boldsymbol{\tau}(t) = \begin{pmatrix} 0 \\ \boldsymbol{\tau}^{\mathrm{act}}(t) \end{pmatrix},$$

where $\boldsymbol{\tau}^{\mathrm{act}}(t) \in \mathbb{R}^{\mathrm{n_{act}}}$ are the actuated generalized forces acting on the actuated joints. In the following we omit the time-dependency.

Our focus lies in modeling the human gait which can be described by a periodic repetition of gait cycles. Each of these gait cycles consists of two steps with almost the same start and end posture. For one gait cycle we consider a sequence of different phases, where in each phase at least one point of the multibody model is in

contact witch the ground. If only one foot is in contact with the ground it is called *single support phase*, if both feet are in contact it is a *double support phase*. Depending on the used contact model for the foot both phases can be divided in even more subphases. With the previously introduced quantities, the generalized coordinates $\boldsymbol{q}$, their corresponding generalized velocities $\dot{\boldsymbol{q}}$, generalized accelerations $\ddot{\boldsymbol{q}}$, generalized forces $\boldsymbol{\tau}$, and model parameters $\boldsymbol{p}$, the dynamics of rigid multibody systems can be described by the following equations of motion

$$\boldsymbol{H}(\boldsymbol{q},\boldsymbol{p})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = \boldsymbol{\tau}, \tag{3.1}$$

where $\boldsymbol{H}(\boldsymbol{q},\boldsymbol{p}) \in \mathbb{R}^{n_{\text{dofs}} \times n_{\text{dofs}}}$ is a symmetric *joint space inertia matrix*, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) \in \mathbb{R}^{n_{\text{dofs}}}$ a *generalized bias force* which includes additional forces such as *Coriolis, gravitational* or *centrifugal force*. If we assume that the mass of each body or segment of the rigid multibody system is non-zero, the inertia matrix $\boldsymbol{H}(\boldsymbol{q},\boldsymbol{p})$ is positive definite and regular. The equations of motion from (3.1) describe the dynamics of a multibody system without external contacts. Considering the human gait, in each phase of the whole gait cycle we have at least one contact point interacting with the ground. Hence, we have to use equations of motion with external contacts to describe the locomotion. In our studies, we focus on so-called *holonomic scleronomic constraints without friction*, see, e.g., [51] for a detailed explanation, which can be written as

$$\boldsymbol{g}(\boldsymbol{q},\boldsymbol{p}) = 0, \tag{3.2}$$

and fix the points of the body to the ground which are part of the contact model in one specific phase of the whole gait cycle. For each of the different phases with varying contact models the dynamics of a rigid multibody system can be expressed by its equations of motion with external contacts

$$\boldsymbol{H}(\boldsymbol{q},\boldsymbol{p})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = \boldsymbol{\tau} + \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})^T\boldsymbol{\lambda}, \tag{3.3a}$$

$$\boldsymbol{g}(\boldsymbol{q},\boldsymbol{p}) = 0, \tag{3.3b}$$

where the so-called *contact Jacobian* is defined by

$$\boldsymbol{G}(\boldsymbol{q},\boldsymbol{p}) := \frac{\partial \boldsymbol{g}(\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{q}},$$

and the *contact force* denoted by $\boldsymbol{\lambda}$. The equation system (3.3) is a DAE system of index 3. It can be reduced to a DAE system of index 1 in the way as described in the following. We define the so-called *contact Hessian* by

$$\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) := -\dot{\boldsymbol{G}}(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}},$$

and differentiate equation (3.3b) twice with respect to time $t$.

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{g}(\boldsymbol{q},\boldsymbol{p}) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}}\right)\\ &= \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{G}}(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}}\\ &= \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})\ddot{\boldsymbol{q}} - \boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p})\end{aligned} \tag{3.4}$$

Setting equation (3.4) to 0 we obtain from (3.3)

$$\boldsymbol{H}(\boldsymbol{q},\boldsymbol{p})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = \boldsymbol{\tau} + \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})^T\boldsymbol{\lambda}, \tag{3.5a}$$

$$\boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})\ddot{\boldsymbol{q}} - \boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = 0. \tag{3.5b}$$

If we ensure that equation (3.2) and $\boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}} = 0$ hold, the equation systems (3.3) and (3.5) are equivalent. Because (3.4) is set to 0 both conditions only have to be ensured at the beginning of the contact. After reformu-

lation we obtain a linear equation system

$$\begin{pmatrix} \boldsymbol{H}(\boldsymbol{q},\boldsymbol{p}) & \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})^T \\ \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}} \\ -\boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) \\ \boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) \end{pmatrix}, \tag{3.6}$$

which can be solved uniquely if the constraints $\boldsymbol{g}(\boldsymbol{q},\boldsymbol{p})$ are not redundant and, hence, the constraint Jacobian $\boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})$ has full rank.

#### Collision Impacts

When the rigid multibody system gets in contact with the environment high forces occur. This process can be described by a collision model. If we assume that the contact happens instantaneously with a discontinuous change in the generalized velocities before and after contact we can formulate the following linear equation system

$$\begin{pmatrix} \boldsymbol{H}(\boldsymbol{q},\boldsymbol{p}) & \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})^T \\ \boldsymbol{G}(\boldsymbol{q},\boldsymbol{p}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{q}}^+ \\ -\boldsymbol{\Lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H}(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}}^- \\ -e\boldsymbol{G}(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}}^- \end{pmatrix}, \tag{3.7}$$

with a *restitution parameter* $e \in [0,1]$, and the *contact impulse* $\boldsymbol{\Lambda}$. The velocity before contact is denoted by $\dot{\boldsymbol{q}}^-$ and the velocity after contact by $\dot{\boldsymbol{q}}^+$. With a restitution parameter of $e = 1$ a perfectly elastic collision can be modeled, whereas for a parameter $e = 0$ the collision is perfectly inelastic. For all human locomotion models described in this thesis we assume, that the contact velocity is 0 at collision, which is equivalent to a restitution parameter $e = 0$ and, hence, a perfectly inelastic collision.

There exist various software tools to set up the equations of motion, e.g. the HUMANS toolbox [1] or the *Rigid-Body Dynamics Library*, RBDL [53]. In the latter software efficient algorithms, such as the *Recursive Newton-Euler Algorithm*, the *Composite Rigid-Body Algorithm*, and the famous *Articulated Body Algorithm*, are implemented as described in the thesis of Felis [52] on the basis of Featherstone [51]. Because of its proven efficiency in many applications including optimal control, e.g. in [54, 126], or bilevel inverse optimal control, e.g. in [49], RBDL is also used to evaluate the dynamics in the gait model for a CP patient, see chapter 6.

### 3.1.2 A General Multi-Stage OCP Formulation

The human gait model for a rigid multibody system can be interpreted as a predefined consecutive order of different phases and the transitions between. These phases incorporate the dynamics described in subsection 3.1.1 by (3.5) with its characteristic constraint sets in each phase. Transitions between these phases have to be defined and occuring collisions, for instance at touch-down of the foot with the ground, can be evaluated by solving the linear equation system (3.7). In the field of biomechanics and robotics "[...] it is a common assumption that optimization is a guiding principle" of human locomotion, [131]. Hence, in the following we interpret the human gait as optimal with respect to a well-chosen but unknown combination of different optimization criteria. With this assumption we can formulate a multi-stage OCP as introduced in subsection 2.1.2. For convenience, we state it here on fixed and *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [t_{\mathrm{s}}^{\mathrm{j}}, t_{\mathrm{f}}^{\mathrm{j}}] = [0,1]$ as follows

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \sum_{\mathrm{j}=1}^{\mathrm{n_S}} \boldsymbol{\Phi}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), d_{\mathrm{j}}, \boldsymbol{\alpha}, \boldsymbol{p}) \tag{3.8a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_{\mathrm{j}}(t) = d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j} = 1,\dots,\mathrm{n_S}, \tag{3.8b}$$

$$\boldsymbol{x}_{\mathrm{j}+1}(t_{\mathrm{s}}^{\mathrm{j}+1}) = \Delta_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{f}}^{\mathrm{j}}), \boldsymbol{p}), \qquad \mathrm{j} = 1,\dots,\mathrm{n_S}-1, \tag{3.8c}$$

$$0 \le \boldsymbol{c}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j} = 1,\dots,\mathrm{n_S}, \tag{3.8d}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{x}_1(t_{\mathrm{s}}^1), \boldsymbol{x}_1(t_{\mathrm{f}}^1), \dots, \boldsymbol{x}_{\mathrm{n_S}}(t_{\mathrm{f}}^{\mathrm{n_S}}), \boldsymbol{p}), \tag{3.8e}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{x}_1(t_{\mathrm{s}}^1), \boldsymbol{x}_1(t_{\mathrm{f}}^1), \dots, \boldsymbol{x}_{\mathrm{n_S}}(t_{\mathrm{f}}^{\mathrm{n_S}}), \boldsymbol{p}), \tag{3.8f}$$

with stage durations $d_j$ combined in the time-independent control parameter vector $\boldsymbol{d} = \begin{pmatrix} d_1 & \dots & d_{n_S} \end{pmatrix}^T$, and stage transition time before and after transition, $t_f^j = 1$ and $t_s^{j+1} = 0$, respectively, for each model stage $j = 1,\dots,n_S$ as defined in subsection 2.1.2 together with the remaining variables and functions. In the following we give a more detailed explanation of the quantities that occur in (3.8) when describing a human gait including phasewise dynamics with jumps.

### Optimization Variables

For setting up an OCP as in (3.8) in addition to the duration parameters $d_j$ of each model stage, $j = 1,\dots,n_S$, differential states $\boldsymbol{x}_j$ and control functions $\boldsymbol{u}_j$ can be defined as

$$\boldsymbol{x}_j(t) := \begin{pmatrix} \boldsymbol{q}_j(t) \\ \dot{\boldsymbol{q}}_j(t) \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{u}(t) := \boldsymbol{\tau}_j^{act}(t), \tag{3.9}$$

where for each model stage the generalized coordinates $\boldsymbol{q}_j(t)$ and the corresponding generalized velocities $\dot{\boldsymbol{q}}_j(t)$ are combined in the differential states and the generalized actuated torques $\boldsymbol{\tau}_j^{act}(t)$ used as control functions. To receive a more complex model of the actuated torques which include active and passive components, $\boldsymbol{\tau}_j^a$ and $\boldsymbol{\tau}_j^p$, respectively,

$$\boldsymbol{\tau}_j^{act}(t) = \boldsymbol{\tau}_j^{act}(\boldsymbol{\tau}_j^a(t), \boldsymbol{\tau}_j^p(t)) \tag{3.10}$$

the active part $\boldsymbol{\tau}_j^a$ can be taken as controls instead of using the total actuated torques $\boldsymbol{\tau}_j^{act}$. Another choice is for instance to add these active torques $\boldsymbol{\tau}_j^a$ to the differential states and use their time derivatives $\dot{\boldsymbol{\tau}}_j^a$ as control functions,

$$\boldsymbol{x}_j(t) := \begin{pmatrix} \boldsymbol{q}_j(t) \\ \dot{\boldsymbol{q}}_j(t) \\ \boldsymbol{\tau}_j^a(t) \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{u}(t) := \dot{\boldsymbol{\tau}}_j^a(t), \tag{3.11}$$

which is done in chapter 6 for the patient-specific CP gait model to obtain more smooth human-like motions. Nevertheless, for a clearer notation we choose representation (3.9) in the following descriptions in this chapter, as it is also used in the *basic walker* gait model of section 5.2 .

### Objective Function

As introduced at the beginning of this subsection 3.1.2 we interpret the human gait as optimal with respect to certain optimization criteria. The objective function is of Bolza-type and represented by

$$\begin{aligned} \boldsymbol{\Phi}_j(\boldsymbol{x}_j(t), \boldsymbol{u}_j(t), d_j, \boldsymbol{\alpha}, \boldsymbol{p}) = &\sum_{k=1}^{n_M} \alpha_k \cdot \boldsymbol{\phi}_{jk}^M(\boldsymbol{x}_j(t_f^j), d_j, \boldsymbol{p}) \\ &+ \sum_{k=n_M+1}^{n_M+n_L} \alpha_k \cdot \int_{t_s^j}^{t_f^j} d_j \cdot \boldsymbol{\phi}_{jk}^L(\boldsymbol{x}_j(t), \boldsymbol{u}_j(t), \boldsymbol{p}) \mathrm{d}t. \end{aligned}$$

Optimization criteria might include criteria like energy consumption, comfort or stability. If we are interested in meeting given measurements, these criteria have to be chosen in such a way that with an selected combination of $n_M + n_L$ criteria and individual weights $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \dots & \alpha_{n_M+n_L} \end{pmatrix}^T$ the gait of the person can be reproduced within an acceptable accuracy. The identification of the usually unknown quantities $\boldsymbol{\alpha}$ and $\boldsymbol{p}$ can be achieved by solving a Bilevel Inverse OCP of type (2.21) from section 2.4 based on given motion capture data. In chapter 5 and chapter 6 Bilevel Inverse OCPs for a basic human like motion and the gait of a CP patient are derived, respectively.

### Dynamics on Model Stages and their Transitions

For each model stage, $\mathrm{j} = 1, \ldots, \mathrm{n_S}$, on the time horizon $\mathcal{T}^{\mathrm{j}} = [t_{\mathrm{s}}^{\mathrm{j}}, t_{\mathrm{f}}^{\mathrm{j}}] = [0,1]$ with duration parameter $d_{\mathrm{j}}$ we can define equations of motion as in (3.5) depending on the actual constraint set $\boldsymbol{g}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}, \boldsymbol{p})$ which applies on the rigid multibody system during walking. With differential states and controls as defined in (3.9) for equation (3.8b) in multi-stage OCP (3.8) we have the following ODE system for model stage j

$$\begin{aligned}\dot{\boldsymbol{x}}_{\mathrm{j}}(t) &= d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), \boldsymbol{p}) \\ &= d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}(t), \dot{\boldsymbol{q}}_{\mathrm{j}}, \boldsymbol{\tau}_{\mathrm{j}}(t), \boldsymbol{p}) \\ &= d_{\mathrm{j}} \cdot \begin{pmatrix} \dot{\boldsymbol{q}}_{\mathrm{j}}(t) \\ \ddot{\boldsymbol{q}}_{\mathrm{j}}(t) \end{pmatrix}, \qquad\qquad t \in \mathcal{T}^{\mathrm{j}}, \end{aligned} \tag{3.13}$$

with initial conditions on $\boldsymbol{x}_{\mathrm{j}}$ at the beginning $t_{\mathrm{s}}^{\mathrm{j}} = 0$ of each stage incorporated in

$$\boldsymbol{g}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}(t_{\mathrm{s}}^{\mathrm{j}}), \boldsymbol{p}) = 0, \tag{3.14a}$$

$$\boldsymbol{G}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}(t_{\mathrm{s}}^{\mathrm{j}}), \boldsymbol{p})\dot{\boldsymbol{q}}_{\mathrm{j}}(t_{\mathrm{s}}^{\mathrm{j}}) = 0, \tag{3.14b}$$

and included in equation (3.8e) in the way as described in the following paragraph. The generalized acceleration $\ddot{\boldsymbol{q}}_{\mathrm{j}}(t)$ in (3.13) can be expressed as part of the solution of

$$\begin{pmatrix} \boldsymbol{H}(\boldsymbol{q}_{\mathrm{j}}, \boldsymbol{p}) & \boldsymbol{G}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}, \boldsymbol{p})^T \\ \boldsymbol{G}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}, \boldsymbol{p}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}}_{\mathrm{j}} \\ -\boldsymbol{\lambda}_{\mathrm{j}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau}_{\mathrm{j}} - \boldsymbol{C}(\boldsymbol{q}_{\mathrm{j}}, \dot{\boldsymbol{q}}_{\mathrm{j}}, \boldsymbol{p}) \\ \boldsymbol{\gamma}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}, \dot{\boldsymbol{q}}_{\mathrm{j}}, \boldsymbol{p}) \end{pmatrix}, \tag{3.15}$$

with contact force $\boldsymbol{\lambda}_{\mathrm{j}}$, contact Jacobian $\boldsymbol{G}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}, \boldsymbol{p})$ and contact Hessian $\boldsymbol{\gamma}_{\mathrm{j}}(\boldsymbol{q}_{\mathrm{j}}, \dot{\boldsymbol{q}}_{\mathrm{j}}, \boldsymbol{p})$.

Without loss of generality we assume that between two dynamic model stages a transition occurs, see (3.8c), which is expressed through

$$\boldsymbol{x}_{\mathrm{j}+1}(t_{\mathrm{s}}^{\mathrm{j}+1}) = \Delta_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{f}}^{\mathrm{j}}), \boldsymbol{p}), \quad \mathrm{j} = 1, \ldots, \mathrm{n_S} - 1,$$

where $\Delta_{\mathrm{j}} : \mathbb{R}^{\mathrm{n_{x_j}}} \times \mathbb{R}^{\mathrm{n_p}} \to \mathbb{R}^{\mathrm{n_{x_j}}}$ is a mapping between differential state $\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{f}}^{\mathrm{j}})$ at the end of model stage j and $\boldsymbol{x}_{\mathrm{j}+1}(t_{\mathrm{s}}^{\mathrm{j}+1})$ at the beginning of the subsequent model stage $\mathrm{j}+1$. In case where no collision appears the function $\Delta_{\mathrm{j}}(\cdot)$ is supposed to be identity mapping. Whereas a jump in the velocities occurs, if for instance one foot gets in touch with the ground and a perfectly inelastic collision takes place. The transition function is defined as identity mapping of the generalized coordinates, $\boldsymbol{q}_{\mathrm{j}}(t_{\mathrm{f}}^{\mathrm{j}}) = \boldsymbol{q}_{\mathrm{j}+1}(t_{\mathrm{s}}^{\mathrm{j}+1})$, and as part of the solution of the linear equation system similar to (3.7) at model stage with index j

$$\begin{pmatrix} \boldsymbol{H}(\boldsymbol{q}_{\mathrm{j}+1}, \boldsymbol{p}) & \boldsymbol{G}_{\mathrm{j}+1}(\boldsymbol{q}_{\mathrm{j}+1}, \boldsymbol{p})^T \\ \boldsymbol{G}_{\mathrm{j}+1}(\boldsymbol{q}_{\mathrm{j}+1}, \boldsymbol{p}) & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{q}}_{\mathrm{j}+1}^{+} \\ -\boldsymbol{\Lambda}_{\mathrm{j}+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H}(\boldsymbol{q}_{\mathrm{j}}, \boldsymbol{p})\dot{\boldsymbol{q}}_{\mathrm{j}}^{-} \\ \boldsymbol{0} \end{pmatrix}, \tag{3.16}$$

with *contact impulse* $\boldsymbol{\Lambda}_{\mathrm{j}+1}$ right after collision. The velocity before contact is denoted by $\dot{\boldsymbol{q}}_{\mathrm{j}}^{-} := \dot{\boldsymbol{q}}_{\mathrm{j}}(t_{\mathrm{f}}^{\mathrm{j}})$ and the velocity right after contact by $\dot{\boldsymbol{q}}_{\mathrm{j}+1}^{+} := \dot{\boldsymbol{q}}_{\mathrm{j}+1}(t_{\mathrm{s}}^{\mathrm{j}+1})$.

### Constraints in the Multi-Stage OCP

Because of transition condition (3.8c) it is sufficient to enforce initial conditions (3.14) to hold at the beginning of the first model stage. The equivalence of equation systems (3.15) and the corresponding equations of motion (3.3) extended by model index j is still ensured for all model stages. Hence, only initial conditions (3.14) for $\mathrm{j} = 1$ are included in equation (3.8e) together with other equality multi-point boundary conditions. To describe the human gait by an OCP as in (3.8) there are various ways to include the appearing constraints, such as collision avoidance with the surrounding, self-penetration avoidance or positive contact force at single support phase.

Additionally, periodicity constraints might be of interest. All these equality and inequality constraints enter the multi-stage OCP (3.8) in equations (3.8d), (3.8e) and (3.8f).

## 3.2 Cerebral Palsy

CP is a group of movement disorders that appear in early childhood and is defined as "[...] a group of permanent disorders of the development of movement and posture, causing activity limitation, that are attributed to non-progressive disturbances that occurred in the developing fetal or infant brain." [147]. In the first section of this chapter we will give a short introduction into CP, its causes, symptoms, and treatment. In the second section we will discuss the pathological gait associated with CP patients. This section is based on [18, 48, 75, 106, 137]

### 3.2.1 Introduction and Classification

CP is the most common childhood movement disorder with a prevalence of approximately 0.2% of live births. While the exact cause is often unknown, risk factors and potential mechanisms include but are not limited to genetic disposition, difficult delivery (hypoxia during birth), preterm birth, head trauma, and certain maternal infections (e.g. toxoplasmosis) during pregnancy. A detailed overview on CP including its causes, symptoms, and treatment is given in, e.g., [18, 47, 48, 65, 75, 106, 167, 137]. As a consequence of the brain lesion, CP is characterized by a variety of symptoms and impairments of motor function but can be also accompanied by problems with speaking, hearing, sensation, and vision. CP manifests in an abnormal muscle tone and sensory system. This especially can affect posture and result in deformed skeleton and joints. CP is primarily associated with abnormal reflexes, muscle tone, or coordination and motor development. This results in progressive orthopedic damage especially to bone and joint deformities.

#### Classification

There are a multitude of classifications used for CP in clinical practice that aim to guide treatment allocation by physicians. The most common systems, which can also be combined, include the classification according to the severity level (no CP, mild, moderate, severe), classification according to the topographical distribution of the impairment, classification based on the motor function, and the Gross Motor Function Classification System (GMFCS) [135].

The topographical classification describes the number of extremities or a defined side of the body that are affected. This includes Mono-, Di-, Tri-, Tetra- or Quadri-, and Pentaplegia for the number of extremities (head and neck as fifth included) and for the sides of the body Hemi- (one arm and one leg on one side of the body) and Paraplegia (upper or usually lower half of the body).

The classification of CP according to motor function gives indication about the region of the brain that is injured. It leads to two main classes, spastic and non-spastic. These have both multiple variations and it is possible that mixtures occur. Spastic CP accounts for the majority of CP cases with approximately 80% of cases being spastic. Spastic CP implies increased muscle tone that results in rigidity and stiffness due to continuously contracting muscles. Non-spastic CP mainly implies involuntary movements that can be also accompanied by tremors, or poor coordination and balance. Non-spastic CP can be further divided into two main subclasses, ataxic (affects coordinate movement, impaired balance and coordination) and dyskinetic (associated with involuntary movements).

CP can also be classified according to the GMFCS, which is a universal scheme that can be applied to all forms of CP. The GMFCS consists of a five level system according to the severity of activity limitation, which correspond to the extent of functional capacities of the patient [135].

### 3.2.2 Characteristics of Cerebral Palsy Gait

CP is a complex and heterogeneous continuum of motor disorders. This is further the case for the resulting orthopedic impairments. As a consequence of the joint and bone deformities and often occurring stiffness and rigidity, CP often leads to gait difficulties that increase over time and result in pathological gaits that span a broad continuum of deviations. The heterogeneous, complex, and wide range of clinical presentation of CP makes diagnosis and treatment allocation difficult.

The treatment of CP is limited to managing the orthopedic impairments and further symptoms. The main approaches are conservative therapy (e.g. physiotherapy), pharmacotherapy with antispasmodics to relax hypertonic muscles (e.g. botulinum toxin), and orthopedic interventions that include orthotic devices to stabilize joints and surgery, which includes loosening tight muscles or osteotomy to correct bone alignment. In Figure 3.1 the posture of a patient with CP before and after multiple orthopedic interventions is shown.

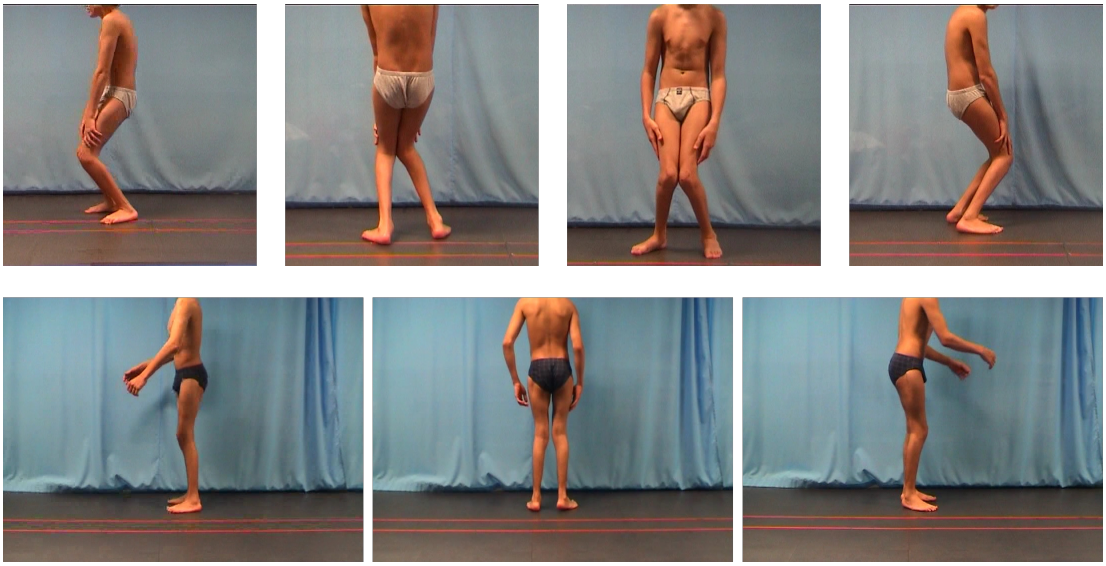

**Figure 3.1:** This figure shows the posture of a patient with CP bofore (upper row) and after (lower row) interventions. The picture is provided by the HEIDELBERG MOTIONLAB [173].

A focus of diagnosis and orthopedic intervention is set on the pathological gait of CP patients. The typical pathological gait patterns of spastic CP patients can be classified into four types for spastic hemiplegia (drop foot, equinus with various knee positions; type 1-4 [172]) and four types for spastic diplegia (true equinus, jump gait, apparent equinus and crouch gait [146]). Nevertheless, gait deviations in CP patients are highly variable and a continuum of deviations rather than well delineated.

A cornerstone of CP diagnosis and treatment planning is gait analysis. Gait analysis collects quantitative, spatiotemporal data on kinematics via motion capture and visual assessment of the gait from different angles, as well as measurement of kinetic data like ground reaction and joint forces, as well as electromyography for measurement of muscle contraction. Combined with further data on muscle strength, passive range of motion, body measurements like bone length, height and weight, gait analysis can generate important characteristics of the pathological gait of CP patients and support intervention evaluation and planning and may distinguish gait deviations from compensatory strategies. Gait analysis aims at identifying the cause of the way that CP patients walk.

Such gait analysis in the context of CP patient treatment are routinely performed in clinical practice. In Heidelberg this is done by the HEIDELBERG MOTIONLAB [173] that is embedded in the Department of Orthopedic Surgery of the Heidelberg University Hospital and supports the project of this thesis by providing gait analysis data of CP patients. In Figure 3.2 it is illustrated how Vicon motion capture marker [2] are attached to the skin of a patient with CP. To further improve the existing protocols and methods this thesis aims at establish-

ing a diagnosis methodology based on Bilevel Inverse Optimal Control with underlying rigid multibody system dynamics to calibrate a patient-specific CP gait model under consideration of given motion capture data, see chapter 6 and chapter 10.

**Figure 3.2:** This figure shows the posture of a patient with CP with attached Vicon motion capture marker. The picture is provided by the HEIDELBERG MOTIONLAB [173].

# Part II

# Contributions

# Chapter 4

# An Efficient Direct Approach for Bilevel Inverse OCPs with Fixed Active Set

In this chapter we present our new mathematical method, the *Direct Simultaneous Approach with Fixed Active Set* - or DISIMFAS for short - for its efficient and reliable use in the identification of unknowns in a CP patient's gait model by solving a Bilevel Inverse OCP. General feasibility is shown by its successful application in two challenging numerical examples apart from human locomotion, the rocket car example and the polar robot example in chapter 8. Furthermore, in chapter 9 its performance is investigated for a basic gait model - the basic walker model from chapter 5.

This chapter is organized as follows. It starts with an introduction, where we give the general problem formulation and the motivation behind the developed numerical method. In section 4.2 the solution approach is derived in detail and in section 4.4 an outlook is given how the identification of the active set can be integrated in a sequential algorithm.

## 4.1 Introduction and Motivation

Our ultimate goal is the identification of optimization criteria and model parameters in an OCP describing the gait of a CP patient by solving a Bilevel Inverse OCP of the form (2.21) as introduced in section 2.4. As already discussed in detail in section 2.4, the numerical solution of these kinds of bilevel problems is challenging. Hence, already available solution approaches mostly cannot easily be applied in the context of human locomotion, where a large-scale Bilevel Inverse OCP has to be solved with a highly non-smooth multi-level OCP on the lower level with mixed control-state constraints. In our investigations, the Direct All-at-Once Approach (see subsection 2.4.3) in the implementation PARAOCP [80] with selected approaches for the treatment of the arising complementarity constraints, has shown to be not sufficient for the basic walker example of chapter 5 using simulated measurements - although it is a basic model for human locomotion with only few DOFs compared to the CP gait model from chapter 6. Even our extension of the proposed method by solving the resulting one-level NLP (2.27) with an efficient interior-point method IPOPT [169] together with the regularization scheme by Scholtes [153] for the complementarity constraints failed for the basic walker example of chapter 5. In all calculations, infeasible quadratic programs occured that arise from the resulting MPCCs and the lack of standard CQs. These could not be tackled within the solution procedure by selected regularization and penalization approaches in PARAOCP. Nevertheless, in a first analysis of a CP patient's gait in [80, Chapter 15] the exact penalty approach as described in Benson et al. [22] lead to a solution. However, in this analysis the complementarity constraints, which are expected to vanish for a sufficiently large penalization parameter, could not be driven to zero in the solution.

Hence, the development of a different solution strategy was necessary to meet our ultimate goal for an efficient and reliable numerical method to solve Bilevel Inverse OCPs for CP gaits. A desired feature is its routinely applicability by physicians. Hence, on the one hand, information on the given dynamic process should be incorporated in such a way that expert knowledge of the physicians is not lost but can be an integral part of the whole solution approach. On the other hand, structure exploitation should facilitate the solution method for easy application.

The Bilevel Inverse OCP (2.21) we consider is of a special class: a PE problem constrained by a multi-stage OCP with discontinuities, where the unknowns in the optimal control model are determined by fitting the

model to given measurement data. This condition enables us to incorporate some given structural information of the dynamic process we are looking at, and in particular of the gait of a CP patient. In general, because in our setting the deviation of the model to measurement data is minimized, bounds on decision variables can often be chosen in such a way that the variables usually stay in the interior of their natural bounds or at least the cases where these bounds might get active are rare. Hence, the considered index set of possible active bounds can be reduced.

In the special case, where the lower level OCP describes the gait of a CP patient the inequality mixed control-state constraints as well as inequality multi-point boundary constraints are usually already met by given measurement data. One prominent example is the constraint which ensures that one foot remains above ground during a single support phase. Another example are self-penetration constraints. Despite their essential role when the OCP is solved alone, in the bilevel framework these constraints can be formulated very loose or might be even neglected completely. Furthermore, we are interested in rigid multibody system models for the human gait, which incorporate potentially appearing limited range of motion already in the equations of motion themselves, rather than using simple bounds on the variables. Especially in modeling the so-called *crouched gait* of CP patients, a parametrized dynamic model should be considered, because essential information regarding possible clinical treatments can be extracted from the individually determined parameters. In chapter 6 such a patient-specific CP gait model is established.

All these circumstances motivated the development of a numerical method for PE problems constrained by a multi-stage OCP with discontinuities under consideration of structural information gained from measurement data. The essence of our method is to make a suitable guess of the optimal active set of the lower level OCP. In the following section the new solution approach for Bilevel Inverse OCPs is described in detail.

## 4.2 Numerical Solution Approach

The main steps of the DISIMFAS are based on the Direct All-at-Once Approach by Hatz et al. [81, 80]. In the first step, the lower level multi-stage OCP is treated with a direct method according to the Direct Multiple Shooting Method described earlier in subsection 2.2.1. The resulting finite-dimensional nonlinear bilevel program comprises a structured NLP on the lower level. In contrast to the Direct All-at-Once Approach, in the newly proposed method this lower level NLP is first reformulated by considering a guess of the optimal active set before replacing it by its KKT conditions. This transforms the bilevel program to a one-level NLP which is no MPCC as in the Direct All-at-Once Approach [80], see subsection 2.4.3, but of general form (1.1) as defined in chapter 1. Hence, numerical methods from section 1.2 can be used.

For an efficient implementation, we exploit the special structure of the NLP and give needed quantities for a tailored General Gauß Newton method in a SQP framework as well as for general SQP and interior-point methods. The identification of the optimal active set in a sequential approach is the topic of section 4.4.

In the following we describe the steps summarized above in more detail under consideration of the Bilevel Inverse OCP (2.22) with a single-stage OCP on the lower level from subsection 2.4.3. The Bilevel Inverse OCP (2.22) is defined on a fixed normalized time horizon $t \in [0,1]$ with free duration parameter $d \in \mathbb{R}$. All arising variables and functions are introduced in subsection 2.4.1. Without loss of generality only Mayer type objectives are considered. As outlined in the introduction of the Direct All-at-Once Approach, at this point, the single-stage formulation facilitates the notation and concurrently captures the main characteristics of the mathematical method as well. Situations where the multi-stage formulation may affect the proposed method are rare and mostly covered by incorporating the transition condition (2.21d).

### 4.2.1 Step 1: Application of the Direct Multiple Shooting Method on the Lower Level

This step is the same as in the Direct All-at-Once Approach [80] and already described in subsection 2.4.3. In the following the application of the Direct Multiple Shooting Method on the lower level OCP is explained briefly with focus on the quantities that are required later in section 4.3. The resulting bilevel NLP (4.5) that is used in the next steps of our DISIMFAS differs in the formulation of the bounds on the variables, but is equivalent to (2.24).

The lower level single-stage OCP is treated with the Direct Multiple Shooting Method according to subsection 2.2.1, where the differential states are parametrized and the control functions, constraints, and lower level objective are discretized on the multiple shooting grid

$$0 = t_0^{\mathrm{ms}} \leq t_1^{\mathrm{ms}} \leq \ldots \leq t_{\mathrm{N}}^{\mathrm{ms}} = 1.$$

This leads to NLP (2.15). For a more compact representation as stated in NLP (2.17), the variables are combined on each shooting node $\mathrm{i} = 0, \ldots, \mathrm{N}$ in the vectors $\boldsymbol{v}_{\mathrm{i}} := \begin{pmatrix} \boldsymbol{s}_{\mathrm{i}}^T & \boldsymbol{q}_{\mathrm{i}}^T \end{pmatrix}^T$ and collected in $\boldsymbol{v} := \begin{pmatrix} \boldsymbol{v}_0^T & \ldots & \boldsymbol{v}_{\mathrm{N}}^T \end{pmatrix}^T$ according to subsection 2.2.1. Furthermore, slack variables $\boldsymbol{w} \in \mathbb{R}^{\mathrm{n_{C_{ic}}}}$ are introduced, such that all constraints can be combined in one constraint function

$$\boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}) := \begin{pmatrix} \boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v}, d, \boldsymbol{p}) \\ \boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v}, \boldsymbol{p}) - \boldsymbol{w} \end{pmatrix} = 0 \quad \text{with} \quad \boldsymbol{w} \geq 0, \tag{4.1}$$

where $\boldsymbol{z} := \begin{pmatrix} \boldsymbol{v}^T & d & \boldsymbol{w}^T \end{pmatrix}^T, \boldsymbol{z} \in \mathbb{R}^{\mathrm{n_z}}$. Simple bounds on the variables of the lower level NLP are defined as $\overline{\boldsymbol{z}} := \begin{pmatrix} \overline{\boldsymbol{v}}^T & \overline{d} & \infty \end{pmatrix}^T$ and $\underline{\boldsymbol{z}} := \begin{pmatrix} \underline{\boldsymbol{v}}^T & \underline{d} & 0 \end{pmatrix}^T$. Without loss of generality, we assume that

$$\overline{z_{\mathrm{r}}} \neq \underline{z_{\mathrm{r}}}, \quad \text{for all} \quad \mathrm{r} = 0, \ldots, \mathrm{n_z} - 1. \tag{4.2}$$

Variables that are fixed to a specific value can be included in the equality constraints $\boldsymbol{C}_{\mathrm{ec}}(\cdot)$. In the compact notation of (4.1) all matching conditions which arise due to the application of the Direct Multiple Shooting Method and remaining boundary equality constraints are combined in the vector

$$\boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v}, d, \boldsymbol{p}) := \begin{pmatrix} \boldsymbol{x}(t_1^{\mathrm{ms}}; \boldsymbol{s}_0, \boldsymbol{q}_0, d, \boldsymbol{p}) - \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{x}(t_{\mathrm{N}}^{\mathrm{ms}}; \boldsymbol{s}_{\mathrm{N}-1}, \boldsymbol{q}_{\mathrm{N}-1}, d, \boldsymbol{p}) - \boldsymbol{s}_{\mathrm{N}} \\ \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}) \end{pmatrix}, \tag{4.3}$$

and all mixed control-state constraints together with further boundary inequality constraints enter

$$\boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v}, \boldsymbol{p}) := \begin{pmatrix} \tilde{\boldsymbol{c}}(\boldsymbol{s}_0, \hat{\boldsymbol{u}}_0(t_0^{\mathrm{ms}}, \boldsymbol{q}_0), \boldsymbol{p}) \\ \vdots \\ \tilde{\boldsymbol{c}}(\boldsymbol{s}_{\mathrm{N}}, \hat{\boldsymbol{u}}_{\mathrm{N}}(t_{\mathrm{N}}^{\mathrm{ms}}, \boldsymbol{q}_{\mathrm{N}}), \boldsymbol{p}) \\ \tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p}) \end{pmatrix}, \tag{4.4}$$

where all simple bounds on the variables are excluded from the original inequality constraints and, hence, the tilde notation is used. Furthermore, please note that in a multi-stage environment in addition to matching conditions also transition conditions between the model stages would enter the equality constraint vector (4.3) as described in subsection 2.2.1.

Altogether, the lower level OCP is now replaced by this structured NLP in compact form as stated in (2.17) with the assumption in (4.2). Finally, this leads to the finite-dimensional bilevel nonlinear problem of the

following form

$$\min_{\boldsymbol{\alpha},\boldsymbol{p},\boldsymbol{z}} \quad \frac{1}{2}\sum_{\mathrm{n}=0}^{\mathrm{n_m}-1}\sum_{\mathrm{k}=0}^{\mathrm{n_h}-1}\frac{\left(h_{\mathrm{k}}(\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{nk}}^2} \tag{4.5a}$$

s.t.

$$\min_{\boldsymbol{z}} \quad \boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{\alpha},\boldsymbol{p}) \tag{4.5b}$$

$$\text{s.t.} \quad 0=\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p}), \tag{4.5c}$$

$$\underline{\boldsymbol{z}}\le\boldsymbol{z}\le\overline{\boldsymbol{z}}, \tag{4.5d}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n_M}}\alpha_{\mathrm{k}}=1,\boldsymbol{\alpha}\ge 0, \tag{4.5e}$$

$$\underline{\boldsymbol{p}}\le\boldsymbol{p}\le\overline{\boldsymbol{p}}, \tag{4.5f}$$

with simple bounds on the decision variables $\boldsymbol{z}=\begin{pmatrix}\boldsymbol{v}^T & d & \boldsymbol{w}^T\end{pmatrix}^T$ as the only inequality constraints of the lower level NLP in (4.5d). The least-squares objective (4.5a) depends on the term $\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}})$ and is treated according to step 1 of subsection 2.4.3.

### 4.2.2 Step 2: Reformulation of the Lower Level NLP on a Fixed Active Set

The finite-dimensional nonlinear bilevel program (4.5) comprises a structured NLP on the lower level for which we assume that the LICQ holds at the solution $\boldsymbol{z}^*$. Furthermore, we assume that $\boldsymbol{z}^*$ is unique in some neighborhood. To provide an adequate reformulation of the lower level NLP we consider general active-set methods for inequality-constrained problems. The essence of active-set methods is to start by making a guess of the optimal active set $\mathcal{A}^*$. The optimal active set is defined as the index set of all constraints on the lower level, which are satisfied as equalities at the solution. Hence, the idea is now to consider a suitable guess of the optimal active set, before the lower level NLP is replaced by its KKT conditions.

We first define an ordered index set for the inequality constraints (4.5d) related to the upper and lower bounds on the lower level optimization variables $\boldsymbol{z}\in\mathbb{R}^{\mathrm{n_z}}$ by

$$\mathcal{I}:=\left\{\mathrm{r}|\underline{z_{\mathrm{r}}}\le z_{\mathrm{r}}\le\overline{z_{\mathrm{r}}}\right\}=\{0,\ldots,\mathrm{n_z}-1\},$$

starting with 0 to simplify the notation in later sections. Furthermore, let $\boldsymbol{z}\in\Omega$ be feasible, $\mathcal{E}$ the index set that corresponds to the equality constraints on the lower level, and order the indices in $\mathcal{E}=\{\mathrm{n_z},\ldots,\mathrm{n_z}+\mathrm{n_{C_{ec}}}+\mathrm{n_{C_{ic}}}-1\}$ such that $\mathcal{E}\cap\mathcal{I}=\emptyset$, then the active set for the lower level NLP in (4.5) can be written as

$$\mathcal{A}(\boldsymbol{z}):=\mathcal{E}\cup\left\{\mathrm{r}\in\mathcal{I}|0=\overline{z_{\mathrm{r}}}-z_{\mathrm{r}}\right\}\cup\left\{\mathrm{r}\in\mathcal{I}|0=z_{\mathrm{r}}-\underline{z_{\mathrm{r}}}\right\},$$

and the active sets that belong to the active upper bounds and active lower bounds as

$$\overline{\mathcal{A}}(\boldsymbol{z}):=\left\{\mathrm{r}\in\mathcal{I}|0=\overline{z_{\mathrm{r}}}-z_{\mathrm{r}}\right\}\qquad\text{and}\qquad\underline{\mathcal{A}}(\boldsymbol{z}):=\left\{\mathrm{r}\in\mathcal{I}|0=z_{\mathrm{r}}-\underline{z_{\mathrm{r}}}\right\},$$

respectively. With the assumption that $\overline{z_{\mathrm{r}}}\neq\underline{z_{\mathrm{r}}}$ for all variables $\mathrm{r}=0,\ldots,\mathrm{n_z}-1$, as stated in subsection 4.2.1, the active sets related to the bounds are disjoint and we have

$$\overline{\mathcal{A}}(\boldsymbol{z})\cap\underline{\mathcal{A}}(\boldsymbol{z})=\emptyset.$$

The optimal active set $\mathcal{A}^*:=\mathcal{A}(\boldsymbol{z}^*)$ and the active sets related to the inequality constraints $\overline{\mathcal{A}^{\mathrm{ic}}}^*:=\overline{\mathcal{A}}(\boldsymbol{z}^*)$ and $\underline{\mathcal{A}^{\mathrm{ic}}}^*:=\underline{\mathcal{A}}(\boldsymbol{z}^*)$ are defined as the active sets at the solution $\boldsymbol{z}^*$. We assume above that the LICQ is satisfied and, hence, the active constraints are linearly independent at the solution. On the lower level we can now formulate

an NLP of the form

$$\min_{\boldsymbol{z}} \quad \boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{\alpha}, \boldsymbol{p}) \tag{4.6a}$$

$$\text{s.t.} \quad 0 = \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}), \tag{4.6b}$$

$$0 = \overline{z}_{\mathrm{r}} - z_{\mathrm{r}}, \qquad \forall \mathrm{r} \in \overline{\mathcal{A}^{\mathrm{ic}}}^{*}, \tag{4.6c}$$

$$0 = z_{\mathrm{r}} - \underline{z}_{\mathrm{r}}, \qquad \forall \mathrm{r} \in \underline{\mathcal{A}^{\mathrm{ic}}}^{*}, \tag{4.6d}$$

which only comprises equality constraints on the optimal active set $\mathcal{A}^*$. NLP (4.6) is equivalent to the lower level NLP in (4.5), if all $z_{\mathrm{r}}$ with $\mathrm{r} \notin \overline{\mathcal{A}^{\mathrm{ic}}}^{*}$ and $\mathrm{r} \notin \underline{\mathcal{A}^{\mathrm{ic}}}^{*}$ satisfy $\underline{z}_{\mathrm{r}} \leq z_{\mathrm{r}} \leq \overline{z}_{\mathrm{r}}$.

Of course, in general the optimal active set is not known in advance. We introduce the *working set* $\mathcal{W}$ as a suitable guess of the optimal active set $\mathcal{A}^*$ with linearly independent constraint gradients. If we replace the optimal active sets in (4.6c) and (4.6d) by the corresponding working sets $\overline{\mathcal{W}}$ and $\underline{\mathcal{W}}$, respectively, we can formulate a bilevel problem as follows

$$\min_{\boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{z}} \quad \frac{1}{2} \sum_{\mathrm{n}=0}^{\mathrm{n_m}-1} \sum_{\mathrm{k}=0}^{\mathrm{n_h}-1} \frac{\left(h_{\mathrm{k}}(\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}}), \boldsymbol{p}) - \eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{nk}}^2} \tag{4.7a}$$

$$\text{s.t.}$$

$$\min_{\boldsymbol{z}} \quad \boldsymbol{\Phi}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{\alpha}, \boldsymbol{p}) \tag{4.7b}$$

$$\text{s.t.} \quad 0 = \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}), \tag{4.7c}$$

$$0 = \overline{z}_{\mathrm{r}} - z_{\mathrm{r}}, \qquad \forall \mathrm{r} \in \overline{\mathcal{W}}, \tag{4.7d}$$

$$0 = z_{\mathrm{r}} - \underline{z}_{\mathrm{r}}, \qquad \forall \mathrm{r} \in \underline{\mathcal{W}}. \tag{4.7e}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} = 1, \boldsymbol{\alpha} \geq 0, \tag{4.7f}$$

$$\underline{\boldsymbol{p}} \leq \boldsymbol{p} \leq \overline{\boldsymbol{p}}, \tag{4.7g}$$

where the working sets related to the bounds are disjoint because of the assumption that $\overline{z_{\mathrm{r}}} \neq \underline{z_{\mathrm{r}}}$ for all variables $\mathrm{r} = 0, \ldots, \mathrm{n_z} - 1$ and we have

$$\overline{\mathcal{W}}(\boldsymbol{z}) \cap \underline{\mathcal{W}}(\boldsymbol{z}) = \emptyset. \tag{4.8}$$

In the case, where the guess of the working set $\mathcal{W}$ is set to the optimal active set $\mathcal{A}^*$, bilevel NLPs (4.5) and (4.7) are equivalent. With a suitable first guess of the working set $\mathcal{W}^0$ the optimal active set $\mathcal{A}^*$ can be identified by a sequential algorithm as to be discussed in section 4.4. In each outer iteration $k$ of the sequential algorithm an NLP of the form (4.7) on a fixed working set $\mathcal{W}^k := \mathcal{W}(\boldsymbol{z}^k)$ is treated according to the following subsection 4.2.3 and subsection 4.2.4.

### 4.2.3 Step 3: Replacement of the Lower Level NLP by its KKT Conditions

In this step, the lower level NLP of the bilevel problem (4.7) is replaced by its KKT conditions. This transforms the bilevel program into a one-level NLP. Because of the reformulation in step 2 according to subsection 4.2.2 the resulting NLP is no MPCC as in the approach of Hatz [80], see also subsection 2.4.3, but of general form (1.1) as defined in chapter 1 and, hence, standard NLP solvers can be applied.

We introduce Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_{c_{ec}}+n_{c_{ic}}}$ related to equality constraints (4.7c) and $\boldsymbol{\mu} \in \mathbb{R}^{n_z}$ related to simple bounds. The entries of $\boldsymbol{\mu}$ are defined by

$$\mu_r := \begin{cases} \mu_{\overline{z}_r}, & \text{if } r \in \overline{\mathcal{W}}, \\ \mu_{\underline{z}_r}, & \text{if } r \in \underline{\mathcal{W}}, \\ 0, & \text{otherwise,} \end{cases}$$

where the value of the r-th component corresponding to active upper bound $\overline{z}_r$ is denoted by $\mu_{\overline{z}_r}$ and the one for the corresponding active lower bound $\underline{z}_r$ by $\mu_{\underline{z}_r}$. Furthermore, we introduce matrices denoted by $\boldsymbol{I}_{\overline{\mathcal{W}}} \in \mathbb{R}^{|\overline{\mathcal{W}}|\times n_z}$ and $\boldsymbol{I}_{\underline{\mathcal{W}}} \in \mathbb{R}^{|\underline{\mathcal{W}}|\times n_z}$, which are composed of standard unit vectors $\boldsymbol{e}_r \in \mathbb{R}^{n_z}$ corresponding to indices $r \in \overline{\mathcal{W}}$ and $r \in \underline{\mathcal{W}}$, respectively. Then multiplier vectors $\boldsymbol{\mu}_{\overline{z}} \in \mathbb{R}^{|\overline{\mathcal{W}}|}$ and $\boldsymbol{\mu}_{\underline{z}} \in \mathbb{R}^{|\underline{\mathcal{W}}|}$ related to upper and lower bounds, respectively, can be defined by

$$\boldsymbol{\mu}_{\overline{z}} := \boldsymbol{I}_{\overline{\mathcal{W}}}^T \cdot \boldsymbol{\mu} \quad \text{and} \quad \boldsymbol{\mu}_{\underline{z}} := \boldsymbol{I}_{\underline{\mathcal{W}}}^T \cdot \boldsymbol{\mu}. \tag{4.9}$$

The newly introduced Lagrange multipliers in (4.9) together with the Lagrange multipliers $\boldsymbol{\lambda}$ related to equality constraints and the decision variables $\boldsymbol{z}$, $\boldsymbol{\alpha}$, and $\boldsymbol{p}$ of the bilevel NLP (4.7) can be combined in the variable vector $\boldsymbol{Z} := \begin{pmatrix} \boldsymbol{z}^T & \boldsymbol{\lambda}^T & \boldsymbol{\mu}_{\overline{z}}^T & \boldsymbol{\mu}_{\underline{z}}^T & \boldsymbol{\alpha}^T & \boldsymbol{p}^T \end{pmatrix}^T$.

Now the Lagrange function of the lower level NLP of (4.7) on a fixed guess $\mathcal{W}$ of the optimal active set $\mathcal{A}^*$ is defined by

$$\mathcal{L}(\boldsymbol{Z}) := \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \boldsymbol{\lambda}^T \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}) - \sum_{r\in\overline{\mathcal{W}}} \mu_r \left(\overline{z}_r - z_r\right) - \sum_{r\in\underline{\mathcal{W}}} \mu_r \left(z_r - \underline{z}_r\right).$$

Furthermore, with the introduction of vectors $\hat{\boldsymbol{\mu}}_{\overline{z}} \in \mathbb{R}^{n_z}$ and $\hat{\boldsymbol{\mu}}_{\underline{z}} \in \mathbb{R}^{n_z}$ related to upper and lower bounds, respectively, defined by

$$(\hat{\mu}_{\overline{z}})_r := \begin{cases} \mu_r, & \text{if } r \in \overline{\mathcal{W}}, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad (\hat{\mu}_{\underline{z}})_r := \begin{cases} \mu_r, & \text{if } r \in \underline{\mathcal{W}}, \\ 0, & \text{otherwise,} \end{cases} \tag{4.10}$$

the gradient of the Lagrangian with respect to the lower level variables $\boldsymbol{z}$ can be written as

$$\begin{aligned} \nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{Z}) &:= \nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p})^T \boldsymbol{\lambda} - \sum_{r\in\overline{\mathcal{W}}} \mu_r \nabla_{\boldsymbol{z}} \left(\overline{z}_r - z_r\right) - \sum_{r\in\underline{\mathcal{W}}} \mu_r \nabla_{\boldsymbol{z}} \left(z_r - \underline{z}_r\right) \\ &= \nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p})^T \boldsymbol{\lambda} + \hat{\boldsymbol{\mu}}_{\overline{z}} - \hat{\boldsymbol{\mu}}_{\underline{z}}. \end{aligned}$$

Note that the non-zero entries of the vectors $\hat{\boldsymbol{\mu}}_{\overline{z}}$ and $\hat{\boldsymbol{\mu}}_{\underline{z}}$ are the corresponding entries of the Lagrange multipliers $\boldsymbol{\mu}_{\overline{z}}$ and $\boldsymbol{\mu}_{\underline{z}}$, respectively.

With the assumption that LICQ holds at the solution of the lower level problem the KKT conditions can now be formulated as

Stationarity:

$$0 = \nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{Z}), \tag{4.12a}$$

Primal Feasibility:

$$0 = \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p}), \tag{4.12b}$$

$$0 = \overline{z}_r - z_r, \qquad \forall r \in \overline{\mathcal{W}}, \tag{4.12c}$$

$$0 = z_r - \underline{z}_r, \qquad \forall r \in \underline{\mathcal{W}}, \tag{4.12d}$$

according to theorem 1.8 in subsection 1.1.1. Replacing the discretized and parametrized lower level problem by conditions (4.12) leads to the following structured one-level NLP

$$\min_{\boldsymbol{Z}} \quad \frac{1}{2}\sum_{\mathrm{n}=0}^{\mathrm{n_m}-1}\sum_{\mathrm{k}=0}^{\mathrm{n_h}-1}\frac{\left(h_{\mathrm{k}}(\boldsymbol{x}(t_{\mathrm{n}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{nk}}^2} \tag{4.13a}$$

$$\text{s.t.} \quad 0 = \nabla_{\boldsymbol{z}}\mathcal{L}(\boldsymbol{Z}), \tag{4.13b}$$

$$0 = \boldsymbol{C}(\boldsymbol{z},\boldsymbol{p}), \tag{4.13c}$$

$$0 = \overline{z}_{\mathrm{r}} - z_{\mathrm{r}}, \qquad \forall \mathrm{r} \in \overline{\mathcal{W}}, \tag{4.13d}$$

$$0 = z_{\mathrm{r}} - \underline{z}_{\mathrm{r}}, \qquad \forall \mathrm{r} \in \underline{\mathcal{W}}, \tag{4.13e}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} = 1, \boldsymbol{\alpha} \geq 0, \tag{4.13f}$$

$$\underline{\boldsymbol{p}} \leq \boldsymbol{p} \leq \overline{\boldsymbol{p}}. \tag{4.13g}$$

If the estimated working set is already the optimal active set $\mathcal{A}^*$ the resulting one-level NLP solves the original Bilevel Inverse OCP (2.22). It should be noted, that as stated in [80] for the Direct All-at-Once Approach "[...] for a general OCP on the lower level in (2.22), the bilevel problem and the resulting one-level problem are not necessarily mathematically equivalent". In this thesis, we do not cover investigations in this direction, and refer the interested reader to, e.g., [44, 45] for a detailed discussion on this topic. In the next subsection we describe how NLP (4.13) can be solved efficiently with standard solvers for constrained NLPs.

### 4.2.4 Step 4: Solution of the One-Level NLP with Tailored Numerical Methods

In the last step, the one-level NLP (4.13) is solved on a given fixed working set $\mathcal{W}$ as a guess for the optimal active set $\mathcal{A}^*$. The identification of the optimal active set with a sequential algorithm is discussed in section 4.4, where step 4 is performed in each outer iteration.

In the following section 4.3, we focus on the exploitation of given structures in NLP (4.13). This investigation is then used for efficient computation and derivative generation of quantities needed by most state-of-the-art solvers, which implement variants of the numerical methods for NLPs described in section 1.2.

In the software package ParDynOpt of chapter 7 the DiSimFAS is realized including the structure exploitation described in the following. For integration of the arising IVPs and the corresponding sensitivity generation, as well as the computation of derivatives of other model functions up to second order are realized within an interface to SolvIND [5]. This software package was developed by Albersmeyer in the Simulation and Optimization group of Bock at Interdisciplinary Center for Scientific Computing at Heidelberg University, and provides ODE and DAE solvers in an IND framework for exact derivative generation using AD with ADOL-C [170]. Furthermore, ParDynOpt provides interfaces to Ipopt [169] for the solution of the resulting structured one-level NLP (4.13) with an efficient interior-point method and a Broyden–Fletcher–Goldfarb–Shanno (BFGS) Hessian approximation. Interfaces for SNOPT [71] and filterSQP [59] are also prepared for implementations of SQP methods with appropriate hessian approximations for a Generalized Gauß-Newton framework as described at the end of the following section 4.3.

## 4.3 Structure Exploitation and Hessian Approximation

This section is structured as follows. We first provide the gradients of the upper level objective, and the constraints with a focus on the gradient of the Lagrangian of the lower level NLP $\nabla_{\boldsymbol{z}}\mathcal{L}(\cdot)$ and exploit their structures. Furthermore, for an efficient implementation the sparsity of the Jacobian of the constraints is investigated. The section 4.3 is completed with a discussion on adequate Hessian approximations of the Lagrangian of the structured one-level NLP (4.13) for tailored interior-point methods or SQP methods, as well as for a Generalized Gauß-Newton framework.

### 4.3.1 Efficient Computation of Objective Gradients

We start with the objective of the one-level NLP (4.13). To simplify the notation we assume that the measurement grid is the same as the multiple shooting grid

$$0 = t_0^{\mathrm{ms}} \leq t_1^{\mathrm{ms}} \leq \ldots \leq t_{\mathrm{N}}^{\mathrm{ms}} = 1,$$

and introduce a weighted residual vector $\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p}) := \boldsymbol{\Sigma}^{-1}(\boldsymbol{h}(\boldsymbol{s},\boldsymbol{p}) - \boldsymbol{\eta})$, such that the objective can be rewritten as

$$\frac{1}{2}\left\|\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p})\right\|_2^2 = \frac{1}{2}\left\|\boldsymbol{\Sigma}^{-1}(\boldsymbol{h}(\boldsymbol{s},\boldsymbol{p}) - \boldsymbol{\eta})\right\|_2^2, \tag{4.14}$$

where $\boldsymbol{\eta} := \begin{pmatrix}\boldsymbol{\eta}_0^T & \ldots & \boldsymbol{\eta}_{\mathrm{N}}^T\end{pmatrix}^T$ combines all measurements into one measurement vector. Furthermore, a vector with corresponding model answers is defined by $\boldsymbol{h}(\boldsymbol{s},\boldsymbol{p}) := \begin{pmatrix}\boldsymbol{h}_0(\boldsymbol{s}_0,\boldsymbol{p})^T & \ldots & \boldsymbol{h}_{\mathrm{N}}(\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p})^T\end{pmatrix}^T$, and a diagonal matrix with standard deviations by $\boldsymbol{\Sigma} := \mathrm{diag}(\boldsymbol{\sigma}_0,\ldots,\boldsymbol{\sigma}_{\mathrm{N}})$. In this notation each vector, $\boldsymbol{\eta}_{\mathrm{i}}$ and $\boldsymbol{h}_{\mathrm{i}}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})$, respectively, includes $\mathrm{n_h}$ entries and the matrices $\boldsymbol{\sigma}_{\mathrm{i}}$ themselves are also diagonal matrices with $\mathrm{n_h}$ non-zero entries related to each shooting node $\mathrm{i} = 0,\ldots,\mathrm{N}$. They read as follows

$$\boldsymbol{\eta}_{\mathrm{i}} = \begin{pmatrix}\eta_{\mathrm{i}0}\\ \vdots \\ \eta_{\mathrm{i}\mathrm{n}_{\mathrm{h}-1}}\end{pmatrix}, \quad \boldsymbol{h}_{\mathrm{i}}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p}) = \begin{pmatrix}h_0(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})\\ \vdots \\ h_{\mathrm{n}_{\mathrm{h}-1}}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})\end{pmatrix}, \quad \text{and} \quad \boldsymbol{\sigma}_{\mathrm{i}} = \begin{pmatrix}\sigma_{\mathrm{i}0} & & \\ & \ddots & \\ & & \sigma_{\mathrm{i}\mathrm{n}_{\mathrm{h}-1}}\end{pmatrix}.$$

Now the gradient of (4.14) with respect to the decision variables $\boldsymbol{Z}$ can be written as

$$\nabla_{\boldsymbol{Z}}\left(\frac{1}{2}\left\|\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p})\right\|_2^2\right) = \nabla_{\boldsymbol{Z}}\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p})^T \cdot \boldsymbol{R}(\boldsymbol{s},\boldsymbol{p}),$$

with $\nabla_{\boldsymbol{Z}}\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p})$ denoted as the Jacobian of the residual vector, where only those entries are non-zero that correspond to the parametrized multiple shooting state variables $\boldsymbol{s}$ and the model parameters $\boldsymbol{p}$. At each multiple shooting node $\mathrm{i} = 0,\ldots,\mathrm{N}$ we define submatrices by

$$\nabla_{\boldsymbol{v}_{\mathrm{i}}}\boldsymbol{R}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p}) = \boldsymbol{\sigma}_{\mathrm{i}}^{-1}\begin{pmatrix}\frac{\partial h_0(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})}{\partial s_{\mathrm{i},0}} & \ldots & \frac{\partial h_0(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})}{\partial s_{\mathrm{i},\mathrm{n_x}-1}} & 0 & \ldots & 0\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_{\mathrm{n}_{\mathrm{h}-1}}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})}{\partial s_{\mathrm{i},0}} & \ldots & \frac{\partial h_{\mathrm{n}_{\mathrm{h}}-1}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p})}{\partial s_{\mathrm{i},\mathrm{n_x}-1}} & 0 & \ldots & 0\end{pmatrix} = \begin{pmatrix}* & 0\end{pmatrix} \in \mathbb{R}^{\mathrm{n_h}\times(\mathrm{n_x}+\mathrm{n_q})},$$

as Jacobians related to node i with respect to variables $\boldsymbol{v}_{\mathrm{i}} := \begin{pmatrix}\boldsymbol{s}_{\mathrm{i}}^T & \boldsymbol{q}_{\mathrm{i}}^T\end{pmatrix}^T$. If we combine them into

$$\nabla_{\boldsymbol{v}}\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p}) = \begin{pmatrix}\nabla_{\boldsymbol{v}_0}\boldsymbol{R}(\boldsymbol{s}_0,\boldsymbol{p}) & 0 & \ldots & 0\\ 0 & \nabla_{\boldsymbol{v}_1}\boldsymbol{R}(\boldsymbol{s}_1,\boldsymbol{p}) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0\\ 0 & \ldots & 0 & \nabla_{\boldsymbol{v}_{\mathrm{N}}}\boldsymbol{R}(\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p})\end{pmatrix} = \begin{pmatrix}\begin{pmatrix}* & 0\end{pmatrix} & \ldots & 0 & \\ 0 & \begin{pmatrix}* & 0\end{pmatrix} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0\\ 0 & \ldots & 0 & \begin{pmatrix}*\end{pmatrix}\end{pmatrix}, \tag{4.15}$$

it results in a block-diagonal structure of the Jacobian of the residual vector $\boldsymbol{R}(\cdot)$ with respect to all variables $\boldsymbol{v}$. Whereas, the Jacobian with respect to model parameters $\boldsymbol{p}$ reads as follows

$$\nabla_{\boldsymbol{p}}\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p}) = \boldsymbol{\Sigma}^{-1}\begin{pmatrix}\nabla_{\boldsymbol{p}}\boldsymbol{h}_0(\boldsymbol{s}_0,\boldsymbol{p})^T & \nabla_{\boldsymbol{p}}\boldsymbol{h}_1(\boldsymbol{s}_1,\boldsymbol{p})^T & \ldots & \nabla_{\boldsymbol{p}}\boldsymbol{h}_{\mathrm{N}}(\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p})^T\end{pmatrix}^T, \tag{4.16}$$

with the Jacobians $\nabla_{\boldsymbol{p}}\boldsymbol{h}_{\mathrm{i}}(\boldsymbol{s}_{\mathrm{i}},\boldsymbol{p}) \in \mathbb{R}^{\mathrm{n_h}\times\mathrm{n_p}}$ of each measurement function $\boldsymbol{h}_{\mathrm{i}}(\cdot)$ with respect to model parameters $\boldsymbol{p}$. All other entries of $\nabla_{\boldsymbol{Z}}\boldsymbol{R}(\boldsymbol{s},\boldsymbol{p})$ are zero. In sum, if we recall the variables vector of the one-level NLP defined

by

$$\begin{aligned}\boldsymbol{Z} &:= \begin{pmatrix} \boldsymbol{z}^T & \boldsymbol{\lambda}^T & \boldsymbol{\mu}_{\overline{z}}^T & \boldsymbol{\mu}_{\underline{z}}^T & \boldsymbol{\alpha}^T & \boldsymbol{p}^T \end{pmatrix}^T \\ &= \left(\begin{array}{c|c|c|c|c|c|c|c} \boldsymbol{v}_0^T & \dots & \boldsymbol{v}_{\mathrm{N}}^T & d & \boldsymbol{w}^T & \boldsymbol{\lambda}^T & \boldsymbol{\mu}_{\overline{z}}^T & \boldsymbol{\mu}_{\underline{z}}^T & \boldsymbol{\alpha}^T & \boldsymbol{p}^T \end{array}\right)^T,\end{aligned}$$

with $\boldsymbol{v}_{\mathrm{i}} = \begin{pmatrix} \boldsymbol{s}_{\mathrm{i}}^T & \boldsymbol{q}_{\mathrm{i}}^T \end{pmatrix}^T$, $\mathrm{i} = 0, \dots, \mathrm{N}$, the Jacobian of the residual vector can be written in compact notation as

$$\nabla_{\boldsymbol{Z}} \boldsymbol{R} := \left(\begin{array}{c|c|c|c|c|c|c|c} \nabla_{\boldsymbol{v}} \boldsymbol{R}^T & 0 & 0 & 0 & 0 & 0 & 0 & \nabla_{\boldsymbol{p}} \boldsymbol{R}^T \end{array}\right)^T, \tag{4.17}$$

where the arguments of $\nabla_{\boldsymbol{v}} \boldsymbol{R}$ and $\nabla_{\boldsymbol{p}} \boldsymbol{R}$ are omitted for a clearer presentation.

### 4.3.2 Constraint Vector with Structure Exploitation of Gradient of Lower Level Lagrangian

We now consider the constraints of the NLP (4.13). The constraints (4.13d) - (4.13g) given above are mostly simple bounds on variables, whereas the constraint vector in (4.13c), e.g., includes the matching conditions that incorporate the result of integrations of the arising dynamic equations systems. The structure of this constraint vector is already given in subsection 4.2.1 in equation (4.1).

It remains to consider the gradient of the Lagrangian of the lower level NLP as it appears as constraint (4.13b) in the one-level NLP. Hence, for efficient calculations its special structure is now exploited. The Lagrangian is defined by

$$\nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{Z}) := \nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) - \nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p})^T \boldsymbol{\lambda} + \hat{\boldsymbol{\mu}}_{\overline{z}} - \hat{\boldsymbol{\mu}}_{\underline{z}},$$

where the vectors $\hat{\boldsymbol{\mu}}_{\overline{z}}$ and $\hat{\boldsymbol{\mu}}_{\underline{z}}$ are already defined in equation (4.10) of subsection 4.2.3, such that we concentrate on $\nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p})$ and $\nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z}, \boldsymbol{p})^T \boldsymbol{\lambda}$ in the following.

The lower level objective is a combination of Mayer type optimization criteria,

$$\boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) = \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} \cdot \boldsymbol{\phi}_{\mathrm{k}}^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{p}).$$

Hence, if we recall the variables of the lower level NLP to be

$$\boldsymbol{z} := \left(\begin{array}{c|c|c|c} \boldsymbol{v}_0^T & \dots & \boldsymbol{v}_{\mathrm{N}}^T & d & \boldsymbol{w}^T \end{array}\right)^T, \quad \text{with} \quad \boldsymbol{v}_{\mathrm{i}} = \begin{pmatrix} \boldsymbol{s}_{\mathrm{i}}^T & \boldsymbol{q}_{\mathrm{i}}^T \end{pmatrix}^T, \ \mathrm{i} = 0, \dots, \mathrm{N}, \tag{4.19}$$

the gradient $\nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p})$ has only non-zero entries for the multiple shooting state parameters $\boldsymbol{s}_{\mathrm{N}}$ at the last node

$$\boldsymbol{M}_{\mathrm{N}}^{\boldsymbol{v}_{\mathrm{N}}} := \begin{pmatrix} (\boldsymbol{M}_{\mathrm{N}}^{\boldsymbol{s}_{\mathrm{N}}})^T & 0 \end{pmatrix}^T \in \mathbb{R}^{\mathrm{n_x} + \mathrm{n_q}}, \quad \text{with} \quad \boldsymbol{M}_{\mathrm{N}}^{\boldsymbol{s}_{\mathrm{N}}} := \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} \cdot \frac{\partial \boldsymbol{\phi}_{\mathrm{k}}^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{p})}{\partial \boldsymbol{s}_{\mathrm{N}}},$$

and the duration parameter $d$

$$\boldsymbol{M}^{\boldsymbol{d}} := \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} \cdot \frac{\partial \boldsymbol{\phi}_{\mathrm{k}}^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}}, d, \boldsymbol{p})}{\partial d} \in \mathbb{R}.$$

In compact notation the gradient of the lower level objective can be written as

$$\nabla_{\boldsymbol{z}} \boldsymbol{\Phi}(\boldsymbol{z}, \boldsymbol{\alpha}, \boldsymbol{p}) := \left(\begin{array}{c|c|c} \overbrace{\begin{array}{ccc} 0 & \dots & 0 \end{array}}^{\boldsymbol{v}} \ (\boldsymbol{M}_{\mathrm{N}}^{\boldsymbol{v}_{\mathrm{N}}})^T & \overbrace{\boldsymbol{M}^{\boldsymbol{d}}}^{\boldsymbol{d}} & \overbrace{0}^{\boldsymbol{w}} \end{array}\right)^T. \tag{4.20}$$

A Lagrange type objective generally leads to a dense gradient of the objective.

The term $\nabla_{\boldsymbol{z}} \boldsymbol{C}(\boldsymbol{z},\boldsymbol{p})^T \boldsymbol{\lambda}$ is of special structure. It incorporates the Jacobian of the lower level constraint vector $\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p}) \in \mathbb{R}^{n_{Cec}+n_{Cic}}$ (4.3) with respect to the variables vector $\boldsymbol{z} \in \mathbb{R}^{n_z}$ in (4.19) multiplied by the corresponding Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{n_{Cec}+n_{Cic}}$.

We start with the matching conditions by defining

$$\boldsymbol{c}_{\mathrm{ms}}(\boldsymbol{v},d,\boldsymbol{p}) := \begin{pmatrix} \boldsymbol{x}(t_1^{\mathrm{ms}};\boldsymbol{s}_0,\boldsymbol{q}_0,d,\boldsymbol{p}) - \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{x}(t_{\mathrm{N}}^{\mathrm{ms}};\boldsymbol{s}_{\mathrm{N}-1},\boldsymbol{q}_{\mathrm{N}-1},d,\boldsymbol{p}) - \boldsymbol{s}_{\mathrm{N}} \end{pmatrix},$$

such that the equality constraint vector in (4.3) reads as

$$\boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v},d,\boldsymbol{p}) := \begin{pmatrix} \boldsymbol{c}_{\mathrm{ms}}(\boldsymbol{v},d,\boldsymbol{p}) \\ \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0,\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p}) \end{pmatrix} \in \mathbb{R}^{n_{Cec}}. \tag{4.21}$$

To give the special structure of the Jacobian of the matching conditions we consider the derivatives of the solutions of the IVPs, $\boldsymbol{x}(t_{i+1}^{\mathrm{ms}};\boldsymbol{s}_i,\boldsymbol{q}_i,d,\boldsymbol{p})$, also called *sensitivities*, with respect to the intermediate initial values $\boldsymbol{s}_i$, the control parameters $\boldsymbol{q}_i$, and the duration parameter $d$, such that we can define

$$\boldsymbol{G}_i^{\boldsymbol{s}} := \frac{\partial \boldsymbol{x}(t_{i+1}^{\mathrm{ms}};\boldsymbol{s}_i,\boldsymbol{q}_i,d,\boldsymbol{p})}{\partial \boldsymbol{s}_i}, \quad \boldsymbol{G}_i^{\boldsymbol{q}} := \frac{\partial \boldsymbol{x}(t_{i+1}^{\mathrm{ms}};\boldsymbol{s}_i,\boldsymbol{q}_i,d,\boldsymbol{p})}{\partial \boldsymbol{q}_i}, \quad \text{and} \quad \boldsymbol{G}_i^{\boldsymbol{d}} := \frac{\partial \boldsymbol{x}(t_{i+1}^{\mathrm{ms}};\boldsymbol{s}_i,\boldsymbol{q}_i,d,\boldsymbol{p})}{\partial d},$$

for each shooting node $i = 0,\dots,N-1$ with the dimensions $\boldsymbol{G}_i^{\boldsymbol{s}} \in (\mathbb{R}^{n_x \times n_x})$, $\boldsymbol{G}_i^{\boldsymbol{q}} \in (\mathbb{R}^{n_x \times n_q})$, and $\boldsymbol{G}_i^{\boldsymbol{d}} \in (\mathbb{R}^{n_x \times 1})$. Furthermore, the derivatives of each matching condition with respect to the parametrized state variables $\boldsymbol{s}_{i+1}$ are

$$\frac{\partial\left(\boldsymbol{x}(t_{i+1}^{\mathrm{ms}};\boldsymbol{s}_i,\boldsymbol{q}_i,d,\boldsymbol{p}) - \boldsymbol{s}_{i+1}\right)}{\partial \boldsymbol{s}_{i+1}} = -\boldsymbol{I}_{n_x}, \quad \forall i = 0,\dots,N-1,$$

where $\boldsymbol{I}_{n_x}$ denotes the identity matrix with $n_x$ diagonal entries. For a more compact presentation in (4.22) the previously defined quantities $\boldsymbol{G}_i^{\boldsymbol{s}}$ and $\boldsymbol{G}_i^{\boldsymbol{q}}$ are combined into submatrices

$$\boldsymbol{G}_i^{\boldsymbol{v}} := \begin{pmatrix} \boldsymbol{G}_i^{\boldsymbol{s}} & \boldsymbol{G}_i^{\boldsymbol{q}} \end{pmatrix} \in \mathbb{R}^{n_x \times (n_x+n_q)}, \quad \forall i = 0,\dots,N-1,$$

and the negative $(n_x \times n_x)$ identity matrix $-\boldsymbol{I}_{n_x}$ and the $(n_x \times n_q)$ null matrix $\boldsymbol{0}_{n_x \times n_q}$ into submatrix

$$\boldsymbol{I}^{\boldsymbol{v}} := \begin{pmatrix} -\boldsymbol{I}_{n_x} & \boldsymbol{0}_{n_x \times n_q} \end{pmatrix} \in \mathbb{R}^{n_x \times (n_x+n_q)}.$$

Then, the Jacobian of the matching conditions $\boldsymbol{c}_{\mathrm{ms}}(\cdot)$ with respect to variables $\hat{\boldsymbol{z}} = \begin{pmatrix} \boldsymbol{v}^T & d \end{pmatrix}^T$, where we excluded the slack variables $\boldsymbol{w}$ from the lower level NLP variables $\boldsymbol{z}$, can be written as

$$\nabla_{\hat{\boldsymbol{z}}} \boldsymbol{c}_{\mathrm{ms}}(\boldsymbol{v},d,\boldsymbol{p}) = \begin{pmatrix} \boldsymbol{G}_0^{\boldsymbol{v}} & \boldsymbol{I}^{\boldsymbol{v}} & 0 & \dots & 0 & \boldsymbol{G}_0^{\boldsymbol{d}} \\ 0 & \boldsymbol{G}_1^{\boldsymbol{v}} & \boldsymbol{I}^{\boldsymbol{v}} & \ddots & \vdots & \boldsymbol{G}_1^{\boldsymbol{d}} \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & \boldsymbol{G}_{N-1}^{\boldsymbol{v}} & \boldsymbol{I}^{\boldsymbol{v}} & \boldsymbol{G}_{N-1}^{\boldsymbol{d}} \end{pmatrix}. \tag{4.22}$$

The first N columns in (4.22) correspond to derivatives with respect to each $\boldsymbol{v}_i := \begin{pmatrix} \boldsymbol{s}_i^T & \boldsymbol{q}_i^T \end{pmatrix}^T$, $i = 0,\dots,N$, which are collected in $\boldsymbol{v} := \begin{pmatrix} \boldsymbol{v}_0^T & \dots & \boldsymbol{v}_N^T \end{pmatrix}^T$. The separability of the Jacobian of the matching conditions with respect to the variables $\boldsymbol{v}$ yields in a block-diagonal structure, where the only dense last column includes the derivatives with respect to the duration parameter $d$. Note that all derivatives of the matching conditions with respect to the slack variables $\boldsymbol{w} \in \mathbb{R}^{n_{Cic}}$ are zero. The needed sensitivities can be calculated by the methods described in subsection 2.2.2. If we have more than one model stage, at transitions between two stages instead of matching

conditions the stage transition conditions defined in equation (2.14) in subsection 2.2.1 have to be considered together with their derivative generation.

The equality boundary conditions are combined in $\boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p})$ and only depend on the parametrized states of the first and the last multiple shooting nodes, $\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}$, beside the model parameters $\boldsymbol{p}$. The derivatives with respect to the lower level variables $\boldsymbol{v}_{\mathrm{i}} := \begin{pmatrix} \boldsymbol{s}_{\mathrm{i}}^T & \boldsymbol{q}_{\mathrm{i}}^T \end{pmatrix}^T$, $\mathrm{i} = 0, \mathrm{N}$, can be written as

$$\boldsymbol{R}^{\boldsymbol{v}}_{\mathrm{ec,i}} := \begin{pmatrix} \boldsymbol{R}^{\boldsymbol{s}}_{\mathrm{ec,i}} & \boldsymbol{0}_{\mathrm{n_{ec}} \times \mathrm{n_q}} \end{pmatrix} \in \mathbb{R}^{\mathrm{n_{ec}} \times (\mathrm{n_x} + \mathrm{n_q})}, \quad \text{for } \mathrm{i} = 0, \mathrm{N},$$

with

$$\boldsymbol{R}^{\boldsymbol{s}}_{\mathrm{ec,0}} := \frac{\partial \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p})}{\partial \boldsymbol{s}_0}, \text{ and } \boldsymbol{R}^{\boldsymbol{s}}_{\mathrm{ec,N}} := \frac{\partial \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p})}{\partial \boldsymbol{s}_{\mathrm{N}}}.$$

Together with equation (4.22) the Jacobian of the equality constraints with respect to variables $\hat{\boldsymbol{z}}$, slack variables $\boldsymbol{w}$ excluded, reads

$$\nabla_{\hat{\boldsymbol{z}}} \boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v}, d, \boldsymbol{p}) = \left(\begin{array}{ccccc|c} \boldsymbol{G}^{\boldsymbol{v}}_0 & \boldsymbol{I}^{\boldsymbol{v}} & 0 & \dots & 0 & \boldsymbol{G}^{\boldsymbol{d}}_0 \\ 0 & \boldsymbol{G}^{\boldsymbol{v}}_1 & \boldsymbol{I}^{\boldsymbol{v}} & \ddots & \vdots & \boldsymbol{G}^{\boldsymbol{d}}_1 \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & \boldsymbol{G}^{\boldsymbol{v}}_{\mathrm{N}-1} & \boldsymbol{I}^{\boldsymbol{v}} & \boldsymbol{G}^{\boldsymbol{d}}_{\mathrm{N}-1} \\ \hline \boldsymbol{R}^{\boldsymbol{v}}_{\mathrm{ec,0}} & 0 & \dots & 0 & \boldsymbol{R}^{\boldsymbol{v}}_{\mathrm{ec,N}} & 0 \end{array}\right). \tag{4.23}$$

Further, we consider the Jabobian of the arising inequality constraint vector, which is defined in (4.4), with respect to the lower level variables $\boldsymbol{z}$. We define the derivatives of the mixed control-state constraints at each shooting node by

$$\boldsymbol{D}^{\boldsymbol{v}}_{\mathrm{i}} := \begin{pmatrix} \boldsymbol{D}^{\boldsymbol{s}}_{\mathrm{i}} & \boldsymbol{D}^{\boldsymbol{q}}_{\mathrm{i}} \end{pmatrix} \in \mathbb{R}^{\mathrm{n_{\tilde{c}}} \times (\mathrm{n_x} + \mathrm{n_q})}, \quad \forall \mathrm{i} = 0, \dots, \mathrm{N},$$

with

$$\boldsymbol{D}^{\boldsymbol{s}}_{\mathrm{i}} := \frac{\partial \tilde{\boldsymbol{c}}(\boldsymbol{s}_{\mathrm{i}}, \hat{\boldsymbol{u}}_{\mathrm{i}}(t^{\mathrm{ms}}_{\mathrm{i}}, \boldsymbol{q}_{\mathrm{i}}), \boldsymbol{p})}{\partial \boldsymbol{s}_{\mathrm{i}}} \text{ and } \boldsymbol{D}^{\boldsymbol{q}}_{\mathrm{i}} := \frac{\partial \tilde{\boldsymbol{c}}(\boldsymbol{s}_{\mathrm{i}}, \hat{\boldsymbol{u}}_{\mathrm{i}}(t^{\mathrm{ms}}_{\mathrm{i}}, \boldsymbol{q}_{\mathrm{i}}), \boldsymbol{p})}{\partial \boldsymbol{q}_{\mathrm{i}}}, \quad \forall \mathrm{i} = 0, \dots, \mathrm{N},$$

and the derivatives of the inequality boundary conditions with respect to $\boldsymbol{v}$ at the first and last multiple shooting node by

$$\boldsymbol{R}^{\boldsymbol{v}}_{\mathrm{ic,i}} := \begin{pmatrix} \boldsymbol{R}^{\boldsymbol{s}}_{\mathrm{ic,i}} & \boldsymbol{0}_{\mathrm{n_{\tilde{ic}}} \times \mathrm{n_q}} \end{pmatrix} \in \mathbb{R}^{\mathrm{n_{\tilde{ic}}} \times (\mathrm{n_x} + \mathrm{n_q})}, \quad \text{for } \mathrm{i} = 0, \mathrm{N},$$

with

$$\boldsymbol{R}^{\boldsymbol{s}}_{\mathrm{ic,0}} := \frac{\partial \tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p})}{\partial \boldsymbol{s}_0}, \text{ and } \boldsymbol{R}^{\boldsymbol{s}}_{\mathrm{ic,N}} := \frac{\partial \tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_{\mathrm{N}}, \boldsymbol{p})}{\partial \boldsymbol{s}_{\mathrm{N}}}.$$

Then the Jacobian of the inequality constraint vector $\boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v}, \boldsymbol{p})$ with respect to variables $\hat{\boldsymbol{z}}$ can be written as

$$\nabla_{\hat{\boldsymbol{z}}} \boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v}, \boldsymbol{p}) = \left(\begin{array}{ccccc|c} \boldsymbol{D}^{\boldsymbol{v}}_0 & 0 & \dots & & 0 & 0 \\ 0 & \boldsymbol{D}^{\boldsymbol{v}}_1 & \ddots & & \vdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ & & & & 0 & \\ 0 & & \dots & 0 & \boldsymbol{D}^{\boldsymbol{v}}_{\mathrm{N}} & 0 \\ \hline \boldsymbol{R}^{\boldsymbol{v}}_{\mathrm{ic,0}} & 0 & \dots & 0 & \boldsymbol{R}^{\boldsymbol{v}}_{\mathrm{ic,N}} & 0 \end{array}\right). \tag{4.24}$$

Because of the decoupled control path constraints, a block-diagonal substructure can be exploited. Note that all derivatives with respect to model parameter $d$ are zero and appear in the last column of (4.24).

It remains to consider the derivatives of the total constraint vector $\boldsymbol{C}(\cdot)$ with respect to the introduced slack variables $\boldsymbol{w}$. Because their only contribution is the subtraction from the inequality constraint vector $\boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v},\boldsymbol{p})-\boldsymbol{w}$, with $\boldsymbol{w}\geq 0$, see definition (4.3), the Jacobian is the negative identity matrix $-\boldsymbol{I}_{\mathrm{n}_{\mathrm{C_{ic}}}}$ with $\mathrm{n}_{\mathrm{C_{ic}}}$ diagonal entries and we have

$$\nabla_{\boldsymbol{w}}\left(\boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v},\boldsymbol{p})-\boldsymbol{w}\right):=-\boldsymbol{I}_{\mathrm{n}_{\mathrm{C_{ic}}}}.$$

Then the Jacobian of the total constraint vector with respect to all lower level NLP variables $\boldsymbol{z}$ can be written in compact notation as

$$\nabla_{\boldsymbol{z}}\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p})=\left(\begin{array}{c|c}\nabla_{\hat{\boldsymbol{z}}}\boldsymbol{C}_{\mathrm{ec}}(\boldsymbol{v},d,\boldsymbol{p}) & 0\\ \hline \nabla_{\hat{\boldsymbol{z}}}\boldsymbol{C}_{\mathrm{ic}}(\boldsymbol{v},\boldsymbol{p}) & -\boldsymbol{I}_{\mathrm{n}_{\mathrm{C_{ic}}}}\end{array}\right), \tag{4.25}$$

with the given structure exploitation in equations (4.23) and (4.24). To provide all summands in the gradient of the Lagrangian of the lower level problem, the remaining quantity $\nabla_{\boldsymbol{z}}\boldsymbol{C}(\boldsymbol{z},\boldsymbol{p})^T\boldsymbol{\lambda}$ can now easily be computed with the investigated special structure of (4.25).

### 4.3.3 Constraint Jacobian of One-Level NLP

The exploitation of the sparsity pattern of the constraint Jacobian provides a great contribution for an efficient implementation of the proposed method, if used by standard NLP solvers. Jacobians of the constraints on the objective weights (4.13f) and model parameters (4.13g) are easily computed and, hence, not considered at this point. We focus on the general structure of the Jacobian of the constraints related to the one-level NLP (4.13).

Similar to equation (4.9) we again introduce matrices $\boldsymbol{I}_{\overline{\mathcal{W}}}\in\mathbb{R}^{|\overline{\mathcal{W}}|\times \mathrm{n}_{\mathrm{z}}}$ and $\boldsymbol{I}_{\underline{\mathcal{W}}}\in\mathbb{R}^{|\underline{\mathcal{W}}|\times \mathrm{n}_{\mathrm{z}}}$, which are composed of standard unit vectors $\boldsymbol{e}_{\mathrm{r}}\in\mathbb{R}^{\mathrm{n}_{\mathrm{z}}}$ corresponding to indices $\mathrm{r}\in\overline{\mathcal{W}}$ and $\mathrm{r}\in\underline{\mathcal{W}}$, respectively. Then the constraints can be combined into one vector

$$\hat{\boldsymbol{C}}(\boldsymbol{Z})=\begin{pmatrix}\nabla_{\boldsymbol{z}}\mathcal{L}(\boldsymbol{Z})\\ \boldsymbol{C}(\boldsymbol{z},\boldsymbol{p})\\ \boldsymbol{I}_{\overline{\mathcal{W}}}^T\cdot\left(\overline{\boldsymbol{z}}-\boldsymbol{z}\right)\\ \boldsymbol{I}_{\underline{\mathcal{W}}}^T\cdot\left(\boldsymbol{z}-\underline{\boldsymbol{z}}\right)\end{pmatrix}. \tag{4.26}$$

The sparse structure of the constraint Jacobian is already visible when constructed as follows

$$\nabla_{\boldsymbol{Z}}\hat{\boldsymbol{C}}=\left(\begin{array}{c|c|c|c|c|c}\nabla_{\boldsymbol{z}}^2\mathcal{L} & \nabla_{\boldsymbol{z}\boldsymbol{\lambda}}^2\mathcal{L} & \nabla_{\boldsymbol{z}\boldsymbol{\mu}_{\overline{\mathrm{z}}}}^2\mathcal{L} & \nabla_{\boldsymbol{z}\boldsymbol{\mu}_{\underline{\mathrm{z}}}}^2\mathcal{L} & \nabla_{\boldsymbol{z}\boldsymbol{\alpha}}^2\mathcal{L} & \nabla_{\boldsymbol{z}\boldsymbol{p}}^2\mathcal{L}\\ \hline \nabla_{\boldsymbol{z}}\boldsymbol{C} & 0 & 0 & 0 & 0 & \nabla_{\boldsymbol{p}}\boldsymbol{C}\\ \hline \nabla_{\boldsymbol{z}}\left(\boldsymbol{I}_{\overline{\mathcal{W}}}^T\cdot\left(\overline{\boldsymbol{z}}-\boldsymbol{z}\right)\right) & 0 & 0 & 0 & 0 & 0\\ \hline \nabla_{\boldsymbol{z}}\left(\boldsymbol{I}_{\underline{\mathcal{W}}}^T\cdot\left(\boldsymbol{z}-\underline{\boldsymbol{z}}\right)\right) & 0 & 0 & 0 & 0 & 0\end{array}\right), \tag{4.27}$$

with respect to the variables vector $\boldsymbol{Z}:=\left(\boldsymbol{z}^T \;\middle|\; \boldsymbol{\lambda}^T \;\middle|\; \boldsymbol{\mu}_{\overline{\mathrm{z}}}^T \;\middle|\; \boldsymbol{\mu}_{\underline{\mathrm{z}}}^T \;\middle|\; \boldsymbol{\alpha}^T \;\middle|\; \boldsymbol{p}^T\right)^T$. Note that we omitted the arguments due to a clearer presentation. This is also done in the following. If we take a closer look at each submatrix of the Jacobian $\nabla_{\boldsymbol{Z}}\hat{\boldsymbol{C}}$ stated in (4.27), the structure of submatrix $\nabla_{\boldsymbol{z}}\boldsymbol{C}$ was already exploited previously, and other submatices are simply the positive or negative matrices $\boldsymbol{I}_{\overline{\mathcal{W}}}$ and $\boldsymbol{I}_{\overline{\mathcal{W}}}$, or its transposed counterpart. Additionally, there are some dense blocks, whereas others have special structures which are provided in the following, e.g. the Hessian of the Lagrangian of the lower level NLP denoted by $\nabla_{\boldsymbol{z}}^2\mathcal{L}$.

We start with the Jacobians corresponding to the submatrices of the last two rows in (4.27) with respect to the lower level NLP variables $\boldsymbol{z}$. These can be written as

$$\nabla_{\boldsymbol{z}}\left(\boldsymbol{I}_{\overline{\mathcal{W}}}^{T}\cdot\left(\overline{\boldsymbol{z}}-\boldsymbol{z}\right)\right) = -\boldsymbol{I}_{\overline{\mathcal{W}}}^{T},$$
$$\nabla_{\boldsymbol{z}}\left(\boldsymbol{I}_{\underline{\mathcal{W}}}^{T}\cdot\left(\boldsymbol{z}-\underline{\boldsymbol{z}}\right)\right) = \quad \boldsymbol{I}_{\underline{\mathcal{W}}}^{T}.$$

Further, the submatrices $\nabla^2_{\boldsymbol{z}\boldsymbol{\mu}_{\overline{z}}}\mathcal{L}$ and $\nabla^2_{\boldsymbol{z}\boldsymbol{\mu}_{\underline{z}}}\mathcal{L}$ are of similar structure and can be derived as

$$\nabla^2_{\boldsymbol{z}\boldsymbol{\mu}_{\overline{z}}}\mathcal{L} = \quad \boldsymbol{I}_{\overline{\mathcal{W}}},$$
$$\nabla^2_{\boldsymbol{z}\boldsymbol{\mu}_{\underline{z}}}\mathcal{L} = -\boldsymbol{I}_{\underline{\mathcal{W}}}.$$

The submatrix $\nabla^2_{\boldsymbol{z}\boldsymbol{\lambda}}\mathcal{L}$ is simply the transposed negative Jacobian of the equality constraints of the lower level NLP $\nabla_{\boldsymbol{z}}\boldsymbol{C}$, defined by

$$\nabla^2_{\boldsymbol{z}\boldsymbol{\lambda}}\mathcal{L} = -\nabla_{\boldsymbol{z}}\boldsymbol{C}^{T}.$$

$\nabla_{\boldsymbol{z}}\boldsymbol{C} \in \mathbb{R}^{\mathrm{n}_{\mathrm{C_{ec}}}\mathrm{n}_{\mathrm{z}}}$ was previously exploited in (4.25) with the given structures in (4.23) and (4.24).

Before we consider the special structure of the Hessian of the Lagrangian of the lower level NLP denoted by $\nabla^2_{\boldsymbol{z}}\mathcal{L}$, we take a look at the gradients with respect to the weights $\boldsymbol{\alpha}$ and the model parameters $\boldsymbol{p}$. We start with the only contribution of the objective weight in $\nabla^2_{\boldsymbol{z}\boldsymbol{\alpha}}\mathcal{L} \in \mathbb{R}^{\mathrm{n}_{\mathrm{z}}\times\mathrm{n}_{\boldsymbol{\alpha}}}$. Because the gradient $\nabla_{\boldsymbol{z}}\boldsymbol{\Phi}(\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{p})$ defined in (4.20) has only non-zero entries for the multiple shooting state parameters $\boldsymbol{s}_{\mathrm{N}}$ at the last node and the duration parameters $d$ we have

$$\nabla^2_{\boldsymbol{z}\boldsymbol{\alpha}}\mathcal{L} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ \frac{\partial\boldsymbol{\phi}_1^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{p})}{\partial\boldsymbol{s}_{\mathrm{N}}} & \frac{\partial\boldsymbol{\phi}_2^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{p})}{\partial\boldsymbol{s}_{\mathrm{N}}} & \dots & \frac{\partial\boldsymbol{\phi}_{\mathrm{n_M}}^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{p})}{\partial\boldsymbol{s}_{\mathrm{N}}} \\ 0 & 0 & \dots & 0 \\ \frac{\partial\boldsymbol{\phi}_1^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{p})}{\partial d} & \frac{\partial\boldsymbol{\phi}_2^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{p})}{\partial d} & \dots & \frac{\partial\boldsymbol{\phi}_{\mathrm{n_M}}^{\mathrm{M}}(\boldsymbol{s}_{\mathrm{N}},d,\boldsymbol{p})}{\partial d} \\ 0 & 0 & \dots & 0 \end{pmatrix}. \tag{4.30}$$

The contributions of model parameters $\boldsymbol{p}$ enter the constraint Jacobian in submatrices $\nabla^2_{\boldsymbol{z}\boldsymbol{p}}\mathcal{L}$ and $\nabla_{\boldsymbol{p}}\boldsymbol{C}$ with the dimensions $\mathbb{R}^{\mathrm{n}_{\mathrm{z}}\times\mathrm{n}_{\mathrm{p}}}$ and $\mathbb{R}^{(\mathrm{n}_{\mathrm{C_{ec}}}+\mathrm{n}_{\mathrm{C_{ic}}})\times\mathrm{n}_{\mathrm{p}}}$, respectively. We start with the latter one and define according to the investigation of the structure of $\nabla_{\boldsymbol{z}}\boldsymbol{C}$ in (4.25) the derivatives related to the arising matching conditions in (4.21) by

$$\boldsymbol{G}_{\mathrm{i}}^{\boldsymbol{p}} := \frac{\partial\boldsymbol{x}(t_{\mathrm{i+1}}^{\mathrm{ms}};\boldsymbol{s}_{\mathrm{i}},\boldsymbol{q}_{\mathrm{i}},d,\boldsymbol{p})}{\partial\boldsymbol{p}} \in \mathbb{R}^{\mathrm{n_x}\times\mathrm{n_p}},$$

for each shooting node $\mathrm{i} = 0,\dots,\mathrm{N}-1$, and the derivatives of the equality and inequality boundary conditions which appear in the sub vectors (4.21) and (4.4) by

$$\boldsymbol{R}_{\mathrm{ec}}^{\boldsymbol{p}} := \frac{\partial\boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0,\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p})}{\partial\boldsymbol{p}}, \quad \text{and} \quad \boldsymbol{R}_{\mathrm{ic}}^{\boldsymbol{p}} := \frac{\partial\tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0,\boldsymbol{s}_{\mathrm{N}},\boldsymbol{p})}{\partial\boldsymbol{p}},$$

respectively, with the dimensions $\boldsymbol{R}_{\mathrm{ec}}^{\boldsymbol{p}} \in \mathbb{R}^{\mathrm{n_{ec}}\times\mathrm{n_p}}$ and $\boldsymbol{R}_{\mathrm{ic}}^{\boldsymbol{p}} \in \mathbb{R}^{\mathrm{n_{\tilde{ic}}}\times\mathrm{n_p}}$. Furthermore, the derivatives of the mixed control-state constraints at each shooting node are denoted by

$$\boldsymbol{D}_{\mathrm{i}}^{\boldsymbol{p}} := \frac{\partial\tilde{\boldsymbol{c}}(\boldsymbol{s}_{\mathrm{i}},\hat{\boldsymbol{u}}_{\mathrm{i}}(t_{\mathrm{i}}^{\mathrm{ms}},\boldsymbol{q}_{\mathrm{i}}),\boldsymbol{p})}{\partial\boldsymbol{p}} \in \mathbb{R}^{\mathrm{n_{\tilde{c}}}\times\mathrm{n_p}}, \quad \forall\mathrm{i} = 0,\dots,\mathrm{N}.$$

In sum, we have a dense representation for the Jacobian of the whole lower level constraint vector with respect to model paramaters defined by

$$\nabla_{\boldsymbol{p}} \boldsymbol{C} = \left( (\boldsymbol{G}_0^{\boldsymbol{p}})^T \quad \dots \quad (\boldsymbol{G}_{\mathrm{N}-1}^{\boldsymbol{p}})^T \;\middle|\; (\boldsymbol{R}_{\mathrm{ec}}^{\boldsymbol{p}})^T \;\middle|\; (\boldsymbol{D}_0^{\boldsymbol{p}})^T \quad \dots \quad (\boldsymbol{D}_{\mathrm{N}}^{\boldsymbol{p}})^T \;\middle|\; (\boldsymbol{R}_{\mathrm{ic}}^{\boldsymbol{p}})^T \right)^T. \tag{4.31}$$

Similar to the above case, because of the global model parameter $\boldsymbol{p}$ the submatrix $\nabla^2_{\boldsymbol{zp}}\mathcal{L} \in \mathbb{R}^{\mathrm{n_z}\times\mathrm{n_p}}$ is dense and has no special structure which can be exploited. Its computation is performed by generating the derivatives of the gradient of the lower level Lagrangian (4.11) with respect to $\boldsymbol{p}$. If we use the notations $\nabla^2_{\boldsymbol{v}_\mathrm{i}\boldsymbol{p}}\mathcal{L} \in \mathbb{R}^{(\mathrm{n_x}+\mathrm{n_q})\times\mathrm{n_p}}$, $\forall \mathrm{i} = 0,\dots,\mathrm{N}$, and $\nabla^2_{d\boldsymbol{p}}\mathcal{L} \in \mathbb{R}^{1\times\mathrm{n_p}}$, we have

$$\nabla^2_{\boldsymbol{zp}}\mathcal{L} = \left( \nabla^2_{\boldsymbol{v}_0\boldsymbol{p}}\mathcal{L}^T \quad \dots \quad \nabla^2_{\boldsymbol{v}_\mathrm{N}\boldsymbol{p}}\mathcal{L}^T \;\middle|\; \nabla^2_{d\boldsymbol{p}}\mathcal{L}^T \;\middle|\; 0 \right)^T, \tag{4.32}$$

with respect to $\boldsymbol{z} := \left( \boldsymbol{v}_0^T \quad \dots \quad \boldsymbol{v}_\mathrm{N}^T \;\middle|\; d \;\middle|\; \boldsymbol{w}^T \right)^T$ where $\boldsymbol{v}_\mathrm{i} = \left( \boldsymbol{s}_\mathrm{i}^T \quad \boldsymbol{q}_\mathrm{i}^T \right)^T$, $\mathrm{i} = 0,\dots,\mathrm{N}$. Note that all second derivatives with respect to the slack variables $\boldsymbol{w}$ are zero. In contrast to the dense submatrix $\nabla^2_{\boldsymbol{zp}}\mathcal{L}$ the Hessian of the Lagrangian of the lower level NLP, denoted by $\nabla^2_{\boldsymbol{z}}\mathcal{L} \in \mathbb{R}^{\mathrm{n_z}\times\mathrm{n_z}}$, comprises a very special structure because of its separability with respect to the variables $\boldsymbol{v}_\mathrm{i}$ on each inner node $\mathrm{i} = 1,\dots,\mathrm{N}-1$. To demonstrate this we introduce the Lagrange multipliers defined as $\boldsymbol{\lambda} := \left( (\boldsymbol{\lambda}^{\mathrm{ms}})^T \quad (\boldsymbol{\lambda}^{\mathrm{ec}})^T \quad (\boldsymbol{\lambda}^{\tilde{c}})^T \quad (\boldsymbol{\lambda}^{\mathrm{ic}})^T \right)^T$, which correspond to the matching conditions, the equality boundary conditions, the mixed control-state constraints, and the inequality boundary conditions, respectively, and consider the Lagrangian of the lower level problem in more detail by

$$\begin{aligned}
\mathcal{L}(\boldsymbol{Z}) :=& \boldsymbol{\Phi}(\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{p}) - \boldsymbol{\lambda}^T \boldsymbol{C}(\boldsymbol{z},\boldsymbol{p}) - \sum_{\mathrm{r}\in\overline{\mathcal{W}}} \mu_\mathrm{r}\left(\overline{z}_\mathrm{r} - z_\mathrm{r}\right) - \sum_{\mathrm{r}\in\underline{\mathcal{W}}} \mu_\mathrm{r}\left(z_\mathrm{r} - \underline{z}_\mathrm{r}\right) \\
=& \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_\mathrm{k} \cdot \boldsymbol{\phi}_\mathrm{k}^{\mathrm{M}}(\boldsymbol{s}_\mathrm{N}, d, \boldsymbol{p}) \\
& - \sum_{\mathrm{i}=0}^{\mathrm{N}-1} (\boldsymbol{\lambda}_\mathrm{i}^{\mathrm{ms}})^T (\boldsymbol{x}(t_{\mathrm{i}+1}^{\mathrm{ms}}; \boldsymbol{s}_\mathrm{i}, \boldsymbol{q}_\mathrm{i}, d, \boldsymbol{p}) - \boldsymbol{s}_{\mathrm{i}+1}) - (\boldsymbol{\lambda}^{\mathrm{ec}})^T \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_\mathrm{N}, \boldsymbol{p}) \\
& - \sum_{\mathrm{i}=0}^{\mathrm{N}} (\boldsymbol{\lambda}_\mathrm{i}^{\tilde{c}})^T (\tilde{\boldsymbol{c}}(\boldsymbol{s}_\mathrm{i}, \hat{\boldsymbol{u}}_\mathrm{i}(t_\mathrm{i}^{\mathrm{ms}}, \boldsymbol{q}_\mathrm{i}), \boldsymbol{p})) - (\boldsymbol{\lambda}^{\mathrm{ic}})^T \tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_\mathrm{N}, \boldsymbol{p}) \\
& + \left( (\boldsymbol{\lambda}^{\tilde{c}})^T \quad (\boldsymbol{\lambda}^{\mathrm{ic}})^T \right) \boldsymbol{w} - \sum_{\mathrm{r}\in\overline{\mathcal{W}}} \mu_\mathrm{r}\left(\overline{z}_\mathrm{r} - z_\mathrm{r}\right) - \sum_{\mathrm{r}\in\underline{\mathcal{W}}} \mu_\mathrm{r}\left(z_\mathrm{r} - \underline{z}_\mathrm{r}\right) \\
=& \sum_{\mathrm{i}=1}^{\mathrm{N}-1} \mathcal{L}_\mathrm{i} + \mathcal{L}_{0,\mathrm{N}} + \mathcal{L}_{\mathrm{lin}}
\end{aligned} \tag{4.33}$$

with the following definitions

$$\begin{aligned}
\mathcal{L}_\mathrm{i} :=& -(\boldsymbol{\lambda}_\mathrm{i}^{\mathrm{ms}})^T \boldsymbol{x}(t_{\mathrm{i}+1}^{\mathrm{ms}}; \boldsymbol{s}_\mathrm{i}, \boldsymbol{q}_\mathrm{i}, d, \boldsymbol{p}) - (\boldsymbol{\lambda}_\mathrm{i}^{\tilde{c}})^T \tilde{\boldsymbol{c}}(\boldsymbol{s}_\mathrm{i}, \hat{\boldsymbol{u}}_\mathrm{i}(t_\mathrm{i}^{\mathrm{ms}}, \boldsymbol{q}_\mathrm{i}), \boldsymbol{p}) \\
\mathcal{L}_{0,\mathrm{N}} :=& \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_\mathrm{k} \cdot \boldsymbol{\phi}_\mathrm{k}^{\mathrm{M}}(\boldsymbol{s}_\mathrm{N}, d, \boldsymbol{p}) \\
& - (\boldsymbol{\lambda}_0^{\mathrm{ms}})^T \boldsymbol{x}(t_1^{\mathrm{ms}}; \boldsymbol{s}_0, \boldsymbol{q}_0, d, \boldsymbol{p}) - (\boldsymbol{\lambda}^{\mathrm{ec}})^T \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{s}_0, \boldsymbol{s}_\mathrm{N}, \boldsymbol{p}) \\
& - (\boldsymbol{\lambda}_0^{\tilde{c}})^T \tilde{\boldsymbol{c}}(\boldsymbol{s}_0, \hat{\boldsymbol{u}}_0(t_0^{\mathrm{ms}}, \boldsymbol{q}_0), \boldsymbol{p}) - (\boldsymbol{\lambda}_\mathrm{N}^{\tilde{c}})^T \tilde{\boldsymbol{c}}(\boldsymbol{s}_\mathrm{N}, \hat{\boldsymbol{u}}_\mathrm{N}(t_\mathrm{N}^{\mathrm{ms}}, \boldsymbol{q}_\mathrm{N}), \boldsymbol{p}) \\
& - (\boldsymbol{\lambda}^{\mathrm{ic}})^T \tilde{\boldsymbol{r}}_{\mathrm{ic}}(\boldsymbol{s}_0, \boldsymbol{s}_\mathrm{N}, \boldsymbol{p}) \\
\mathcal{L}_{\mathrm{lin}} :=& \sum_{\mathrm{i}=0}^{\mathrm{N}-1} (\boldsymbol{\lambda}_\mathrm{i}^{\mathrm{ms}})^T \boldsymbol{s}_{\mathrm{i}+1} + \left( (\boldsymbol{\lambda}^{\tilde{c}})^T \quad (\boldsymbol{\lambda}^{\mathrm{ic}})^T \right) \boldsymbol{w} - \sum_{\mathrm{r}\in\overline{\mathcal{W}}} \mu_\mathrm{r}\left(\overline{z}_\mathrm{r} - z_\mathrm{r}\right) - \sum_{\mathrm{r}\in\underline{\mathcal{W}}} \mu_\mathrm{r}\left(z_\mathrm{r} - \underline{z}_\mathrm{r}\right).
\end{aligned}$$

With the compact notation in (4.33) it is obvious that the only coupling between two shooting variables appears between the first and the last state variables. With respect to the inner multiple shooting nodes the corresponding Hessian blocks are separable. Furthermore, all contributions in the Hessian of the lower level Lagrangian

related to $\mathcal{L}_{\text{lin}}$ are zero because of the linear dependency of the state parameters $\boldsymbol{s}_\text{i}$, the slack variables $\boldsymbol{w}$ and the NLP variables $z_\text{r} \forall \text{r} \in \mathcal{W}$ related to the working set. However, the Hessian blocks with respect to the duration parameter $d$ are all dense.

We now exploit the structure of the block-diagonal Hessian with respect to the variables $\boldsymbol{v}_\text{i} = \begin{pmatrix} \boldsymbol{s}_\text{i}^T & \boldsymbol{q}_\text{i}^T \end{pmatrix}^T$ as a result of the separability. Each $(\text{n}_\text{x} + \text{n}_\text{q}) \times (\text{n}_\text{x} + \text{n}_\text{q})$-block is defined by

$$\nabla^2_{\boldsymbol{v}_\text{i}} \mathcal{L}_\text{i} = \begin{pmatrix} \frac{\partial^2}{\partial \boldsymbol{s}_\text{i}^2} \mathcal{L}_\text{i} & \frac{\partial^2}{\partial \boldsymbol{q}_\text{i} \partial \boldsymbol{s}_\text{i}} \mathcal{L}_\text{i} \\ \frac{\partial^2}{\partial \boldsymbol{s}_\text{i} \partial \boldsymbol{q}_\text{i}} \mathcal{L}_\text{i} & \frac{\partial^2}{\partial \boldsymbol{q}_\text{i}^2} \mathcal{L}_\text{i} \end{pmatrix}, \quad \text{for all} \;\; \text{i} = 1, \dots, \text{N} - 1.$$

Furthermore, we define the following blocks for the first and the last shooting node

$$\nabla^2_{\boldsymbol{v}_\text{i}} \mathcal{L}_{0,\text{N}} = \begin{pmatrix} \frac{\partial^2}{\partial \boldsymbol{s}_\text{i}^2} \mathcal{L}_{0,\text{N}} & \frac{\partial^2}{\partial \boldsymbol{q}_\text{i} \partial \boldsymbol{s}_\text{i}} \mathcal{L}_{0,\text{N}} \\ \frac{\partial^2}{\partial \boldsymbol{s}_\text{i} \partial \boldsymbol{q}_\text{i}} \mathcal{L}_{0,\text{N}} & \frac{\partial^2}{\partial \boldsymbol{q}_\text{i}^2} \mathcal{L}_{0,\text{N}} \end{pmatrix}, \quad \text{i} = 0, \text{N},$$

and

$$\nabla^2_{\boldsymbol{v}_\text{i} \boldsymbol{v}_\text{j}} \mathcal{L}_{0,\text{N}} = \begin{pmatrix} \frac{\partial^2}{\partial \boldsymbol{s}_\text{j} \partial \boldsymbol{s}_\text{i}} \mathcal{L}_{0,\text{N}} & \frac{\partial^2}{\partial \boldsymbol{q}_\text{j} \partial \boldsymbol{s}_\text{i}} \mathcal{L}_{0,\text{N}} \\ \frac{\partial^2}{\partial \boldsymbol{s}_\text{j} \partial \boldsymbol{q}_\text{i}} \mathcal{L}_{0,\text{N}} & \frac{\partial^2}{\partial \boldsymbol{q}_\text{j} \partial \boldsymbol{q}_\text{i}} \mathcal{L}_{0,\text{N}} \end{pmatrix}, \quad \text{i}, \text{j} \in \{0, \text{N}\}, \text{i} \neq \text{j}.$$

For the dense part of the Hessian with respect to the duration parameters we use the notations $\nabla^2_{\boldsymbol{v}_\text{i} d} \mathcal{L} = \nabla^2_{d \boldsymbol{v}_\text{i}} \mathcal{L}^T \in \mathbb{R}^{(\text{n}_\text{x} + \text{n}_\text{q}) \times 1}$ and $\nabla^2_d \mathcal{L} \in \mathbb{R}$. In sum, this yields to the following symmetric structure of the Hessian of the Lagrangian of the lower level NLP with respect to the lower level decision variables $\boldsymbol{z}$

$$\nabla^2_{\boldsymbol{z}} \mathcal{L} = \begin{pmatrix} \nabla^2_{\boldsymbol{v}_0} \mathcal{L}_{0,\text{N}} & 0 & \dots & 0 & \nabla^2_{\boldsymbol{v}_0 \boldsymbol{v}_\text{N}} \mathcal{L}_{0,\text{N}} & \nabla^2_{\boldsymbol{v}_0 d} \mathcal{L} & 0 \\ 0 & \nabla^2_{\boldsymbol{v}_1} \mathcal{L}_1 & \ddots & \vdots & 0 & & \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \dots & 0 & \nabla^2_{\boldsymbol{v}_{\text{N}-1}} \mathcal{L}_{\text{N}-1} & 0 & & \vdots \\ \nabla^2_{\boldsymbol{v}_\text{N} \boldsymbol{v}_0} \mathcal{L}_{0,\text{N}} & 0 & \dots & 0 & \nabla^2_{\boldsymbol{v}_\text{N}} \mathcal{L}_{0,\text{N}} & & \\ & & & & & \nabla^2_{\boldsymbol{v}_0 d} \mathcal{L} & \\ \nabla^2_{d \boldsymbol{v}_0} \mathcal{L} & & \dots & & \nabla^2_{d \boldsymbol{v}_\text{N}} \mathcal{L} & \nabla^2_d \mathcal{L} & \\ 0 & & & \dots & & & 0 \end{pmatrix}. \tag{4.35}$$

Together with the definitions and the exploited structures given above for the Hessian of the lower level Lagrangian $\nabla^2_{\boldsymbol{z}} \mathcal{L}$ in (4.35), the Jacobians of the lower level constraint vector $\nabla_{\boldsymbol{z}} \boldsymbol{C}$ in (4.25) and $\nabla_{\boldsymbol{p}} \boldsymbol{C}$ in (4.31) with respect to variables $\boldsymbol{z}$ and $\boldsymbol{p}$, respectively, the Jacobians of the gradient of the Lagrangian of the lower level $\nabla^2_{\boldsymbol{z}\boldsymbol{\alpha}} \mathcal{L}$ in (4.30) and $\nabla^2_{\boldsymbol{z}\boldsymbol{p}} \mathcal{L}$ in (4.32) with respect to weights $\boldsymbol{\alpha}$ and parameters $\boldsymbol{p}$, respectively, we can now investigate the structure of the whole Jacobian of the one-level NLP first defined in (4.27) as follows

$$\nabla_{\boldsymbol{Z}} \hat{\boldsymbol{C}}(\boldsymbol{Z}) = \left( \begin{array}{c|c|c|c|c|c} \nabla^2_{\boldsymbol{z}} \mathcal{L} & -\nabla_{\boldsymbol{z}} \boldsymbol{C}^T & \boldsymbol{I}_{\overline{\mathcal{W}}} & -\boldsymbol{I}_{\underline{\mathcal{W}}} & \nabla^2_{\boldsymbol{z}\boldsymbol{\alpha}} \mathcal{L} & \nabla^2_{\boldsymbol{z}\boldsymbol{p}} \mathcal{L} \\ \hline \nabla_{\boldsymbol{z}} \boldsymbol{C} & 0 & 0 & 0 & 0 & \nabla_{\boldsymbol{p}} \boldsymbol{C} \\ \hline -\boldsymbol{I}^T_{\overline{\mathcal{W}}} & 0 & 0 & 0 & 0 & 0 \\ \hline \boldsymbol{I}^T_{\underline{\mathcal{W}}} & 0 & 0 & 0 & 0 & 0 \end{array} \right), \tag{4.36}$$

with respect to the variables vector $\boldsymbol{Z} := \begin{pmatrix} \boldsymbol{z}^T & \vert & \boldsymbol{\lambda}^T & \vert & \boldsymbol{\mu}_{\overline{z}}^T & \vert & \boldsymbol{\mu}_{\underline{z}}^T & \vert & \boldsymbol{\alpha}^T & \vert & \boldsymbol{p}^T \end{pmatrix}^T$, where $\boldsymbol{I}_{\overline{\mathcal{W}}} \in \mathbb{R}^{|\overline{\mathcal{W}}| \times \text{n}_\text{z}}$ and $\boldsymbol{I}_{\underline{\mathcal{W}}} \in \mathbb{R}^{|\underline{\mathcal{W}}| \times \text{n}_\text{z}}$, are composed of standard unit vectors $\boldsymbol{e}_\text{r} \in \mathbb{R}^{\text{n}_\text{z}}$ corresponding to indices $\text{r} \in \overline{\mathcal{W}}$ and $\text{r} \in \underline{\mathcal{W}}$, respectively.

### 4.3.4 Approximation of Hessian of Lagrangian

In interior-point methods as well as in SQP methods the Hessian of the Lagrangian of the one-level NLP (4.13) appears in the basic algorithms. Replacing the Hessian by a quasi-Newton approximation might be beneficial due to several reasons. For instance, computations of exact Hessians can be very expensive, especially when using finite differences. Especially, in large-scale NLPs, which might result from modeling the human gait in detail, this could have a tremendous impact on the efficiency of the proposed method. Furthermore, the Hessian of the Lagrangian might get indefinite when computed exactly, which causes difficulties in finding a solution. However, with an adequate, computationally less expensive quasi-Newton approximation one can ensure positive definiteness of the Hessian. BFGS update formulas, see Broyden [29], Fletcher [57], Goldfarb [73], and Shanno [158] , or limited-memory variants, denoted by L-BFGS, for large-scale and sparse NLPs are prominent choices. We refer to the textbook of Nocedal and Wright [132] for a comprehensive discussion on this topic.

In our implementation of the proposed method in PARDYNOPT, see chapter 7, and the numerical results of chapter 8 and chapter 9 we use IPOPT [169] by Wächter and Biegler as an implementation of an interior-point method. There, we choose an L-BFGS update formula for the Hessian approximation to overcome the drawbacks of an exact Hessian.

### 4.3.5 Approximation of Hessian of Lagrangian in a Generalized Gauß-Newton Framework

One-level NLP (4.13) belongs to the special class of nonlinear least-squares problems. Full step Generalized Gauss-Newton methods converge locally to stable minima, see [25]. This advantage over SQP methods in general can be exploited by an appropriate approximation of the Hessian of the Lagrangian and realized within a SQP framework, see subsection 1.2.3 for more detail. With the definition of the Gauß-Newton objective in (4.14) and the Jacobian of the residual vector in (4.17) with the submatrices stated in (4.15) and (4.16) the Hessian approximation of the Lagrangian as $\nabla_{\boldsymbol{Z}} \boldsymbol{R}^T \nabla_{\boldsymbol{Z}} \boldsymbol{R}$ can be computed and used in standard SQP solvers, e.g. FILTERSQP [59] and SNOPT[71]. In our implementation of the proposed method in PARDYNOPT a Generalized Gauß-Newton type approximation of the Hessian of the Lagrangian is integrated and interfaces to the SQP solvers mentioned above are prepared and can easily be extended for future research.

## 4.4 Outlook: Sequential Algorithm with Identification of Active Set

In this section we give an outlook how the optimal active set $\mathcal{A}^*$ related to the lower level OCP in the structured one-level (4.13) could be identified within a sequential algorithm and discuss possible realizations.

### 4.4.1 Determination of Active Set

Active-set methods are broadly used in constrained optimization, e.g. in linear programming or quadratic programming. In these methods estimates of the active set are determined and updated in each step of the algorithms. Active-set methods play an important role in SQP methods, where variants of two different approaches are common - an *inequality constrained QP* approach and an *equality constrained QP* approach. In the first approach in each iteration of the SQP algorithm an inequality constrained QP is solved and with information from resulting multipliers a guess of the optimal active set is performed until a KKT point of the NLP of interest is found. Whereas in the latter approach, the identification of the active set and the calculation of iterates, where an equality constrained QP is solved on the estimated active set, are separated from each other. For a comprehensive discussion the reader is advised to the textbook of Nocedal and Wright [132, Chapter 18]. Inspired by these approaches a possible sequential algorithm is sketched in the following subsection.

### 4.4.2 A Sequential Algorithm for Active-Set Identification

In problem (4.13) from section 4.2 the working set $\mathcal{W}(\boldsymbol{z})$ is a suitable guess of the optimal active set $\mathcal{A}^*$ related to the lower level OCP with linearly independent constraint gradients. In the case, where the guess of the working set is the same as the optimal active set, bilevel NLPs (4.5) and (4.7) are equivalent and, hence, the solution of the structured one-level NLP (4.13) on this optimal active set is the final solution of the original Bilevel Inverse OCP (2.22) and nothing else has to be done. If this is not the case, the optimal active set $\mathcal{A}^*$ can be identified by a sequential algorithm as sketched in the following.

**Algorithm 3** Sequential Determination of Active Set

1: Choose feasible $\boldsymbol{z}^0$ and set initial working set $\mathcal{W}^0$
2: Obtain $\boldsymbol{Z}^0$ from solution of one-level NLP (4.13) on $\mathcal{W}^0$, $k \leftarrow 0$
3: **while** KKT conditions (2.25) of lower level in (4.5) are not satisfied at $\boldsymbol{Z}^k$ **do**
4: Determine new active set and obtain $\mathcal{W}^{k+1}$
5: Compute solution $\boldsymbol{Z}^{k+1}$ of one-level NLP (4.13) on new working set $\mathcal{W}^{k+1}$
6: $k \leftarrow k+1$
7: **end while**

We choose an initial $\boldsymbol{z}^0$ to be feasible, and start with a suitable first guess of the working set $\mathcal{W}^0 := \mathcal{W}(\boldsymbol{z}^0)$. Initial iterate $\boldsymbol{Z}^0$ with

$$\boldsymbol{Z}^k := \begin{pmatrix} (\boldsymbol{z}^k)^T & (\boldsymbol{\lambda}^k)^T & (\boldsymbol{\mu}_{\overline{z}}^k)^T & (\boldsymbol{\mu}_{\underline{z}}^k)^T & (\boldsymbol{\alpha}^k)^T & (\boldsymbol{p}^k)^T \end{pmatrix}^T,$$

for $k = 0$ is then obtained by solving the one-level NLP (4.13) on fixed $\mathcal{W}^0$. In each iteration $k$ of this algorithm an NLP of the form (4.7) is treated according to the steps from subsection 4.2.3 and subsection 4.2.4, where the resulting structured one-level NLP (4.13) is solved on a fixed guess $\mathcal{W}^k := \mathcal{W}(\boldsymbol{z}^k)$. If the solution $\boldsymbol{Z}^k$ at iteration $k$ meets the KKT conditions of the lower level of the original bilevel NLP (4.5), the guess $\mathcal{W}^k$ is supposed to be equivalent to the optimal active set $\mathcal{A}^*$ and the algorithm terminates. The KKT conditions of the lower level problem are stated in the following. With the assumption that LICQ holds and with the introduction of Lagrange multiplier vectors $\hat{\boldsymbol{\mu}}_{\overline{z}} \in \mathbb{R}^{n_z}$ and $\hat{\boldsymbol{\mu}}_{\underline{z}} \in \mathbb{R}^{n_z}$ related to upper and lower bounds, respectively, defined in (4.10) and combined in $\boldsymbol{\mu} := \begin{pmatrix} \hat{\boldsymbol{\mu}}_{\overline{z}}^T & \hat{\boldsymbol{\mu}}_{\underline{z}}^T \end{pmatrix}^T$, the KKT conditions of the lower level problem of bilevel NLP (4.5) can be formulated as (2.25) in subsection 2.4.3.

If at iterate $k$ in algorithm 3 the solution $\boldsymbol{Z}^k$ of the one-level NLP (4.13) does not meet KKT conditions (2.25), a new suitable guess of the active set has to be determined. This can be done for instance by adding the indices of the inequality constraints in (2.25c) and (2.25d) to the next working set $\mathcal{W}^{k+1}$ at iterate $k+1$ which are not satisfied, and, in addition, dropping the indices related to Lagrange multipliers $\boldsymbol{\mu} < 0$, which do not fulfill equation (2.25e). Furthermore, the complementarity constraints (2.25f) and (2.25f) have to be fulfilled. In the identification of the optimal active set $\mathcal{A}^*$ it is important to ensure that all constraint gradients related to working set $\mathcal{W}^k$ are linearly independent at iteration point $\boldsymbol{Z}^k$.

The whole procedure in the active-set determination might cause *combinatorial difficulty* of nonlinear programming, see [132, Chapter 15.2], hence, further investigation is mandatory for a sophisticated approach applied to a more general problem class. However, in our research we consider a special class of problems: a parameter estimation problem constrained by an OCP, where the human gait is modeled. In this particular case given measurement data can be incorporated for the choice of the working set. As already mentioned in section 4.1 the indices related to simple bounds on the variables $\boldsymbol{v}$ as well as bounds on the duration parameter $d$ can mostly be dropped completely. Furthermore, the upper bounds on the slack variables $\boldsymbol{w}$ are set to $\infty$ and can also be neglected. For the lower bounds on the slack variables the situation becomes different. The indices related to this bounds incorporate information about active mixed control-state constraints and active boundary inequality constraints of the original problem at the solution. However, in many cases due to mea-

surements one can reduce the index set of all possible working sets and, hence, can already start with a suitable guess of the optimal active set. This facilitates the identification of the active set and reduces the combinatorial difficulty significantly.

# Chapter 5

# Bilevel Inverse OCP for Identification of Unknowns in a Basic Walker Gait Model

In this chapter we give a Bilevel Inverse OCP formulation to identify unknown quantities in a basic model for human locomotion meeting given measurements. We derive a suitable multi-stage OCP for a two dimensional rigid multibody system - a basic walker as the one first stated in [66]. Its dynamics with external contacts, which enter our gait model, was already introduced and derived in the work of Schlöder [151]. A multi-stage OCP formulation is also given there. In our work we follow the detailed description of [151] and adapt the OCP to a more suitable formulation for our purposes with extended bounds and a terminal condition. This can be done because the OCP serves as the lower level in our Bilevel Inverse OCP formulation.

This chapter is structured as follows. In the first section we give the underlying phasewise dynamics of a basic walker, which are described by equations of motion in implicit form as in the general formulation (3.6) in subsection 3.1.1. Furthermore, we derive the equations of motion in explicit form. The second section is concerned with the derivation of a suitable gait model for the basic walker as the solution of an OCP. Finally, in the last section a Bilevel Inverse OCP is stated as a PE Problem constrained by the previously defined multi-stage OCP. Therein, the gait model is fitted to given measurement data for the identification of unknown quantities in a basic walker model.

## 5.1 Dynamics of a Basic Walker Model

The basic walker model consists of two rigid and massless sticks of length $\ell$ which are connected through a revolute joint located at the base segment with point mass $M$. At the end of both sticks feet segments are added with point masses $m = m_l = m_r$ as illustrated in Figure 5.1. In the following the length of the sticks, as well as the point masses of the head and the feet are combined in the model parameter vector $\boldsymbol{p} = \begin{pmatrix} \ell & M & m \end{pmatrix}^T$.

We consider a full gait cycle with two steps where the initial and final posture coincide and start with a single support phase 1 with the right foot fixed to the ground. After an instantaneous transition where a perfectly inelastic collision of the left foot takes place, a single support phase 2 with left foot contact follows. The gait cycle ends when the right foot touches the ground again and a second transition with a perfectly inelastic collision is performed. In total, two dynamic phases have to be considered and the transitions between. To describe the full periodic gait cycle of the basic walker multibody system we start with the generalized coordinates. They include the horizontal and vertical position of the base segment or head, $x_h(t)$ and $y_h(t)$, respectively. Furthermore, they incorporate the angles, $\varphi_l(t)$ and $\varphi_r(t)$, which describe the rotation of the left and right leg around the base segment, respectively. In summary, we have the following generalized coordinate vectors $\boldsymbol{q}$ with four entries and its corresponding generalized velocities denoted by $\dot{\boldsymbol{q}}$ and accelerations by $\ddot{\boldsymbol{q}}$

$$\boldsymbol{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} x_h \\ y_h \\ \varphi_l \\ \varphi_r \end{pmatrix},$$

where we omit the time dependency which is also done in the following sections for a more compact representation. Furthermore, the basic walker model is actuated through torques acting at the revolute joint located at

**Table 5.1:** Definition of quantities which appear to describe the equations of motion (3.15) for the basic walker example with the definitions given in this chapter.

| Symbol | Description |
|---|---|
| $x_h$ | horizontal position of base segment (head) |
| $y_h$ | vertical position of base segment (head) |
| $\varphi_l$ | rotation of left leg around head |
| $\varphi_r$ | rotation of right leg around head |
| $\dot{\varphi}_l$ | angular velocity of left leg |
| $\dot{\varphi}_r$ | angular velocity of right leg |
| $\tau_l$ | left torque around head |
| $\tau_r$ | right torque around head |
| $x_l$ | horizontal position of left foot |
| $y_l$ | vertical position of left foot |
| $x_k$ | horizontal position of right foot |
| $y_k$ | vertical position of right foot |
| $\ell$ | length of legs |
| $M$ | mass of base segment |
| $m$ | mass of foot segment |
| $g$ | gravitational acceleration |

the base segment. In the equations of motion the following torque vector is considered:

$$\boldsymbol{\tau} = \begin{pmatrix} 0 \\ 0 \\ \tau_0^{\text{act}} \\ \tau_1^{\text{act}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \tau_l \\ \tau_r \end{pmatrix}.$$

In Table 5.1 the most frequently used quantities are summarized and in Figure 5.1 illustrated to describe the equations of motion of the rigid multibody system for the basic walker.

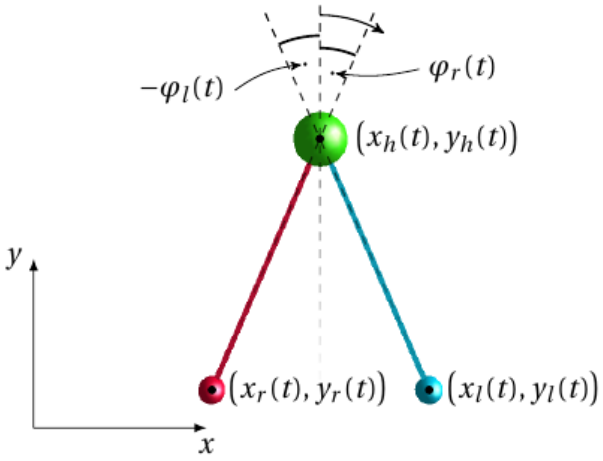


**Figure 5.1:** Basic walker model described by a rigid multibody system model with four DOFs. With red segments the figure depicts the right leg and with blue segments the left leg. Illustration taken from [151].

### 5.1.1 Equations of Motion for Single Support Phase Right

In the first phase we fix the right foot to the ground at $\begin{pmatrix} x_0 & 0 \end{pmatrix}^T$ and let the basic walker multibody system perform one step with the left foot. This single support phase of the right foot is denoted by phase 1 in the

following. To set up the underlying dynamics we start with the Cartesian coordinates of the right foot denoted by

$$x_r = x_h - \ell \sin\varphi_r,$$
$$y_r = y_h - \ell \cos\varphi_r,$$

and define the following constraint set

$$\boldsymbol{g}_1(\boldsymbol{q},\boldsymbol{p}) = \begin{pmatrix} x_h - \ell\sin\varphi_r \\ y_h - \ell\cos\varphi_r \end{pmatrix} - \begin{pmatrix} x_0 \\ 0 \end{pmatrix} = 0.$$

Together with the contact Jacobian described by

$$\boldsymbol{G}_1(\boldsymbol{q},\boldsymbol{p}) = \frac{\partial \boldsymbol{g}_1(\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{q}} = \begin{pmatrix} 1 & 0 & 0 & -\ell\cos\varphi_r \\ 0 & 1 & 0 & \ell\sin\varphi_r \end{pmatrix},$$

and the contact Hessian

$$\boldsymbol{\gamma}_1(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = -\dot{\boldsymbol{G}}_1(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}} = \begin{pmatrix} -\ell\dot{\varphi}_r^2\sin\varphi_r \\ -\ell\dot{\varphi}_r^2\cos\varphi_r \end{pmatrix},$$

the equations of motion can be set up under consideration of Lagrangian mechanics as done in [151]. We then obtain a symmetric and positive definite joint space inertia matrix

$$\boldsymbol{H}(\boldsymbol{q},\boldsymbol{p}) = \begin{pmatrix} M+2m & 0 & -m\ell\cos\varphi_l & -m\ell\cos\varphi_r \\ 0 & M+2m & m\ell\sin\varphi_l & m\ell\sin\varphi_r \\ -m\ell\cos\varphi_l & m\ell\sin\varphi_l & m\ell^2 & 0 \\ -m\ell\cos\varphi_r & m\ell\sin\varphi_r & 0 & m\ell^2 \end{pmatrix},$$

and the generalized bias force defined by

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = \begin{pmatrix} m\ell\dot{\varphi}_l^2\sin\varphi_l + m\ell\dot{\varphi}_r^2\sin\varphi_r \\ m\ell\dot{\varphi}_l^2\cos\varphi_l + m\ell\dot{\varphi}_r^2\cos\varphi_r + (M+2m)g \\ mg\ell\sin\varphi_l \\ mg\ell\sin\varphi_r \end{pmatrix},$$

with gravitational acceleration $g$. Together with all these quantities the equations of motion are given by equation system (3.15) in subsection 3.1.2.

### 5.1.2 Equations of Motion for Single Support Phase Left

In a second phase the left foot is fixed to the ground and the right foot swings above ground to perform a second step. The Cartesian coordinates of the left foot are defined by

$$x_l = x_h - \ell \sin\varphi_l,$$
$$y_l = y_h - \ell \cos\varphi_l.$$

Similar to the previous phase 1, the single support phase of the left foot is denoted by phase 2 in the following. For the dynamics of phase 2 the joint space inertia and generalized bias force are the same as above. The contact Jacobian reads

$$\boldsymbol{G}_2(\boldsymbol{q},\boldsymbol{p}) = \frac{\partial \boldsymbol{g}_2(\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{q}} = \begin{pmatrix} 1 & 0 & -\ell\cos\varphi_l & 0 \\ 0 & 1 & \ell\sin\varphi_l & 0 \end{pmatrix},$$

and contact Hessian is denoted by

$$\boldsymbol{\gamma}_2(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{p}) = -\dot{\boldsymbol{G}}_2(\boldsymbol{q},\boldsymbol{p})\dot{\boldsymbol{q}} = \begin{pmatrix} -\ell\dot{\varphi}_l^2\sin\varphi_l \\ -\ell\dot{\varphi}_l^2\cos\varphi_l \end{pmatrix}.$$

Analogous to the first phase the equations of motion for the second phase are given by equation (3.15) in subsection 3.1.2 with the previously defined quantities.

### 5.1.3 Collision Impacts after each Phase

With all quantities defined in the previous subsections we can formulate the collision after the first phase where the left foot touches the ground, as well as the second collision of the right foot after the second phase. The change in the generalized velocities can be computed as part of the equation system (3.16). For a more detailed derivation of the equations of motion in implicit form as summarized in this subsection we refer to appendix A.1 in [151].

### 5.1.4 Explicit Formulation of the Equations of Motion

In this subsection we derive the equations of motion in explicit form to use them in a Bilevel Inverse OCP framework in section 9.1 treated with our DISIMFAS, which was developed and implemented as part of this thesis in the software package PARDYNOPT. Because SOLVIND [5] with its interface to ADOL-C [171] is chosen for sensitivity and derivative generation using AD in the sense of IND, an explicit formula for the equations of motion is mandatory. We compute the inverses of the matrices

$$\boldsymbol{M}_\mathrm{j} := \begin{pmatrix} \boldsymbol{H}(\boldsymbol{q},\boldsymbol{p}) & \boldsymbol{G}_\mathrm{j}(\boldsymbol{q},\boldsymbol{p})^T \\ \boldsymbol{G}_\mathrm{j}(\boldsymbol{q},\boldsymbol{p}) & \boldsymbol{0} \end{pmatrix},$$

which arise in the equations of motion for each phase $\mathrm{j} = 1,2$, as well as in the linear equation system to describe the collision of the left or right foot at touch-down. The matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are symmetric and, hence, so are their inverses $\boldsymbol{M}_1^{-1}$ and $\boldsymbol{M}_2^{-1}$, respectively. In the following we present our results achieved for both inverses, which were independently confirmed by [151]. With the greatest common divisor

$$\mathrm{gcd} = M + m\sin^2(\varphi_l - \varphi_r),$$

and $\boldsymbol{M}_\mathrm{j}(k,l)$, where $k = 0,\ldots,5$ denotes the row entry and $l = 0,\ldots,5$ the column, the first row of $\boldsymbol{M}_1^{-1}$ reads

$$\begin{aligned}
\boldsymbol{M}_1^{-1}(0,0) &= \frac{1}{\mathrm{gcd}}\cos^2\varphi_r, \\
\boldsymbol{M}_1^{-1}(0,1) &= -\frac{1}{2\,\mathrm{gcd}}\sin(2\varphi_r), \\
\boldsymbol{M}_1^{-1}(0,2) &= \frac{1}{\ell\,\mathrm{gcd}}\cos\varphi_r\cos(\varphi_l - \varphi_r), \\
\boldsymbol{M}_1^{-1}(0,3) &= \frac{1}{\ell\,\mathrm{gcd}}\cos\varphi_r, \\
\boldsymbol{M}_1^{-1}(0,4) &= \frac{1}{\mathrm{gcd}}\left((m+M)\sin^2\varphi_r - m\sin\varphi_l\sin\varphi_r\cos(\varphi_l - \varphi_r)\right), \\
\boldsymbol{M}_1^{-1}(0,5) &= \frac{1}{\mathrm{gcd}}\cos\varphi_r\left(M\sin\varphi_r + m\cos\varphi_l\sin(\varphi_r - \varphi_l)\right).
\end{aligned}$$

The second row is:

$$
\begin{aligned}
\boldsymbol{M}_1^{-1}(1,0) &= \boldsymbol{M}_1^{-1}(0,1),\\
\boldsymbol{M}_1^{-1}(1,1) &= \frac{1}{\text{gcd}}\sin^2\varphi_r,\\
\boldsymbol{M}_1^{-1}(1,2) &= -\frac{1}{\ell\,\text{gcd}}\sin\varphi_r\cos(\varphi_l-\varphi_r),\\
\boldsymbol{M}_1^{-1}(1,3) &= -\frac{1}{\ell\,\text{gcd}}\sin\varphi_r,\\
\boldsymbol{M}_1^{-1}(1,4) &= \frac{1}{\text{gcd}}\left((\cos\varphi_r\sin\varphi_r(M+m\sin^2\varphi_l)-m\cos\varphi_l\sin\varphi_l\sin^2\varphi_r)\right),\\
\boldsymbol{M}_1^{-1}(1,5) &= \frac{1}{\text{gcd}}\left((m+M)\cos^2\varphi_r-m\cos\varphi_l\cos\varphi_r\cos(\varphi_l-\varphi_r)\right).
\end{aligned}
$$

The third row is:

$$
\begin{aligned}
\boldsymbol{M}_1^{-1}(2,0) &= \boldsymbol{M}_1^{-1}(0,2),\\
\boldsymbol{M}_1^{-1}(2,1) &= \boldsymbol{M}_1^{-1}(1,2),\\
\boldsymbol{M}_1^{-1}(2,2) &= \frac{1}{m\ell^2\,\text{gcd}}(M+m),\\
\boldsymbol{M}_1^{-1}(2,3) &= \frac{1}{\ell^2\,\text{gcd}}\cos(\varphi_l-\varphi_r),\\
\boldsymbol{M}_1^{-1}(2,4) &= \frac{1}{\ell\,\text{gcd}}(m+M)\sin\varphi_r\sin(\varphi_r-\varphi_l),\\
\boldsymbol{M}_1^{-1}(2,5) &= \frac{1}{\ell\,\text{gcd}}(m+M)\cos\varphi_r\sin(\varphi_r-\varphi_l).
\end{aligned}
$$

The fourth row is:

$$
\begin{aligned}
\boldsymbol{M}_1^{-1}(3,0) &= \boldsymbol{M}_1^{-1}(0,3),\\
\boldsymbol{M}_1^{-1}(3,1) &= \boldsymbol{M}_1^{-1}(1,3),\\
\boldsymbol{M}_1^{-1}(3,2) &= \boldsymbol{M}_1^{-1}(2,3),\\
\boldsymbol{M}_1^{-1}(3,3) &= \frac{1}{\ell^2\,\text{gcd}},\\
\boldsymbol{M}_1^{-1}(3,4) &= \frac{1}{\ell\,\text{gcd}}\left(m\sin\varphi_l\sin(\varphi_r-\varphi_l)-M\cos\varphi_r\right),\\
\boldsymbol{M}_1^{-1}(3,5) &= \frac{1}{\ell\,\text{gcd}}\left(m\cos\varphi_l\sin(\varphi_r-\varphi_l)+M\sin\varphi_r\right).
\end{aligned}
$$

The fifth row is:

$$
\begin{aligned}
\boldsymbol{M}_1^{-1}(4,0) &= \boldsymbol{M}_1^{-1}(0,4),\\
\boldsymbol{M}_1^{-1}(4,1) &= \boldsymbol{M}_1^{-1}(1,4),\\
\boldsymbol{M}_1^{-1}(4,2) &= \boldsymbol{M}_1^{-1}(2,4),\\
\boldsymbol{M}_1^{-1}(4,3) &= \boldsymbol{M}_1^{-1}(3,4),\\
\boldsymbol{M}_1^{-1}(4,4) &= -\frac{1}{\text{gcd}}\left(M^2\sin^2\varphi_r+Mm(\sin^2\varphi_r+1)+m^2\sin^2(\varphi_r-\varphi_l)\right),\\
\boldsymbol{M}_1^{-1}(4,5) &= -\frac{1}{2\,\text{gcd}}M(m+M)\sin(2\varphi_r),
\end{aligned}
$$

and the sixth row is:

$$
\begin{aligned}
\boldsymbol{M}_1^{-1}(5,0) &= \boldsymbol{M}_1^{-1}(0,5),\\
\boldsymbol{M}_1^{-1}(5,1) &= \boldsymbol{M}_1^{-1}(1,5),\\
\boldsymbol{M}_1^{-1}(5,2) &= \boldsymbol{M}_1^{-1}(2,5),\\
\boldsymbol{M}_1^{-1}(5,3) &= \boldsymbol{M}_1^{-1}(3,5),\\
\boldsymbol{M}_1^{-1}(5,4) &= \boldsymbol{M}_1^{-1}(4,5),\\
\boldsymbol{M}_1^{-1}(5,5) &= -\frac{1}{\mathrm{gcd}}\left(M^2\cos^2\varphi_r + Mm(\cos^2\varphi_r + 1) + m^2\sin^2(\varphi_r - \varphi_l)\right).
\end{aligned}
$$

The results for the inverse $\boldsymbol{M}_2^{-1}$ for phase 2 are summarized in the following. The first row reads:

$$
\begin{aligned}
\boldsymbol{M}_2^{-1}(0,0) &= \frac{1}{\mathrm{gcd}}\cos^2\varphi_l,\\
\boldsymbol{M}_2^{-1}(0,1) &= -\frac{1}{2\,\mathrm{gcd}}\sin(2\varphi_l),\\
\boldsymbol{M}_2^{-1}(0,2) &= \frac{1}{\ell\,\mathrm{gcd}}\cos\varphi_l,\\
\boldsymbol{M}_2^{-1}(0,3) &= \frac{1}{\ell\,\mathrm{gcd}}\cos\varphi_l\cos(\varphi_l - \varphi_r),\\
\boldsymbol{M}_2^{-1}(0,4) &= \frac{1}{\mathrm{gcd}}\left((m+M)\sin^2\varphi_l - m\sin\varphi_l\sin\varphi_r\cos(\varphi_l - \varphi_r)\right),\\
\boldsymbol{M}_2^{-1}(0,5) &= \frac{1}{\mathrm{gcd}}\cos\varphi_l\left(M\sin\varphi_l + m\cos\varphi_r\sin(\varphi_l - \varphi_r)\right).
\end{aligned}
$$

The second row reads:

$$
\begin{aligned}
\boldsymbol{M}_2^{-1}(1,0) &= \boldsymbol{M}_2^{-1}(0,1),\\
\boldsymbol{M}_2^{-1}(1,1) &= \frac{1}{\mathrm{gcd}}\sin^2\varphi_l,\\
\boldsymbol{M}_2^{-1}(1,2) &= -\frac{1}{\ell\,\mathrm{gcd}}\sin\varphi_l,\\
\boldsymbol{M}_2^{-1}(1,3) &= -\frac{1}{\ell\,\mathrm{gcd}}\sin\varphi_l\cos(\varphi_l - \varphi_r),\\
\boldsymbol{M}_2^{-1}(1,4) &= \frac{1}{\mathrm{gcd}}\left((\cos\varphi_l\sin\varphi_l(M + m\sin^2\varphi_r) - m\cos\varphi_r\sin\varphi_r\sin^2\varphi_l)\right),\\
\boldsymbol{M}_2^{-1}(1,5) &= \frac{1}{\mathrm{gcd}}\left((m+M)\cos^2\varphi_l - m\cos\varphi_l\cos\varphi_r\cos(\varphi_l - \varphi_r)\right).
\end{aligned}
$$

The third row reads:

$$
\begin{aligned}
\boldsymbol{M}_2^{-1}(2,0) &= \boldsymbol{M}_2^{-1}(0,2),\\
\boldsymbol{M}_2^{-1}(2,1) &= \boldsymbol{M}_2^{-1}(1,2),\\
\boldsymbol{M}_2^{-1}(2,2) &= \frac{1}{\ell^2\,\mathrm{gcd}},\\
\boldsymbol{M}_2^{-1}(2,3) &= \frac{1}{\ell^2\,\mathrm{gcd}}\cos(\varphi_l - \varphi_r),\\
\boldsymbol{M}_2^{-1}(2,4) &= \frac{1}{\ell\,\mathrm{gcd}}\left(m\sin\varphi_r\sin(\varphi_l - \varphi_r) - M\cos\varphi_l\right),\\
\boldsymbol{M}_2^{-1}(2,5) &= \frac{1}{\ell\,\mathrm{gcd}}\left(m\cos\varphi_r\sin(\varphi_l - \varphi_r) + M\sin\varphi_l\right).
\end{aligned}
$$

The fourth row reads:

$$
\begin{aligned}
\boldsymbol{M}_2^{-1}(3,0) &= \boldsymbol{M}_2^{-1}(0,3),\\
\boldsymbol{M}_2^{-1}(3,1) &= \boldsymbol{M}_2^{-1}(1,3),\\
\boldsymbol{M}_2^{-1}(3,2) &= \boldsymbol{M}_2^{-1}(2,3),\\
\boldsymbol{M}_2^{-1}(3,3) &= \frac{1}{m\ell^2\,\mathrm{gcd}}(M+m),\\
\boldsymbol{M}_2^{-1}(3,4) &= \frac{1}{\ell\,\mathrm{gcd}}(m+M)\sin\varphi_l\sin(\varphi_l-\varphi_r),\\
\boldsymbol{M}_2^{-1}(3,5) &= \frac{1}{\ell\,\mathrm{gcd}}(m+M)\cos\varphi_l\sin(\varphi_l-\varphi_r).
\end{aligned}
$$

The fifth row reads:

$$
\begin{aligned}
\boldsymbol{M}_2^{-1}(4,0) &= \boldsymbol{M}_2^{-1}(0,4),\\
\boldsymbol{M}_2^{-1}(4,1) &= \boldsymbol{M}_2^{-1}(1,4),\\
\boldsymbol{M}_2^{-1}(4,2) &= \boldsymbol{M}_2^{-1}(2,4),\\
\boldsymbol{M}_2^{-1}(4,3) &= \boldsymbol{M}_2^{-1}(3,4),\\
\boldsymbol{M}_1^{-1}(4,4) &= -\frac{1}{\mathrm{gcd}}\left(M^2\sin^2\varphi_l + Mm(\sin^2\varphi_l+1) + m^2\sin^2(\varphi_r-\varphi_l)\right),\\
\boldsymbol{M}_2^{-1}(4,5) &= -\frac{1}{2\,\mathrm{gcd}}M(m+M)\sin(2\varphi_l).
\end{aligned}
$$

The sixth row reads:

$$
\begin{aligned}
\boldsymbol{M}_2^{-1}(5,0) &= \boldsymbol{M}_2^{-1}(0,5),\\
\boldsymbol{M}_2^{-1}(5,1) &= \boldsymbol{M}_2^{-1}(1,5),\\
\boldsymbol{M}_2^{-1}(5,2) &= \boldsymbol{M}_2^{-1}(2,5),\\
\boldsymbol{M}_2^{-1}(5,3) &= \boldsymbol{M}_2^{-1}(3,5),\\
\boldsymbol{M}_2^{-1}(5,4) &= \boldsymbol{M}_2^{-1}(4,5),\\
\boldsymbol{M}_1^{-1}(5,5) &= -\frac{1}{\mathrm{gcd}}\left(M^2\cos^2\varphi_l + Mm(\cos^2\varphi_l+1) + m^2\sin^2(\varphi_r-\varphi_l)\right).
\end{aligned}
$$

## 5.2 A Multi-Stage OCP for the Gait of a Basic Walker

To set up an OCP for the gait of the basic walker multibody system from section 5.1 with its dynamics, we consider the general multi-stage OCP formulation (3.8) from subsection 3.1.2 describing the human gait. The problem formulation chosen here serves as the lower level in the Bilevel Inverse OCP later introduced in section 5.3. Therein, a PE Problem is stated constrained by a two-stage OCP describing two steps.

We start with the definition of optimization variables as in subsection 3.1.2. The differential states are phase-wise defined as

$$
\boldsymbol{x}_{\mathrm{j}} = \begin{pmatrix}\boldsymbol{q}_{\mathrm{j}}\\ \dot{\boldsymbol{q}}_{\mathrm{j}}\end{pmatrix} = \begin{pmatrix} q_{\mathrm{j}0}\\ q_{\mathrm{j}1}\\ q_{\mathrm{j}2}\\ q_{\mathrm{j}3}\\ \dot{q}_{\mathrm{j}0}\\ \dot{q}_{\mathrm{j}1}\\ \dot{q}_{\mathrm{j}2}\\ \dot{q}_{\mathrm{j}3}\end{pmatrix} = \begin{pmatrix} x_h^{\mathrm{j}}\\ y_h^{\mathrm{j}}\\ \varphi_l^{\mathrm{j}}\\ \varphi_r^{\mathrm{j}}\\ \dot{x}_h^{\mathrm{j}}\\ \dot{y}_h^{\mathrm{j}}\\ \dot{\varphi}_l^{\mathrm{j}}\\ \dot{\varphi}_r^{\mathrm{j}}\end{pmatrix},
$$

with model stage index j = 1,2. The controls are set to the actuated generalized forces and enter the equations of motion with

$$\boldsymbol{\tau}_\mathrm{j} = \begin{pmatrix} 0 \\ 0 \\ u_{\mathrm{j}0} \\ u_{\mathrm{j}1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{u}_\mathrm{j} = \begin{pmatrix} u_{\mathrm{j}0} \\ u_{\mathrm{j}1} \end{pmatrix} = \begin{pmatrix} \tau_{\mathrm{j}0}^{\mathrm{act}} \\ \tau_{\mathrm{j}1}^{\mathrm{act}} \end{pmatrix} = \begin{pmatrix} \tau_l^\mathrm{j} \\ \tau_r^\mathrm{j} \end{pmatrix}.$$

With all later in this section introduced constraints in subsection 5.2.3 together with objective function (5.17) in subsection 5.2.1, dynamics (5.18), and transition conditions (5.19) in subsection 5.2.2, the multi-stage OCP is set up completely to describe two steps of a basic walker multibody system and reads as follows:

$$\min_{\substack{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{u}_1,\boldsymbol{u}_2,\\ d_1,d_2}} \quad \alpha_1(d_1+d_2) + \alpha_2 \sum_{\mathrm{j}=1}^{2} \int_{t_\mathrm{s}^\mathrm{j}}^{t_\mathrm{f}^\mathrm{j}} \left( {u_{\mathrm{j}0}}^2(t) + {u_{\mathrm{j}1}}^2(t) \right) \mathrm{d}t \tag{5.16a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_\mathrm{j}(t) = d_\mathrm{j} \cdot \boldsymbol{f}_\mathrm{j}(\boldsymbol{x}_\mathrm{j}(t), \boldsymbol{u}_\mathrm{j}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}^\mathrm{j},\ \mathrm{j}=1,2, \tag{5.16b}$$

$$\boldsymbol{x}_2(t_\mathrm{s}^2) = \Delta_1(\boldsymbol{x}_1(t_\mathrm{f}^1), \boldsymbol{p}), \tag{5.16c}$$

$$\boldsymbol{x}_1(t_\mathrm{s}^1) = \Delta_2(\boldsymbol{x}_2(t_\mathrm{f}^2), \boldsymbol{p}), \tag{5.16d}$$

$$0 \le \boldsymbol{c}_\mathrm{j}(\boldsymbol{x}_\mathrm{j}(t), \boldsymbol{p}), \qquad t \in \mathcal{T}^\mathrm{j},\ \mathrm{j}=1,2, \tag{5.16e}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{x}_1(t_\mathrm{s}^1), \boldsymbol{x}_1(t_\mathrm{f}^1), \boldsymbol{x}_2(t_\mathrm{f}^2), \boldsymbol{p}), \tag{5.16f}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{x}_1(t_\mathrm{f}^1), \boldsymbol{x}_2(t_\mathrm{f}^2), \boldsymbol{p}), \tag{5.16g}$$

$$\underline{\boldsymbol{x}_\mathrm{j}} \le \boldsymbol{x}_\mathrm{j}(t) \le \overline{\boldsymbol{x}_\mathrm{j}}, \qquad t \in \mathcal{T}^\mathrm{j},\ \mathrm{j}=1,2, \tag{5.16h}$$

$$\underline{\boldsymbol{u}_\mathrm{j}} \le \boldsymbol{u}_\mathrm{j}(t) \le \overline{\boldsymbol{u}_\mathrm{j}}, \qquad t \in \mathcal{T}^\mathrm{j},\ \mathrm{j}=1,2, \tag{5.16i}$$

$$0 \le d_\mathrm{j}, \qquad \mathrm{j}=1,2. \tag{5.16j}$$

It is defined on fixed and *normalized* time horizons $\mathcal{T}^\mathrm{j} := [t_\mathrm{s}^\mathrm{j}, t_\mathrm{f}^\mathrm{j}] = [0,1]$ with stage duration parameters $d_\mathrm{j}$ for each model stage j = 1,2 and stage transition times before and after collision, $t_\mathrm{f}^1 = t_\mathrm{f}^2 = 1$ and $t_\mathrm{s}^2 = t_\mathrm{s}^1 = 0$, respectively, as introduced in subsection 2.1.2 together with the remaining variables and functions. The arising quantities $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \alpha_2 \end{pmatrix}^T$ and $\boldsymbol{p} = \begin{pmatrix} \ell & M & m \end{pmatrix}^T$ are variables in Bilevel Inverse OCP (5.28). In the following we give a more detailed explanation of all the quantities that occur in OCP (5.16).

### 5.2.1 Objective Function

The objective function we define is of Bolza-type and consists of a weighted sum of a Mayer-type objective that summarizes the duration parameters, and a Lagrange-type objective that sums over the integrated controls squared on each model stage j = 1,2

$$\sum_{\mathrm{j}=1}^{2} \boldsymbol{\Phi}_\mathrm{j}(\boldsymbol{x}_\mathrm{j}(t), \boldsymbol{u}_\mathrm{j}(t), \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{p}) = \alpha_1(d_1+d_2) + \alpha_2 \sum_{\mathrm{j}=1}^{2} \int_{t_\mathrm{s}^\mathrm{j}}^{t_\mathrm{f}^\mathrm{j}} \left( {u_{\mathrm{j}0}}^2(t) + {u_{\mathrm{j}1}}^2(t) \right) \mathrm{d}t, \tag{5.17}$$

with weights $\alpha_1$ and $\alpha_2$. The first term minimizes the total duration of the whole process, whereas the second term minimizes the energy required to make two steps.

### 5.2.2 Dynamics and its Transitions

The dynamics on each model stage j = 1,2 enter the multi-stage OCP formulation in (5.16b) and include the following ODE system

$$\begin{aligned}\dot{\boldsymbol{x}}_{\mathrm{j}}(t) &= d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), \boldsymbol{p}) \\ &= d_{\mathrm{j}} \cdot \begin{pmatrix} \dot{\boldsymbol{q}}_{\mathrm{j}}(t) \\ \ddot{\boldsymbol{q}}_{\mathrm{j}}(t) \end{pmatrix}, \qquad t \in \mathcal{T}^{\mathrm{j}} = [t_{\mathrm{s}}^{\mathrm{j}}, t_{\mathrm{f}}^{\mathrm{j}}],\end{aligned} \tag{5.18}$$

where the generalized acceleration $\ddot{\boldsymbol{q}}_{\mathrm{j}}$ is part of the solution of the equations of motion (3.15) with the quantities defined in section 5.1. Their transitions are denoted by

$$\boldsymbol{x}_2(t_{\mathrm{s}}^2) = \Delta_1(\boldsymbol{x}_1(t_{\mathrm{f}}^1), \boldsymbol{p}) \quad \text{and} \quad \boldsymbol{x}_1(t_{\mathrm{s}}^1) = \Delta_2(\boldsymbol{x}_2(t_{\mathrm{f}}^2), \boldsymbol{p}), \tag{5.19}$$

where the first condition enters the OCP in (5.16c) and couples the differential states $\boldsymbol{x}_1$ of model stage 1 at final time $t_{\mathrm{f}}^1$ with the differential states $\boldsymbol{x}_2$ of model stage 2 at initial time $t_{\mathrm{s}}^2$. The second condition in (5.16d) maps the differential states $\boldsymbol{x}_2$ of model stage 2 at final time $t_{\mathrm{f}}^2$ to the differential states $\boldsymbol{x}_1$ of model stage 1 at initial time $t_{\mathrm{s}}^1$. This is a small difference to the general problem formulation of (3.8) but simplifies the formulation of periodicity constraints. For the first transition $\Delta_1(\cdot)$ the generalized coordinates $q_1(t_{\mathrm{f}}^1) = q_2(t_{\mathrm{s}}^2)$ stay the same at collision. Whereas, in the second transition condition $\Delta_2(\cdot)$, terminal condition (5.24) and periodicity condition (5.25) are included, see subsection 5.2.3. The instantaneous change in the generalized velocities $\dot{\boldsymbol{q}}_{\mathrm{j}}$ can be calculated using the general formulation (3.16) with the given quantities of the previous section.

### 5.2.3 Constraints

In this subsection we formulate path constraints (5.16e) and multi-point boundary constraints (5.16f) and (5.16g) arising in the multi-stage OCP describing the gait of a basic walker multibody system. The Cartesian coordinates as defined in section 5.1 and its velocities for left and right foot, $k \in \{l, r\}$, are denoted by

$$\begin{aligned} x_k^{\mathrm{j}} &= x_h^{\mathrm{j}} - \ell \sin\varphi_k^{\mathrm{j}}, \\ y_k^{\mathrm{j}} &= y_h^{\mathrm{j}} - \ell \cos\varphi_k^{\mathrm{j}}, \\ \dot{x}_k^{\mathrm{j}} &= \dot{x}_h^{\mathrm{j}} - \ell \dot{\varphi}_k^{\mathrm{j}} \cos\varphi_k^{\mathrm{j}}, \\ \dot{y}_k^{\mathrm{j}} &= \dot{y}_h^{\mathrm{j}} + \ell \dot{\varphi}_k^{\mathrm{j}} \sin\varphi_k^{\mathrm{j}}, \end{aligned}$$

with model stage index $\mathrm{j} = 1, 2$.

#### Initial Conditions

At initial time, $t_{\mathrm{s}}^1 = 0, t_{\mathrm{s}}^1 \in \mathcal{T}^1$, the horizontal position of the base segment is fixed to 0 and the vertical position of both feet to the ground. The angles $\varphi_l^1(t_{\mathrm{s}}^1)$ and $\varphi_r^1(t_{\mathrm{s}}^1)$ are left free for optimization of the initial posture of the basic walker model. Furthermore, we ensure that the velocities in the right foot are set to 0. It is sufficient to enforce the initial conditions

$$\begin{aligned} x_h^1(t_{\mathrm{s}}^1) &= 0, \\ y_l^1(t_{\mathrm{s}}^1) &= 0, \\ y_r^1(t_{\mathrm{s}}^1) &= 0, \\ \dot{x}_r^1(t_{\mathrm{s}}^1) &= 0, \\ \dot{y}_r^1(t_{\mathrm{s}}^1) &= 0, \end{aligned}$$

to hold at the beginning of the first model stage incorporated in (5.16f) because of the transition conditions (5.16c) and (5.16d).

#### Constraints at Phase 1: Single Support Right Foot

During the first single support phase of the right foot in model stage 1 we have to ensure that the left foot does penetrate the ground only within a given threshold

$$y_l^1(t) \geq -\epsilon, t \in \mathcal{T}^1,$$

where $\epsilon > 0$ is a small number and for instance chosen to be $0.1\ell$ in section 9.1. As the legs in the basic walker multibody system are modeled as rigid sticks without knees, this relaxation has to be included and is chosen in such a way that this constraint is not violated in the solution of the Bilevel Inverse OCP in the following section.

#### Constraints at First Transition

At first transition time, $t_\mathrm{f}^1 = 1, t_\mathrm{f}^1 \in \mathcal{T}^1$, right before the collision of the left foot takes place, the following conditions are posed:

$$y_l^1(t_\mathrm{f}^1) = 0,$$
$$\dot{y}_l^1(t_\mathrm{f}^1) \leq 0,$$

where the left foot touches the ground with a negative velocity, which ensures that the foot approaches the ground from above.

#### Constraints at Phase 2: Single Support Left Foot

Similar to the first phase during the second phase the right foot should swing above ground allowing a small penetration $\epsilon > 0$, and the following constraint should hold:

$$y_r^2(t) \geq -\epsilon, t \in \mathcal{T}^2.$$

#### Constraints at Second Transition

At second transition time, right before collision, $t_\mathrm{f}^2 = 1, t_\mathrm{f}^2 \in \mathcal{T}^2$, the following conditions are posed:

$$y_r^2(t_\mathrm{f}^2) = 0,$$
$$\dot{y}_r^2(t_\mathrm{f}^2) \leq 0,$$

which are similar to the ones at first transition time.

#### Terminal Condition, Periodicity Constraints and Bounds on Differential States and Controls

The multi-stage OCP formulation described in this section serves as the lower level in the Bilevel Inverse OCP of section 5.3 where the upper level is a PE fitting the gait model to measurement data. Because of the given data a fixed terminal condition can be set to some measured value

$$x_h^2(t_\mathrm{f}^2) = \eta_\mathrm{f}, \tag{5.24}$$

for instance $\eta_\mathrm{f} = 0.9\ell$ as chosen in section 9.1, where simulated measurements are used in the Bilevel Inverse OCP. Furthermore, for the gait model of a basic walker multibody system we want periodicity constraints on the generalized coordinates to hold, such as

$$y_h^1(t_\mathrm{s}^1) = y_h^2(t_\mathrm{f}^2), \tag{5.25a}$$
$$\varphi_l^1(t_\mathrm{s}^1) = \varphi_l^2(t_\mathrm{f}^2), \tag{5.25b}$$

$$\varphi_r^1(t_\mathrm{s}^1) = \varphi_r^2(t_\mathrm{f}^2). \tag{5.25c}$$

Together with an instantaneous jump in the generalized velocities at the second phase transition between differential states $\boldsymbol{x}_2(t_\mathrm{f}^2)$ at the end of model stage 2 and differential states $\boldsymbol{x}_1(t_\mathrm{s}^1)$ at the beginning of model stage 1, the constraints (5.24) and (5.25) are already included in the second transition condition of (5.16d).

The given measurements in the Bilevel Inverse OCP allow to use extended bounds in the lower level OCP formulation. Therefore, bounds (5.16h) for the differential states

$$\begin{aligned}
-1\ell &\le x_h^\mathrm{j} \le 3\ell,\\
-5\ell &\le y_h^\mathrm{j} \le 5\ell,\\
-\pi &\le \varphi_k^\mathrm{j} \le \pi, \quad k \in \{l, r\},\\
-10\ell &\le \dot{q}_\mathrm{jk} \le 10\ell \; k \in \{0,1,2,3\},
\end{aligned}$$

can be chosen in such a way that they are not reached in the solution. In the same way extended bounds (5.16i) for the controls are chosen. In section 9.1 for $\ell = 1, M = 2, m = 1$ lower bounds $\underline{\boldsymbol{u}_\mathrm{jk}} = -10$ and upper bounds $\overline{\boldsymbol{u}_\mathrm{jk}} = 10$ were sufficient for $k \in \{0,1\}$. Furthermore, the duration parameters for each model stage have to be positive, see (5.16j).

## 5.3 Bilevel Inverse OCP of a Basic Walker Gait Model

In this section we formulate a Bilevel Inverse OCP of the form (2.21) for the identification of weights $\boldsymbol{\alpha} = \begin{pmatrix}\alpha_1 & \alpha_2\end{pmatrix}^T$ and model parameters $\boldsymbol{p} = \begin{pmatrix}\ell & M & m\end{pmatrix}^T$ in the underlying multi-stage OCP for the gait of a basic walker model. On the upper level we have a PE problem with given measurements constrained by the lower level OCP (5.16) of section 5.2, and, furthermore, conditions on the unknown objective weights and model parameters. In section 9.1 we use simulated measurements on each model stage, j = 1,2, for the position of the base segment, $x_h^\mathrm{j}$ and $y_h^\mathrm{j}$, and the angles $\varphi_l^\mathrm{j}$ and $\varphi_r^\mathrm{j}$, for a better investigation of the DISIMFAS from chapter 4 on the basic walker example - as a basic model for human locomotion. The measurements are generated by solving an OCP for some given weights $\boldsymbol{\alpha}$ and model parameters $\boldsymbol{p}$. We add multivariate normally distributed noise $\boldsymbol{\varepsilon}$ with zero mean and standard deviation $\boldsymbol{\sigma}$ to the generalized coordinates $\boldsymbol{q}_\mathrm{j}^{*\mathrm{OCP}}$ of the solution trajectory of the corresponding lower level OCP and get

$$\eta_\mathrm{jnk} = q_\mathrm{jk}^{*\mathrm{OCP}}(t_\mathrm{jn}^\mathrm{m}) + \varepsilon_\mathrm{jnk}, \qquad \mathrm{j} = 1,2,\ \mathrm{n} = 0,\dots,\mathrm{n}_\mathrm{m}^\mathrm{j} - 1,\ \mathrm{k} = 0,\dots,3. \tag{5.27}$$

For this set-up, where we have simulated measurements $\boldsymbol{\eta}$ from (5.27), we can formulate the following Bilevel Inverse OCP

$$\min_{\substack{\boldsymbol{\alpha},\boldsymbol{p},\\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \frac{1}{2}\sum_{\mathrm{j}=1}^{2}\sum_{\mathrm{n}=0}^{\mathrm{n}_\mathrm{m}^\mathrm{j}-1}\sum_{\mathrm{k}=0}^{\mathrm{n}_\mathrm{h}^\mathrm{j}-1}\frac{\left(q_\mathrm{jk}(t_\mathrm{jn}^\mathrm{m}) - \eta_\mathrm{jnk}\right)^2}{\sigma_\mathrm{jnk}^2} \tag{5.28a}$$

$$\text{s.t.} \quad (\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}) \text{ solve OCP (5.16)}, \tag{5.28b}$$

$$\alpha_1 \ge 0, \alpha_2 = 1, \tag{5.28c}$$

$$\underline{\boldsymbol{p}} \le \boldsymbol{p} \le \overline{\boldsymbol{p}}, \tag{5.28d}$$

where the weight $\alpha_2$ is set to 1 and, thus, eliminated as variable in the Bilevel Inverse OCP. This can be done, if we know that its value is not 0, which is the case in our application in section 9.1 and used at this point. Another choice is to define $\alpha_1 + \alpha_2 = 1,\ \boldsymbol{\alpha} \ge 0$ as in the general formulation (2.21) in section 2.4. Furthermore, in a more general set-up, where we have other experimental data $\boldsymbol{\eta}$ with standard deviation $\boldsymbol{\sigma}$ and a given model response

$\boldsymbol{h}$ as stated in section 2.4, we can define the upper level objective (5.28a) of the Bilevel Inverse OCP by

$$\frac{1}{2}\sum_{\mathrm{j}=1}^{2}\sum_{\mathrm{n}=0}^{\mathrm{n_m^j}-1}\sum_{\mathrm{k}=0}^{\mathrm{n_h^j}-1}\frac{\left(h_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{jn}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{jnk}}\right)^2}{\sigma_{\mathrm{jnk}}^2}.$$

All other arising quantities and functions in (5.28) are already introduced and defined in the previous two sections. Numerical results using the derived Bilevel Inverse OCP formulation for the basic walker gait model are presented in chapter 9, where the performance of our DISIMFAS from chapter 4 is investigated.

# Chapter 6

# Modeling of Cerebral Palsy Patients' Gait

Modeling the gait of patients with CP is a challenging task and already proven gait models for healthy persons, such as the HEIMAN model by Felis [52], might fail because of significant differences in their gaits. These differences in the gait models originate, e.g., from the typical club foot of CP patients, where the heels never touch the ground, as well as from their asymmetrical stature and the resulting importance to capture the full 3-D motion. In this chapter we derive a rigid multibody system model for one patient with CP and the underlying dynamics, and formulate a multi-stage OCP describing the gait based on the work of Hatz [80]. Therein, on the one side a sufficiently detailed model has to be considered to reproduce the main characteristics of the CP gait, and on the other side the computational complexity should not exceed a desired level, because the developed gait model is meant to serve as a constraint in a Bilevel Inverse OCP fitting the gait of a patient with CP to given measurement data. With the developed framework, in the future gait models for other patients with CP can be easily stated using measurements, e.g. from the HEIDELBERG MOTIONLAB [173], and furthermore investigated by solving Bilevel Inverse OCPs. In chapter 10 the discussion on modeling of CP patients' gait is continued. There, numerical results concerning the proposed dynamics reconstruction of a patient with CP from section 6.3 are given and an analysis of varying gaits is provided by solving the multi-stage OCP from section 6.4 with differently weighted optimization criteria.

## 6.1 Rigid Multibody System Model for a Patient with Cerebral Palsy

The rigid multibody system model for a patient with CP developed in this work is based on the CP gait model published in the thesis of Hatz [80, Chapter 12]. We first describe our model in detail and later give some common features of both models and emphasise their differences in section 6.5. The patient-specific rigid multibody system model with 20 DOFs is composed of 14 bodies or *segments* - head, upper, middle and lower trunk, upper and lower arms, shanks thighs, and feet - which are connected through *joints* in the way as illustrated in Figure 6.1. The lower trunk segment is also called *pelvis* and used as basis segment with its origin located in the so-called *floating base joint* or *pelvis joint* with six DOFs, which is attached to the *global reference frame*. The location and orientation of each segment or body of the patient-specific model is then defined by *local coordinate frames* at the connecting joints and their transformations based on the textbook of Featherstone [51]. The following description of the developed CP model is inspired by the HEIMAN model in the thesis of Felis [52].

### 6.1.1 Segments, Joints and DOFs

The properties of the segments which build the CP rigid multibody system model are defined by length, mass, Center of Mass (CoM) and inertia matrix of the body. In Table 6.1 the lengths and the relative masses of each segment are listed based on measurements from the HEIDELBERG MOTIONLAB, estimates by Hatz [80], and values provided in de Leva [40]. Furthermore, in Table 6.2 the relative CoMs $\tilde{\boldsymbol{r}}^{\mathrm{CoM}}$ in local coordinates and the corresponding relative radii of gyration $\tilde{\boldsymbol{r}}^{\mathrm{gyr}}$ of each segment are given, see [80] with values based on [1] and [40]. With the total body mass $m_{\mathrm{body}}$ from Table 6.4 the mass of each segment i is defined by $m_{\mathrm{i}} = m_{\mathrm{body}} \cdot \tilde{m}_{\mathrm{i}}$. Here, the subscripts i $\in$ {pelvis, thigh_r, shank_r, foot_r, thigh_l, shank_l, foot_l, m_trunk, u_trunk, u_arm_r, l_arm_r, u_arm_l, l_arm_l, head} correspond to the segment names as defined in Table 6.1. Together with segment length $l_{\mathrm{i}}$ the inertia matrix of each segment at its CoM, calculated by $\boldsymbol{r}_{\mathrm{i}}^{\mathrm{CoM}} = l_{\mathrm{i}} \cdot \tilde{\boldsymbol{r}}_{\mathrm{i}}^{\mathrm{CoM}}$, is then given by

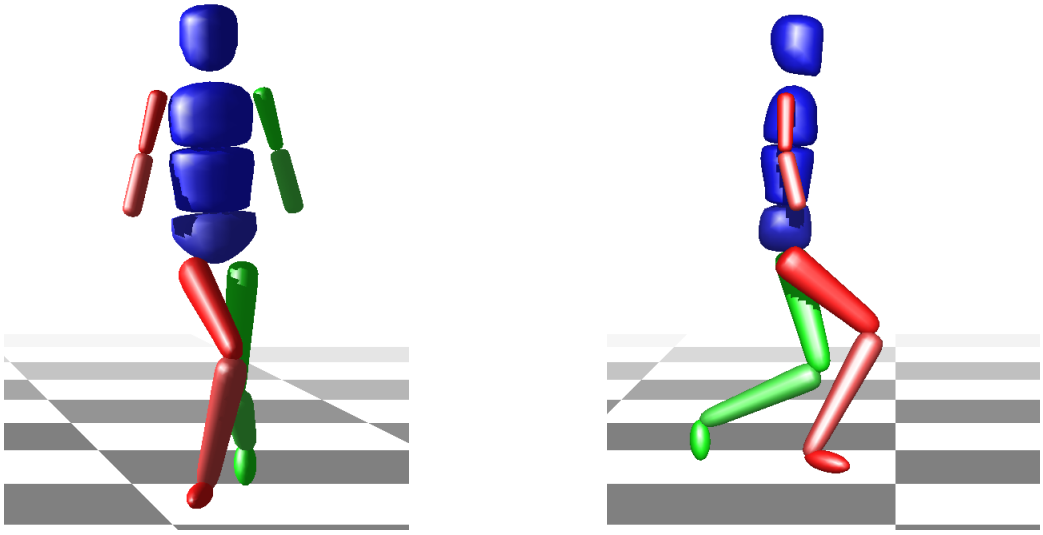

**Figure 6.1:** Model for a CP patient described by a rigid multibody system with 20 DOFs. With red segments the figure depicts the right leg and with green segments the left leg. Illustration created using MESHUP[52].

the following formula

$$\boldsymbol{I}_{\mathrm{i}}^{\mathrm{CoM}} = \begin{pmatrix} (l_{\mathrm{i}} \cdot \tilde{r}_{\mathrm{i}x}^{\mathrm{gyr}})^2 \cdot m_{\mathrm{i}} & 0 & 0 \\ 0 & (l_{\mathrm{i}} \cdot \tilde{r}_{\mathrm{i}y}^{\mathrm{gyr}})^2 \cdot m_{\mathrm{i}} & 0 \\ 0 & 0 & (l_{\mathrm{i}} \cdot \tilde{r}_{\mathrm{i}z}^{\mathrm{gyr}})^2 \cdot m_{\mathrm{i}} \end{pmatrix}.$$

Additional quantities needed for setting up the rigid multibody system model for a CP patient are listed in Table 6.3.

In the following we give the locations, the corresponding DOFs and the connections to neighboring segments of each joint in the CP multibody system model. We use *local coordinate frames* at each joint and describe the relative translation and rotation of the *child segment's* frame to the coordinate frame of the *parent segment*, see Figure 6.2 for an illustration on the locations of the local coordinate frames in our rigid multibody system model for a CP patient. Translations are defined by 3-D linear displacement vectors denoted by $\boldsymbol{r} \in \mathbb{R}^3$, whereas rotations are defined by matrices $\boldsymbol{E} \in \mathbb{R}^{3\times3}$. In general, with angles $\boldsymbol{\varphi}_{\mathrm{i}}$ located at each joint i ∈ {pelvis, hip_r, hip_l, ankle_r, ankle_l} and with the notations $s_{\mathrm{j}} = \sin(\varphi_{\mathrm{ij}})$ and $c_{\mathrm{j}} = \cos(\varphi_{\mathrm{ij}})$ for j = $\{y, x, z\}$ all arising rotations of the local coordinate frames for segments around the $y$-,$x$- and $z$-axis (fixed upper body joints and knee joints excluded) are described by

$$\boldsymbol{E}^{yxz}(\boldsymbol{\varphi}_{\mathrm{i}}) = \boldsymbol{R}_y \boldsymbol{R}_x \boldsymbol{R}_z = \begin{pmatrix} c_y c_z + s_x s_y s_z & s_x s_y c_z - c_y s_z & c_x s_y \\ c_x s_z & c_x c_z & -s_x \\ s_x c_y s_z - s_y c_z & s_y s_z + s_x c_y c_z & c_x c_y \end{pmatrix}, \tag{6.1}$$

using the Euler-angle convention $YXZ$ with (3 × 3)-rotation matrices $\boldsymbol{R}_x, \boldsymbol{R}_y$ and $\boldsymbol{R}_z$ defined by

$$\boldsymbol{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_x & -s_x \\ 0 & s_x & c_x \end{pmatrix}, \quad \boldsymbol{R}_y = \begin{pmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{pmatrix}, \quad \text{and} \quad \boldsymbol{R}_z = \begin{pmatrix} c_z & -s_z & 0 \\ s_z & c_z & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{6.2}$$

see, e.g. [39]. The subscripts i ∈ {pelvis, hip_r, hip_l, ankle_r, ankle_l} in $\boldsymbol{\varphi}_{\mathrm{i}}$ correspond to the joint names in the lower body defined in the following. In section 6.2 an extra subsection 6.2.8 is dedicated to the determination of the patient-specific knee axes for the rotational DOF in the corresponding knee joints i ∈ {knee_r, knee_l} based

**Table 6.1:** Individual segment masses used in the rigid multibody system model of a CP patient taken from thesis of Hatz [80] based on own estimates, measurement data of a CP patient from HEIDELBERG MOTIONLAB, and [40].

| Segment Name | Length | [m] | Mass | [%] |
|---|---|---|---|---|
| Pelvis | $l_{\text{pelvis}}$ | 0.135 | $\tilde{m}_{\text{pelvis}}$ | 0.1117 |
| Thigh Right | $l_{\text{thigh\_r}}$ | 0.3545 | $\tilde{m}_{\text{thigh\_r}}$ | 0.1279 |
| Shank Right | $l_{\text{shank\_r}}$ | 0.3990 | $\tilde{m}_{\text{shank\_r}}$ | 0.0433 |
| Foot Right | $l_{\text{foot\_r}}$ | 0.1405 | $\tilde{m}_{\text{foot\_r}}$ | 0.0137 |
| Thigh Left | $l_{\text{thigh\_l}}$ | 0.3616 | $\tilde{m}_{\text{thigh\_l}}$ | 0.1279 |
| Shank Left | $l_{\text{shank\_l}}$ | 0.3792 | $\tilde{m}_{\text{shank\_l}}$ | 0.0433 |
| Foot Left | $l_{\text{foot\_l}}$ | 0.1175 | $\tilde{m}_{\text{foot\_l}}$ | 0.0137 |
| Middle Trunk | $l_{\text{m\_trunk}}$ | 0.17 | $\tilde{m}_{\text{m\_trunk}}$ | 0.1633 |
| Upper Trunk | $l_{\text{u\_trunk}}$ | 0.17 | $\tilde{m}_{\text{u\_trunk}}$ | 0.1596 |
| Upper Arm Right | $l_{\text{u\_arm\_r}}$ | 0.17 | $\tilde{m}_{\text{u\_arm\_r}}$ | 0.0271 |
| Lower Arm Right | $l_{\text{l\_arm\_r}}$ | 0.17 | $\tilde{m}_{\text{l\_arm\_r}}$ | 0.0162 |
| Upper Arm Left | $l_{\text{u\_arm\_l}}$ | 0.17 | $\tilde{m}_{\text{u\_arm\_l}}$ | 0.0271 |
| Lower Arm Left | $l_{\text{l\_arm\_l}}$ | 0.17 | $\tilde{m}_{\text{l\_arm\_l}}$ | 0.0162 |
| Head | $l_{\text{head}}$ | 0.1 | $\tilde{m}_{\text{head}}$ | 0.0694 |

on motion capture data. For more details on joints, coordinate frames and transformations in rigid multibody system models we refer the reader to the textbooks of Featherstone [51] and Craig [39].

**Pelvis** The joint located at the base segment - called *pelvis joint* - has six DOFs which allows three translations (along $x$-, $y$- and $z$- axis) and three rotations around the axes $y$, $x$ and $z$. The pelvis joint is connected to the origin of the global reference frame and can take values $\boldsymbol{r}_{\text{pelvis}} = \begin{pmatrix} x_{\text{pelvis}} & y_{\text{pelvis}} & z_{\text{pelvis}} \end{pmatrix}^T$ for the 3-D linear displacement and $\boldsymbol{E}_{\text{pelvis}} = \boldsymbol{E}^{yxz}(\boldsymbol{\varphi}_{\text{pelvis}})$ as defined in (6.1) for the $(3 \times 3)$ orientation matrix using the Euler-angle convention $YXZ$.

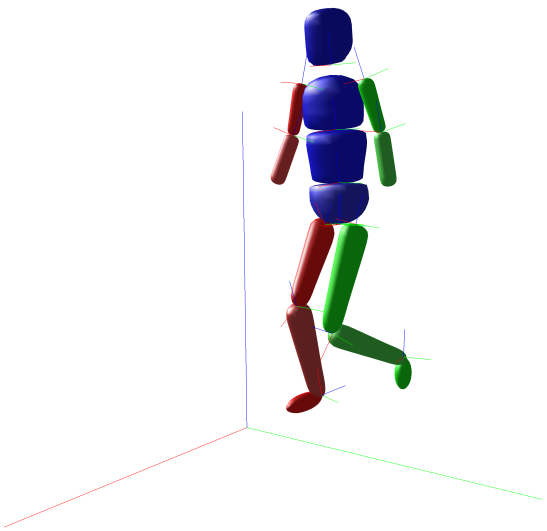

**Figure 6.2:** This figure depicts the global and local coordinate frames located at the joints of the CP rigid multibody system with 20 DOFs. The green axes depict the $y$-, the red axes the $x$-, and the blue axes the $z$-direction. Illustration created using MESHUP[52].

**Table 6.2:** Location of relative CoM for individual segments and its relative radii of gyration in % of the segment length in local coordinates used in the rigid multibody system model of a CP patient taken from thesis of Hatz [80] based on [1] and [40].

| Segment Name | Relative CoM ($\tilde{\boldsymbol{r}}^{\mathrm{CoM}}$) | Relative Radius of Gyration ($\tilde{\boldsymbol{r}}^{\mathrm{gyr}}$) |
|---|---|---|
| Pelvis | $\begin{pmatrix}0 & 0 & (1-0.6115)\end{pmatrix}^T$ | $\begin{pmatrix}0.615 & 0.551 & 0.587\end{pmatrix}^T$ |
| Thigh Right | $\begin{pmatrix}0 & 0 & -0.4095\end{pmatrix}^T$ | $\begin{pmatrix}0.329 & 0.329 & 0.149\end{pmatrix}^T$ |
| Shank Right | $\begin{pmatrix}0 & 0 & -0.4365\end{pmatrix}^T$ | $\begin{pmatrix}0.251 & 0.246 & 0.102\end{pmatrix}^T$ |
| Foot Right | $\begin{pmatrix}0 & 0 & -0.4\end{pmatrix}^T$ | $\begin{pmatrix}0.124 & 0.245 & 0.257\end{pmatrix}^T$ |
| Thigh Left | $\begin{pmatrix}0 & 0 & -0.4095\end{pmatrix}^T$ | $\begin{pmatrix}0.329 & 0.329 & 0.149\end{pmatrix}^T$ |
| Shank Left | $\begin{pmatrix}0 & 0 & -0.4395\end{pmatrix}^T$ | $\begin{pmatrix}0.251 & 0.246 & 0.102\end{pmatrix}^T$ |
| Foot Left | $\begin{pmatrix}0 & 0 & -0.4\end{pmatrix}^T$ | $\begin{pmatrix}0.124 & 0.245 & 0.257\end{pmatrix}^T$ |
| Middle Trunk | $\begin{pmatrix}0 & 0 & (1-0.4502)\end{pmatrix}^T$ | $\begin{pmatrix}0.482 & 0.383 & 0.468\end{pmatrix}^T$ |
| Upper Trunk | $\begin{pmatrix}0 & 0 & (1-0.5066)\end{pmatrix}^T$ | $\begin{pmatrix}0.505 & 0.320 & 0.465\end{pmatrix}^T$ |
| Upper Arm Right | $\begin{pmatrix}0 & 0 & -0.5772\end{pmatrix}^T$ | $\begin{pmatrix}0.285 & 0.269 & 0.158\end{pmatrix}^T$ |
| Lower Arm Right | $\begin{pmatrix}0 & 0 & -0.4574\end{pmatrix}^T$ | $\begin{pmatrix}0.276 & 0.265 & 0.121\end{pmatrix}^T$ |
| Upper Arm Left | $\begin{pmatrix}0 & 0 & -0.5772\end{pmatrix}^T$ | $\begin{pmatrix}0.285 & 0.269 & 0.158\end{pmatrix}^T$ |
| Lower Arm Left | $\begin{pmatrix}0 & 0 & -0.4574\end{pmatrix}^T$ | $\begin{pmatrix}0.276 & 0.265 & 0.121\end{pmatrix}^T$ |
| Head | $\begin{pmatrix}0 & 0 & (1-0.5002)\end{pmatrix}^T$ | $\begin{pmatrix}0.303 & 0.261 & 0.315\end{pmatrix}^T$ |

**Table 6.3:** Additional quantities to set up a rigid mulitbody system model for a CP patient taken from thesis of Hatz [80] based on measurements from HEIDELBERG MOTIONLAB [173], and the Plug-In-Gait model in the Vicon system [2].

| Quantity | Value [m] | Description |
|---|---|---|
| $d_{\mathrm{hip\_l}}$ | 0.068 | width of left hip |
| $d_{\mathrm{hip\_r}}$ | 0.068 | width of right hip |
| $d_{\mathrm{shoulder\_l}}$ | 0.068 | width of left shoulder |
| $d_{\mathrm{shoulder\_r}}$ | 0.068 | width of right shoulder |
| $d_{\mathrm{neck}}$ | 0.1 | length of neck |

**Joints of Upper Body** All joints of the upper body are fixed in an average position according to [80], because the provided motion capture data from the HEIDELBERG MOTIONLAB [173] for the chosen patient only include motions of the lower body. The connections of these joints are described in the following, where we start with the joint between the middle trunk segment and the pelvis segment, the so-called *lumbar joint*. It is located at $\boldsymbol{r}_{\mathrm{lumbar}} = \begin{pmatrix}0 & 0 & l_{\mathrm{pelvis}}\end{pmatrix}^T$. The upper body of the CP model is slightly inclined. This rotation of the local coordinate frame is defined by the rotation matrix $\boldsymbol{E}_{\mathrm{lumbar}} = \boldsymbol{E}^{yxz}(\boldsymbol{\varphi}_{\mathrm{lumbar}})$ from (6.1) with fixed angle vector

$$\boldsymbol{\varphi}_{\mathrm{lumbar}} = \begin{pmatrix}\varphi_{\mathrm{lumbar},y} & \varphi_{\mathrm{lumbar},x} & \varphi_{\mathrm{lumbar},z}\end{pmatrix}^T = \begin{pmatrix}0.12 & -0.05 & 0\end{pmatrix}^T.$$

**Table 6.4:** Total body mass and height of CP patient taken from thesis of Hatz [80] based on measurements from HEIDELBERG MOTIONLAB [173].

| Quantity | Value | Unit |
|---|---|---|
| total body mass | 39.2 | [kg] |
| total body height | 1.46 | [m] |

All following transformations of the local coordinate frames located at the joints of the upper body are characterized by a fixed 3-D linear displacement vector. The joint which connects the middle trunk segment and the upper trunk segment is called *thorax joint* and is located in the parent frame of the lumbar joint at $\boldsymbol{r}_{\text{thorax}} = \begin{pmatrix} 0 & 0 & l_{\text{m\_trunk}} \end{pmatrix}^T$. This thorax joint serves as origin of the parent coordinate frame for three joints: the *neck joint*, which connects the upper trunk segment and the head, and the *shoulder right* and *shoulder left joints*, which connect the upper trunk segment and the upper arm right and upper arm left segments, respectively. Their locations in the parent frame are defined by

$$\begin{aligned} \boldsymbol{r}_{\text{neck}} &= \begin{pmatrix} 0 & 0 & l_{\text{u\_trunk}} \end{pmatrix}^T, \\ \boldsymbol{r}_{\text{shoulder\_r}} &= \begin{pmatrix} 0 & -d_{\text{shoulder\_r}} & l_{\text{u\_trunk}} \end{pmatrix}^T, \\ \boldsymbol{r}_{\text{shoulder\_l}} &= \begin{pmatrix} 0 & d_{\text{shoulder\_l}} & l_{\text{u\_trunk}} \end{pmatrix}^T. \end{aligned}$$

Finally, the lower left and right arm segments are connected to the corresponding upper arm segments by *elbow right* and *elbow left joints* located at $\boldsymbol{r}_{\text{neck}} = \begin{pmatrix} 0 & 0 & -l_{\text{u\_arm\_r}} \end{pmatrix}^T$ and $\boldsymbol{r}_{\text{neck}} = \begin{pmatrix} 0 & 0 & -l_{\text{u\_arm\_l}} \end{pmatrix}^T$, respectively.

**Hip Right, Hip Left** The right and left *hip joints* are located in the local coordinate frame of the pelvis at $\boldsymbol{r}_{\text{hip\_r}} = \begin{pmatrix} 0 & -d_{\text{hip\_r}} & 0 \end{pmatrix}^T$ and $\boldsymbol{r}_{\text{hip\_l}} = \begin{pmatrix} 0 & d_{\text{hip\_l}} & 0 \end{pmatrix}^T$, respectively. They connect the right and left thighs with the pelvis and each of the hip joints has three rotational DOFs around the $y$-,$x$-, and $z$-axis defined by $\boldsymbol{E}_{\text{i}} = \boldsymbol{E}^{yxz}(\boldsymbol{\varphi}_{\text{i}}), \text{i} \in \{\text{hip\_r}, \text{hip\_l}\}$ from (6.1).

**Knee Right, Knee Left** The right *knee joint* is located at $\boldsymbol{r}_{\text{knee\_r}} = \begin{pmatrix} 0 & 0 & -l_{\text{thigh\_r}} \end{pmatrix}^T$ and the left *knee joint* at $\boldsymbol{r}_{\text{knee\_l}} = \begin{pmatrix} 0 & 0 & -l_{\text{thigh\_l}} \end{pmatrix}^T$ in local coordinate frames of the corresponding left and right hip joint, respectively. They connect the shanks with the thighs and each of the knee joints has only one rotational DOFs around patient-specific knee axes. The determination of the knee axes from given motion capture data is described in a following subsection 6.2.8. Here, we denote each knee axis with $\boldsymbol{k}_{\text{i}} = \begin{pmatrix} k_{\text{i},x} & k_{\text{i},y} & k_{\text{i},z} \end{pmatrix}^T, \text{i} \in \{\text{knee\_r}, \text{knee\_l}\}$ and achieve a rotation around this axis by an equivalent transformation. We first rotate the local coordinate frame described by a fixed rotation matrix

$$\boldsymbol{E}_{\text{i}} = \boldsymbol{E}^{xz}(\boldsymbol{\varphi}_{\text{i}}) = (\boldsymbol{R}_x \boldsymbol{R}_z)^T, \quad \text{i} \in \{\text{knee\_r}, \text{knee\_l}\} \tag{6.3}$$

with matrices (6.2), where the fixed angles $\varphi_{\text{i,x}}$ and $\varphi_{\text{i,z}}$ of the vector $\boldsymbol{\varphi}_{\text{i}} = \begin{pmatrix} \varphi_{\text{i,y}} & \varphi_{\text{i,x}} & \varphi_{\text{i,z}} \end{pmatrix}^T$ are defined by the equation system

$$\begin{pmatrix} k_{\text{i},x} \\ k_{\text{i},y} \\ k_{\text{i},z} \end{pmatrix} = \boldsymbol{E}^{xz}(\boldsymbol{\varphi}_{\text{i}})^T \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -s_z \\ c_x c_z \\ s_x c_z \end{pmatrix}, \quad \text{i} \in \{\text{knee\_r}, \text{knee\_l}\}, \tag{6.4}$$

with the notations $s_{\text{j}} = \sin(\varphi_{\text{ij}})$ and $c_{\text{j}} = \cos(\varphi_{\text{ij}})$ for $\text{j} = \{x, z\}$. The calculated values for the previously introduced quantities are given in Table 6.5. Each knee joint has one DOF around the local $y$-axis.

**Ankle Right, Ankle Left** The right *ankle joint* is located at $\boldsymbol{r}_{\text{ankle\_r}} = \begin{pmatrix} 0 & 0 & -l_{\text{shank\_r}} \end{pmatrix}^T$ and the left *ankle joint* at $\boldsymbol{r}_{\text{ankle\_l}} = \begin{pmatrix} 0 & 0 & -l_{\text{shank\_l}} \end{pmatrix}^T$ in local coordinate frames of the corresponding right and left knee joint, respectively. The ankle joints connect feet with shanks and each joint has three DOFs. For easier comparison to given measurements the fixed rotations of the local coordinate frames in the knee joints are first reversed by applying a rotation described by the transposed matrix $\boldsymbol{E}_{\text{i}}^T$ of (6.3) for $\text{i} \in \{\text{knee\_r}, \text{knee\_l}\}$. Then the three rotational DOFs in the ankle joints are described by a rotation matrix $\boldsymbol{E}_{\text{i}} = \boldsymbol{E}(\varphi_{\text{i}}), \text{i} = \in \{\text{ankle\_r}, \text{ankle\_l}\}$ as defined in (6.1).

**Table 6.5:** Calculated values of quantities for definition of patient-specific knee axes of CP patient based on measurements from HEIDELBERG MOTIONLAB [173] and a transformation of Euler-Angle to Axis-Angle representation.

| Quantity | Values | Description | Unit |
|---|---|---|---|
| $\boldsymbol{k}_{\text{knee\_r}}$ | $\begin{pmatrix}0.0363 & 0.9791 & 0.2000\end{pmatrix}^T$ | knee axis right | [m] |
| $\boldsymbol{\varphi}_{\text{knee\_r}}$ | $\begin{pmatrix}\varphi_{\text{knee\_r,y}} & 0.2015 & -0.0363\end{pmatrix}^T$ | fixed angles in $\boldsymbol{E}_{\text{knee\_r}}$ (6.3) | [rad] |
| $\boldsymbol{k}_{\text{knee\_l}}$ | $\begin{pmatrix}-0.0667 & 0.9801 & -0.1869\end{pmatrix}^T$ | knee axis left | [m] |
| $\boldsymbol{\varphi}_{\text{knee\_l}}$ | $\begin{pmatrix}\varphi_{\text{knee\_l,y}} & 0.1884 & 0.0667\end{pmatrix}^T$ | fixed angles in $\boldsymbol{E}_{\text{knee\_l}}$ (6.3) | [rad] |

### 6.1.2 Implementation Notes

We use the customizable implementation of the HEIMAN model described in the thesis of Felis [52] as basis and inspiration for our multibody system model of a CP patient. In the software package RBDL, which is used for the generation of equations of motion, models can be loaded from LUAMODEL files. This enables us in the future to easily use the implemented framework developed in this thesis for multiple CP patients by only customizing the LUAMODEL file after determination of the patient-specific knee axes under consideration of given measurement data.

## 6.2 Modeling of the Dynamics for a CP Patient

In this section we define required quantities, such as generalized coordinates, velocities, and accelerations, as well as active and passive generalized forces acting at the joints and arising model parameters. Together with a given foot contact model, as discussed in subsection 6.2.5, the phasewise equations of motion can then be set up. They are then used to describe a full gait cycle of the rigid multibody system model for a CP patient from section 6.1 under consideration of constraints such as, e.g. self-penetration avoidance. Furthermore, the applied methods for accessing motion capture data from the HEIDELBERG MOTIONLAB [173] and the derivation of the patient-specific knee axes in the CP gait model from this measurements are described in the following.

### 6.2.1 Full Gait Cycle and Phasewise Dynamics

According to the work of Hatz [80] we choose a full gait cycle without considering double support phases because of their minor importance, but use a different start and end point of the gait sequence with only two full single support phases instead of three. Therein, the initial and final posture coincide within a suitable range. We start with a single support phase 1 with right foot fixed to the ground. After an instantaneous transition where a perfectly inelastic collision of the left foot takes place, a single support phase 2 with left foot contact follows. The gait cycle ends when the right foot touches the ground again and a second transition with a per-

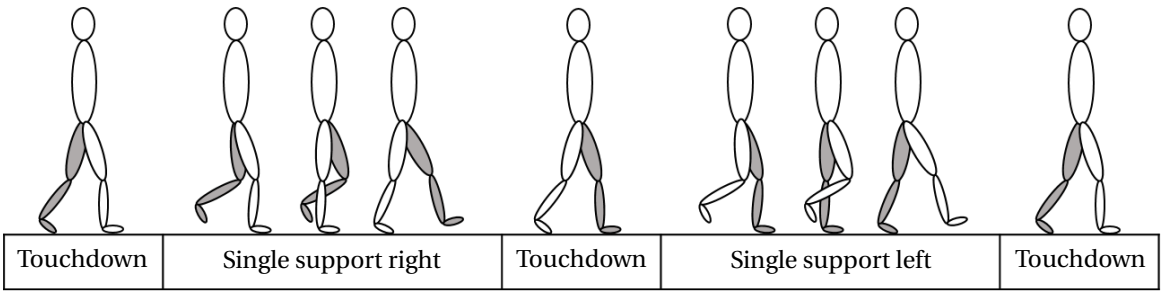


**Figure 6.3:** This figure illustrates the gait cycle considered in the CP gait model.

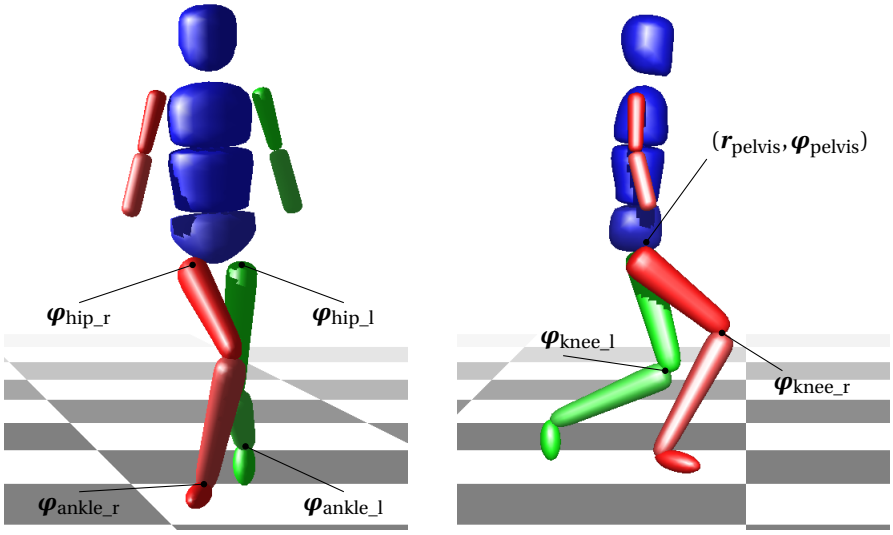


**Figure 6.4:** Model for a CP patient described by a rigid multibody system with 20 DOFs. The figure depicts the corresponding generalized coordinates. Illustration created using MESHUP[52].

fectly inelastic collision is performed. In total, two dynamic phases have to be considered and their transitions between at touch down, see Figure 6.3.

As introduced in subsection 3.1.1 the equations of motion (3.6) at each single support phase, as well as linear equation systems (3.7) for the transitions between for the rigid multibody system model of a CP patient from section 6.1 can be set up under consideration of the quantities described in the following and the given contact model of subsection 6.2.5.

### 6.2.2 Generalized Coordinates, Velocities, and Accelerations

To describe the full periodic gait cycle of the multibody system of a CP patient we start with the generalized coordinates. They include three DOFs for the position of the pelvis joint, $\boldsymbol{r}_{\text{pelvis}}(t)$, and three DOFs for its orientation, $\boldsymbol{\varphi}_{\text{pelvis}}(t)$, in the global reference frame. Furthermore, the generalized coordinates incorporate all angles, which describe the rotations of the segments connected through joints as described in the last section, denoted by $\boldsymbol{\varphi}_{\text{i}}(t)$ for i $\in$ {pelvis, hip_r, hip_l, knee_r, knee_l, ankle_r, ankle_l}. In summary, we have the following generalized coordinate vector

$$\boldsymbol{q} = \begin{pmatrix} \boldsymbol{r}_{\text{pelvis}} \\ \boldsymbol{\varphi}_{\text{pelvis}} \\ \boldsymbol{\varphi}_{\text{hip\_r}} \\ \boldsymbol{\varphi}_{\text{knee\_r}} \\ \boldsymbol{\varphi}_{\text{ankle\_r}} \\ \boldsymbol{\varphi}_{\text{hip\_l}} \\ \boldsymbol{\varphi}_{\text{knee\_l}} \\ \boldsymbol{\varphi}_{\text{ankle\_l}} \end{pmatrix} \in \mathbb{R}^{\text{n}_{\text{dofs}}}, \tag{6.5}$$

with $\text{n}_{\text{dofs}} = 20$ entries and $\boldsymbol{r}_{\text{pelvis}} = \begin{pmatrix} x_{\text{pelvis}} & y_{\text{pelvis}} & z_{\text{pelvis}} \end{pmatrix}^T$, $\boldsymbol{\varphi}_{\text{i}} = \begin{pmatrix} \varphi_{\text{i,y}} & \varphi_{\text{i,x}} & \varphi_{\text{i,z}} \end{pmatrix}^T$ for i $\in$ {pelvis, hip_r, hip_l, ankle_r, ankle_l}, and $\boldsymbol{\varphi}_{\text{i}} = \varphi_{\text{i,y}}$ for i $\in$ {knee_r, knee_l}. The time dependency is omitted for a more compact representation, which is often done in the following. Its corresponding generalized velocities are denoted by $\dot{\boldsymbol{q}} \in \mathbb{R}^{\text{n}_{\text{dofs}}}$ and accelerations by $\ddot{\boldsymbol{q}} \in \mathbb{R}^{\text{n}_{\text{dofs}}}$.

### 6.2.3 Active and Passive Joint Actuation

The rigid multibody system model is actuated through torques acting directly at the joints. Therefore, in the equations of motion the following generalized forces vector is considered:

$$\boldsymbol{\tau} = \begin{pmatrix} 0 \\ \boldsymbol{\tau}^{\text{act}} \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{\tau}_{\text{hip\_r}} \\ \boldsymbol{\tau}_{\text{knee\_r}} \\ \boldsymbol{\tau}_{\text{ankle\_r}} \\ \boldsymbol{\tau}_{\text{hip\_l}} \\ \boldsymbol{\tau}_{\text{knee\_l}} \\ \boldsymbol{\tau}_{\text{ankle\_l}} \end{pmatrix} \in \mathbb{R}^{\text{n}_{\text{dofs}}}, \tag{6.6}$$

that combines all relevant forces and torques. In our gait model for a CP patient, where each step is performed by interaction with the ground, the pelvis segment is not actuated, such that corresponding generalized forces are zero, $\boldsymbol{\tau}_{\text{pelvis}} = 0 \in \mathbb{R}^6$. Hence, the number of actuated DOFs is $\text{n}_{\text{act}} = \text{n}_{\text{dofs}} - 6$ and the actuated generalized forces or torques are denoted by $\boldsymbol{\tau}^{\text{act}} \in \mathbb{R}^{\text{n}_{\text{act}}}$. To model active actuated forces $\boldsymbol{\tau}^{\text{a}} \in \mathbb{R}^{\text{n}_{\text{act}}}$ as a result of muscle contractions on the one hand, and passive forces $\boldsymbol{\tau}^{\text{p}} \in \mathbb{R}^{\text{n}_{\text{act}}}$ caused by tendons and ligaments on the other hand, the total actuated torque is represented as a sum of both contributions and defined by

$$\boldsymbol{\tau}_{\text{i}}^{\text{act}} = \boldsymbol{\tau}_{\text{i}}^{\text{a}} + \boldsymbol{\tau}_{\text{i}}^{\text{p}}, \tag{6.7}$$

for i ∈ {hip_r, hip_l, knee_r, knee_l, ankle_r, ankle_l}. To avoid oscillations around rotation axes we introduce damping terms as, e.g. in [92, 131], as part of the passively actuated torques by

$$\boldsymbol{\tau}_{\text{i}}^{\text{damp}} = -\boldsymbol{\delta}_{\text{i}}\dot{\boldsymbol{\varphi}}_{\text{i}}, \tag{6.8}$$

with damping parameter vectors $\boldsymbol{\delta}_{\text{i}} > 0$ and joint velocity vectors $\dot{\boldsymbol{\varphi}}_{\text{i}}$ related to the corresponding generalized coordinates for i ∈ {hip_r, hip_l, knee_r, knee_l, ankle_r, ankle_l}. Furthermore, we consider so-called *passive reset forces* as part of the passively actuated torques to model the restricted range of motion in the knees of many CP patients, and specifically in the gait model developed in this thesis for one particular CP patient. With given bounds on the range of motion, denoted by $\underline{\beta}_{\text{i}}$ for the lower bound and $\overline{\beta}_{\text{i}}$ for the upper bound, passive reset forces describe the lower and upper limit of the range of motion by

$$\underline{\boldsymbol{\tau}}_{\text{i}}^{\text{reset}} = \exp\left(-\underline{\kappa}_{\text{i}}(\boldsymbol{\varphi}_{\text{i}} - \underline{\beta}_{\text{i}})\right), \tag{6.9}$$

$$\overline{\boldsymbol{\tau}}_{\text{i}}^{\text{reset}} = -\exp\left(\overline{\kappa}_{\text{i}}(\boldsymbol{\varphi}_{\text{i}} - \overline{\beta}_{\text{i}})\right), \tag{6.10}$$

respectively, with curvature parameters $\underline{\kappa}_{\text{i}}, \overline{\kappa}_{\text{i}} > 0$ and generalized coordinates $\boldsymbol{\varphi}_{\text{i}}$ for i ∈ {knee_r, knee_l}. With this approximation the *crouched* gait of patients with CP can be modeled within an acceptable amount of complexity introduced into the dynamics of our rigid multibody system model for a CP patient with 20 DOFs. This approach is inspired by [13, 125] and has been successfully applied in [151] for a case study of a simpler rigid multibody system model with 7 DOFs to model interventions of CP patients. In sum, with the definition in (6.8) for the damping terms, the passively actuated torques in equation (6.7) can be written as

$$\boldsymbol{\tau}_{\text{i}}^{\text{p}} = \boldsymbol{\tau}_{\text{i}}^{\text{damp}}, \tag{6.11}$$

for the hip and ankle joints, i ∈ {hip_r, hip_l, ankle_r, ankle_l}, and together with the definition for the reset forces in (6.9) as

$$\boldsymbol{\tau}_{\text{i}}^{\text{p}} = \boldsymbol{\tau}_{\text{i}}^{\text{damp}} + \underline{\boldsymbol{\tau}}_{\text{i}}^{\text{reset}} + \overline{\boldsymbol{\tau}}_{\text{i}}^{\text{reset}}, \tag{6.12}$$

for the knee joints, i ∈ {knee_r, knee_l}. In Table 6.6 and Table 6.7 the most frequently used quantities are summarized and in Figure 6.4 illustrated to describe the dynamics of the rigid multibody system for a CP patient.

**Table 6.6:** Definition of quantities which appear to describe the equations of motion for the rigid multibody system model for a CP patient with the definitions given in this chapter. The corresponding angular velocities and accelerations are denoted by $\dot{\boldsymbol{\varphi}}_\mathrm{i}$ and $\ddot{\boldsymbol{\varphi}}_\mathrm{i}$, respectively, for i ∈ {pelvis, hip_r, hip_l, knee_r, knee_l, ankle_r, ankle_l}. The translational velocities and accelerations at the pelvis joint are defined by $\dot{\boldsymbol{r}}_\mathrm{pelvis}$ and $\ddot{\boldsymbol{r}}_\mathrm{pelvis}$, respectively.

| Symbol | Description |
|---|---|
| $\boldsymbol{r}_\mathrm{pelvis}$ | position of pelvis joint in global reference frame |
| $\boldsymbol{\varphi}_\mathrm{pelvis}$ | orientation of local frame of pelvis joint in global reference frame |
| $\boldsymbol{\varphi}_\mathrm{hip\_r}$ | rotation of right thigh around $y, x, z$-axes in local frame of right hip joint |
| $\boldsymbol{\varphi}_\mathrm{knee\_r}$ | rotation of right shank around $y$-axis in local frame of right knee joint |
| $\boldsymbol{\varphi}_\mathrm{ankle\_r}$ | rotation of right foot around $y, x, z$-axes in local frame of right ankle joint |
| $\boldsymbol{\varphi}_\mathrm{hip\_l}$ | rotation of left thigh around $y, x, z$-axes in local frame of left hip joint |
| $\boldsymbol{\varphi}_\mathrm{knee\_l}$ | rotation of left shank around $y$-axis in local frame of left knee joint |
| $\boldsymbol{\varphi}_\mathrm{ankle\_l}$ | rotation of left foot around $y, x, z$-axes in local frame of left ankle joint |
| $\boldsymbol{\tau}_\mathrm{hip\_r}$ | right torque around $y, x, z$-axes in local frame of right hip joint |
| $\boldsymbol{\tau}_\mathrm{knee\_r}$ | right torque around $y$-axis in local frame of right knee joint |
| $\boldsymbol{\tau}_\mathrm{ankle\_r}$ | right torque around $y, x, z$-axes in local frame of right ankle joint |
| $\boldsymbol{\tau}_\mathrm{hip\_l}$ | left torque around $y, x, z$-axes in local frame of left hip joint |
| $\boldsymbol{\tau}_\mathrm{knee\_l}$ | left torque around $y$-axis in local frame of left knee joint |
| $\boldsymbol{\tau}_\mathrm{ankle\_l}$ | left torque around $y, x, z$-axes in local frame of left ankle joint |

**Table 6.7:** Definition of model parameters which appear to describe the passively actuated forces or torques ${\boldsymbol{\tau}^\mathrm{p}}_\mathrm{i}$ for each joint i ∈ {hip_r, hip_l, knee_r, knee_l, ankle_r, ankle_l} of the rigid multibody system model for a CP patient with the definitions given in this chapter.

| Model Parameter | Description |
|---|---|
| $\boldsymbol{\delta}_\mathrm{i}$ | damping parameter vector with contributions around $y, x, z$-axes for i ∈ {hip_r, hip_l, ankle_r, ankle_l} and contributions around $y$-axis for i ∈ {knee_r, knee_l} |
| $\underline{\beta}_\mathrm{i}$ | lower bound of range of motion for i ∈ {knee_r, knee_l} |
| $\overline{\beta}_\mathrm{i}$ | upper bound of range of motion for i ∈ {knee_r, knee_l} |
| $\underline{\kappa}_\mathrm{i}$ | curvature parameter of lower limit for i ∈ {knee_r, knee_l} |
| $\overline{\kappa}_\mathrm{i}$ | curvature parameter of upper limit for i ∈ {knee_r, knee_l} |

### 6.2.4 Patient-Specific Model Parameters

We define all arising parameters described in the last subsection and listed in Table 6.7 as patient-specific model parameters and summarize them into a vector $\boldsymbol{p} = \begin{pmatrix} \boldsymbol{\delta}^T & \boldsymbol{\zeta}^T \end{pmatrix}^T \in \mathbb{R}^{\mathrm{n_p}}$, where all damping parameters are combined in

$$\boldsymbol{\delta} := \begin{pmatrix} \boldsymbol{\delta}_\mathrm{hip\_r}^T & \boldsymbol{\delta}_\mathrm{hip\_l}^T & \boldsymbol{\delta}_\mathrm{knee\_r}^T & \boldsymbol{\delta}_\mathrm{knee\_l}^T & \boldsymbol{\delta}_\mathrm{ankle\_r}^T & \boldsymbol{\delta}_\mathrm{ankle\_l}^T \end{pmatrix}^T,$$

and all parameters corresponding to the restricted range of motion in the knees in

$$\boldsymbol{\zeta} := \begin{pmatrix} \underline{\beta}_\mathrm{knee\_r} & \overline{\beta}_\mathrm{knee\_r} & \underline{\kappa}_\mathrm{knee\_r} & \overline{\kappa}_\mathrm{knee\_r} & \underline{\beta}_\mathrm{knee\_l} & \overline{\beta}_\mathrm{knee\_l} & \underline{\kappa}_\mathrm{knee\_l} & \overline{\kappa}_\mathrm{knee\_l} \end{pmatrix}^T.$$

Each of these parameters depends on the particular CP patient and the given motion capture data. A suitable subset of identifiable model parameters can be determined by solving a Bilevel Inverse OCP as stated in sec-

tion 6.6, where a PE Problem, which minimizes the deviation of the model response to given measurements, is constrained by the multi-stage OCP describing the gait of a patient with CP. In general, the Bilevel Inverse OCP (6.42), as well as the dynamics reconstruction (6.18) of a CP gait, which we both derive in the following, may have more than one local solution for varying parameter realizations. Hence, a suitable subset of unknown model parameters in both problem formulations has to be considered with structurally and practically identifiable parameters as defined in the thesis of Sommer [161].

### 6.2.5 Foot Contact Model and Self-Penetration Constraints

We model ankle joints and feet contacts with the ground at the tip of each foot as, e.g. in [49, 80]. This enables us to capture the typical gait of CP patients, where often no contact at the heels takes place, However, in our gait model contrary to [80], where only one point per foot serves as contact with the ground, we consider two contact points per feet $\boldsymbol{\rho}_i^k$ for $k \in \{\text{hal,met5}\}$ approximately at Hallux (hal) and fifth metatarsal (met5) positions at each foot, $i \in \{\text{foot\_l,foot\_r}\}$, see Figure 6.5.

**Figure 6.5:** On the left subfigure the positions of the foot contact at Hallux point (blue circle) and fifth Metatarsal point (green circle) at each foot are illustrated, picture designed by Freepik [62]. They are transferred to the derived gait model for a CP patient as shown on the right subfigure (red circles are the contact points). Illustration created using MESHUP[52].

The constraint vectors can then be written as

$$\boldsymbol{g}_i(\boldsymbol{q}) = \begin{pmatrix} g_{i,x}^{\text{hal}}(\boldsymbol{q}) \\ g_{i,y}^{\text{hal}}(\boldsymbol{q}) \\ g_{i,z}^{\text{hal}}(\boldsymbol{q}) \\ g_{i,x}^{\text{met5}}(\boldsymbol{q}) \\ g_{i,z}^{\text{met5}}(\boldsymbol{q}) \end{pmatrix} \in \mathbb{R}^5, \quad \text{for} \quad i \in \{\text{foot\_r}, \text{foot\_l}\}, \tag{6.13}$$

at the right and left foot, respectively.

In section 6.4 we state a multi-stage OCP formulation for the gait of a rigid multibody system model for a CP patient. Therein, for differently weighted optimization criteria varying gaits can be synthesized. To avoid self-penetration of the segments in the rigid multibody system model, constraints have to be fulfilled in the solution of the corresponding OCP. We introduce points $\boldsymbol{\rho}_i(\boldsymbol{q}) \in \mathbb{R}^3$ for $i = 0, \ldots, n_\rho$ on the lower body segments of the CP model and ensure that the euclidean distances

$$\boldsymbol{\delta}_k^{\text{pen}}(\boldsymbol{q}) = \|\boldsymbol{\rho}_i(\boldsymbol{q}) - \boldsymbol{\rho}_j(\boldsymbol{q})\|_2 > \epsilon^{\text{pen}}, \qquad \text{for} \quad k = 0, \ldots, n_{\text{pen}} - 1, \tag{6.14}$$

are always greater than some given threshold $\epsilon^{\text{pen}}$ of $n_{\text{pen}}$ selected pairs $(i,j)$, with $i,j \in \{0, \ldots, n_\rho\}$ such that $i \neq j$. In the reconstruction of the dynamics in section 6.3, as well as in the Bilevel Inverse OCP stated in section 6.6, where the objectives are least-squares terms, which minimize the deviation of the model response to some given motion capture data, these self-penetration constraints can be neglected.

### 6.2.6 Accessing Motion Capture Data

The measurements taken in this thesis for modeling the gait of a CP patient are provided by the HEIDELBERG MOTIONLAB [173] using the Vicon motion capture system with a corresponding Plug-In-Gait model [2]. As a result of the used Plug-In-Gait model offsets have to be considered to reproduce given motion capture data as described by Hatz [80] in more detail. One important remark is that in our calculations these offsets, which are summarized in Table 6.8, are already applied on the used measurement data for dynamics reconstruction, such that they do not enter the gait model anymore in contrast to [80].

**Table 6.8:** Values of offsets added to measured Euler angles $\varphi_i^v$ of the Plug-In-Gait model in the Vicon system [2] used at the HEIDELBERG MOTIONLAB [173] for i ∈ {knee_r, knee_l} around the $z$-axes and for i ∈ {ankle_r, ankle_l} around the $y, x-$ axes; source HUMANS implementations of the CP model used in Hatz [80].

| Offsets | Value [rad] | Description |
|---|---|---|
| $o_{\text{knee\_r,z}}$ | 0.1011 | offset added to measured angles $\varphi^m_{\text{knee\_r,z}} = \varphi^v_{\text{knee\_r,z}} + o_{\text{knee\_r,z}}$ |
| $o_{\text{ankle\_r,y}}$ | −1.5133 | offset added to measured angles $\varphi^m_{\text{ankle\_r,y}} = \varphi^v_{\text{ankle\_r,y}} + o_{\text{ankle\_r,y}}$ |
| $o_{\text{ankle\_r,x}}$ | −0.5932 | offset added to measured angles $\varphi^m_{\text{ankle\_r,x}} = \varphi^v_{\text{ankle\_r,x}} + o_{\text{ankle\_r,x}}$ |
| $o_{\text{knee\_l,z}}$ | −0.1843 | offset added to measured angles $\varphi^m_{\text{knee\_l,z}} = \varphi^v_{\text{knee\_l,z}} + o_{\text{knee\_l,z}}$ |
| $o_{\text{ankle\_l,y}}$ | −1.5463 | offset added to measured angles $\varphi^m_{\text{ankle\_l,y}} = \varphi^v_{\text{ankle\_l,y}} + o_{\text{ankle\_l,y}}$ |
| $o_{\text{ankle\_l,x}}$ | 0.5666 | offset added to measured angles $\varphi^m_{\text{ankle\_l,x}} = \varphi^v_{\text{ankle\_l,x}} + o_{\text{ankle\_l,x}}$ |

We use the open-source biomechanical toolkit BTK [3] to access the motion capture data generated within the Plug-In-Gait model provided by Vicon, which is stored in acquisition files in c3d format. The data incorporates recorded motion capture data of the whole measuring process at each time frame, and other static measurements, such as, e.g., body mass and height, as well as positions of joint centers. We extract data of one gait cycle, which starts at take-off of the left foot, and ends after performing two steps at a second take-off of the same foot. The given data is then processed within an own MATLAB implementation using an interface to BTK for accessing the needed data for our purposes.

### 6.2.7 Creation of a Digital Twin

On the one side, we generate initial guesses for the reconstruction of the dynamics in section 6.3 of the previously introduced rigid multibody system model of a CP patient in section 6.1 based on all joint angles provided by the Vicon Plug-In-Gait model at the corresponding time frames. On the other side, for the generation of measurements used in the dynamics reconstruction we again consider all joint angles provided by the Vicon Plug-In-Gait model at the corresponding time frames and transform them as described in the following. We develop a second rigid multibody system model as a *digital twin* for a CP patient in the same way as before in section 6.1, but with three DOFs in each knee joint. The provided angles by the Vicon Plug-In-Gait model can now be used directly within this model for motion generation of the whole gait cycle after processing and applying offsets in our MATLAB implementation. We specify identical points on the lower bodies in the LUAMODELS as illustrated in Figure 6.6 of both models - the previously developed CP model with 1-DOF knee axes for dynamics reconstruction and the CP model with 3-DOFs knee axes, which is used as reference. On the latter reference model the measured and processed Vicon joint angles are applied, such that the positions of the points in the global reference frame at each time frame of the gait cycle are used as measurements denoted by $\boldsymbol{\eta}^P \in \mathbb{R}^{n_{meas}}$ in the OCP formulation (6.18) later introduced in section 6.3. Therein, the deviation of the positions of the corresponding points of the original CP model with patient-specific 1-DOF knee axes to these generated measurements is minimized by solving the multi-stage OCP for the reconstruction of the dynamics. Furthermore, the local knee joint angles based on the Vicon Plug-In-Gait model are transformed to an adequate format, such that they can be used in the following section for the determination of patient-specific knee axes.

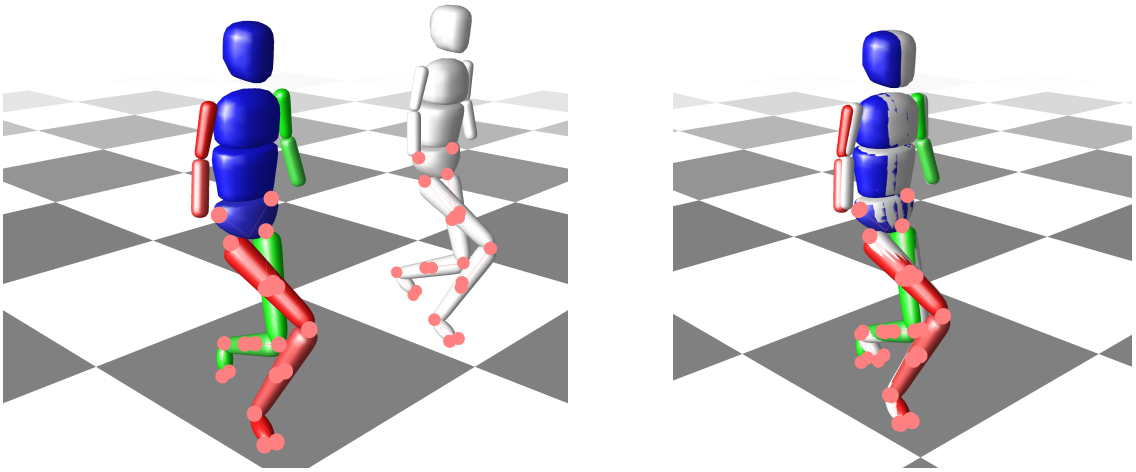

**Figure 6.6:** The left subfigure depicts selected points, which are attached to the rigid multibody system model for a CP patient with 1−DOF in each knee (left body), and the corresponding multibody system with 3−DOFs in each knee (right body in grey) for comparison to given motion capture data via Vicon joint angles. The right subfigure shows the overlap of both models while minimizing the deviation of the corresponding points at selected measurement time. Illustration created using MESHUP[52].

### 6.2.8 Determination of Patient-Specific Knee Axes

For the determination of the knee axes in the developed rigid multibody system model of a CP patient of section 6.1 we use the processed local knee joint angles based on the Vicon Plug-In-Gait model measured at each time frame of the whole gait cycle. These extracted Euler angles at time points $n \in \{0, \dots, n_m - 1\}$ are denoted by

$$(\boldsymbol{\varphi}_i^m)_n = \left((\varphi_{i,y}^m)_n \quad (\varphi_{i,x}^m)_n \quad (\varphi_{i,z}^m)_n\right)^T,$$

at right and left knee joints $i \in \{\text{knee\_r, knee\_l}\}$ with added offsets to the angles around the $z$-axes as described in Table 6.8. They are used to construct rotation matrices

$$\begin{aligned}(\boldsymbol{R}_i^{yxz})_n = \boldsymbol{R}^{yxz}((\boldsymbol{\varphi}_i^m)_n) &= \begin{pmatrix} c_y c_z + s_x s_y s_z & s_x s_y c_z - c_y s_z & c_x s_y \\ c_x s_z & c_x c_z & -s_x \\ s_x c_y s_z - s_y c_z & s_y s_z + s_x c_y c_z & c_x c_y \end{pmatrix} \\ &= \begin{pmatrix} (r_i^{00})_n & (r_i^{01})_n & (r_i^{02})_n \\ (r_i^{10})_n & (r_i^{11})_n & (r_i^{12})_n \\ (r_i^{20})_n & (r_i^{21})_n & (r_i^{22})_n \end{pmatrix},\end{aligned}$$

at each time point for the right and left knee with the notations $s_j = \sin(\varphi_{ij}^m)_n$ and $c_j = \cos(\varphi_{ij}^m)_n$ for $j = \{y, x, z\}$ using the Euler-angle convention $YXZ$ as in (6.1). With these rotation matrices $(\boldsymbol{R}_i^{yxz})_n$ for each knee joint $i \in \{\text{knee\_r, knee\_l}\}$ the Euler angles can be transformed to angle-axis-representations as, e.g., described in the book of Craig [39] by

$$(\theta_i)_n = \arccos\left(\frac{(r_i^{00})_n + (r_i^{11})_n + (r_i^{22})_n - 1}{2}\right), \tag{6.15a}$$

$$(\boldsymbol{k}_i)_n = \frac{1}{2\sin(\theta_i)_n}\begin{pmatrix} (r_i^{21})_n - (r_i^{12})_n \\ (r_i^{02})_n - (r_i^{20})_n \\ (r_i^{10})_n - (r_i^{01})_n \end{pmatrix}. \tag{6.15b}$$

For the determination of the resulting patient-specific knee axes $\boldsymbol{k}_\mathrm{i}$ at the right and left knee joints $\mathrm{i} \in \{\text{knee\_r}, \text{knee\_l}\}$, equation (6.15b) is considered and the normalized means of all corresponding vectors $(\boldsymbol{k}_\mathrm{i})_\mathrm{n}$ at time points $\mathrm{n} \in \{0, \ldots, \mathrm{n_m} - 1\}$ are calculated. The values for $\boldsymbol{k}_\mathrm{i}$ are summarized in Table 6.5. The resulting fixed angles $\varphi_{\mathrm{i,x}}$ and $\varphi_{\mathrm{i,z}}$ of the vector $\boldsymbol{\varphi}_\mathrm{i} = \begin{pmatrix} \varphi_{\mathrm{i,y}} & \varphi_{\mathrm{i,x}} & \varphi_{\mathrm{i,z}} \end{pmatrix}^T$ are then defined by the equation system (6.4).

### 6.2.9 Implementation Notes

There exist various software tools to set up the equations of motion, e.g. the HUMANS toolbox [1] and the RBDL C++ software package by Felis [53] developed in the group of Mombaur at Heidelberg University. In this work we decided to calculate the dynamics with the efficient implementation in RBDL of some essential dynamics algorithms based on Featherstone [51]. Among other algorithms it contains the Articulated Body Algorithm for forward dynamics, and the Composite Rigid Body Algorithm for an efficient computation of the joint space inertia matrix. Its successful use is proven in many publications, such as [54, 49, 38, 92], to mention only a few.

## 6.3 Dynamics Reconstruction to Motion Capture Data as a Multi-Stage OCP with Least-Squares Objective

Before we state the multi-stage OCP formulation in section 6.4 describing the gait of a CP patient, we first reconstruct the underlying dynamics to comply with formulated constraints under consideration of given motion capture data provided by the HEIDELBERG MOTIONLAB [173]. This reconstruction is already a multi-stage OCP with a least-squares objective function tracking measurements of a sequence of two steps performed by the CP patient. It serves as starting point for the analysis and synthesis of gaits with differently weighted objectives in section 6.4 and fixed model parameters, and gives already a first hint if the dynamics of the developed multibody system model with the patient-specific knee axes can reproduce the gait of the CP patient under consideration of an adequate contact model and other suitable constraints describing two steps.

We start with the definition of optimization variables as introduced in (3.11) in subsection 3.1.2. The differential states are phasewise defined by

$$\boldsymbol{x}_\mathrm{j} = \begin{pmatrix} \boldsymbol{q}_\mathrm{j} \\ \dot{\boldsymbol{q}}_\mathrm{j} \\ \boldsymbol{\tau}_\mathrm{j}^\mathrm{a} \end{pmatrix} \in \mathbb{R}^{2\mathrm{n_{dofs}} + \mathrm{n_{act}}}, \tag{6.16}$$

with model stage index $\mathrm{j} = 1, 2$. The first component of $\boldsymbol{x}_\mathrm{j}$ represents the generalized coordinates $\boldsymbol{q}_\mathrm{j} \in \mathbb{R}^{\mathrm{n_{dofs}}}$ defined in (6.5) for the rigid multibody system model of a CP patient with $\mathrm{n_{dofs}} = 20$. The second component includes the corresponding generalized velocities denoted by $\dot{\boldsymbol{q}}_\mathrm{j} \in \mathbb{R}^{\mathrm{n_{dofs}}}$. As third component the active joint forces denoted by $\boldsymbol{\tau}_\mathrm{j}^\mathrm{a} \in \mathbb{R}^{\mathrm{n_{act}}}$, $\mathrm{n_{act}} = \mathrm{n_{dofs}} - 6 = 14$ are set, which are defined according to (6.6) and (6.7) described in subsection 6.2.3. The passive forces $\boldsymbol{\tau}_\mathrm{j}^\mathrm{p} \in \mathbb{R}^{\mathrm{n_{act}}}$ enter the equations of motion for the rigid multibody system model with (6.11) at the hips and ankle joints, and with (6.12) at the knee joints. The controls are set to the derivatives of the active generalized forces as

$$\boldsymbol{u}_\mathrm{j} = \dot{\boldsymbol{\tau}}_\mathrm{j}^\mathrm{a} \in \mathbb{R}^{\mathrm{n_{act}}}. \tag{6.17}$$

With all later introduced constraints in subsection 6.3.3 together with the least-squares objective function (6.20), the dynamics (6.21), and transition conditions (6.22) the multi-stage OCP for the reconstruction of given motion capture data is set up completely to meet the dynamics and constraints describing two steps of the rigid

multibody system model for a CP patient and reads as follows:

$$\min_{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{u}_1,\boldsymbol{u}_2,\hat{\boldsymbol{p}}} \quad \frac{1}{2}\sum_{\mathrm{j}=1}^{2}\sum_{\mathrm{n}=0}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1}\sum_{\mathrm{k}=0}^{\mathrm{n}_{\mathrm{h}}^{\mathrm{j}}-1}\frac{\left(h_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{jn}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{jnk}}^{\mathrm{P}}\right)^2}{\sigma_{\mathrm{jnk}}^2}+\gamma^{\mathrm{R}}\sum_{\mathrm{j}=1}^{2}\int_{t_{\mathrm{s}}^{\mathrm{j}}}^{t_{\mathrm{f}}^{\mathrm{j}}}\left(\boldsymbol{W}^{\boldsymbol{t}}\|\boldsymbol{u}_{\mathrm{j}}(t)\|_2^2\right)\mathrm{d}t \tag{6.18a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_{\mathrm{j}}(t) = d_{\mathrm{j}}\cdot\boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t\in\mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.18b}$$

$$\boldsymbol{x}_2(t_{\mathrm{s}}^2) = \Delta_1(\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{p}), \tag{6.18c}$$

$$\boldsymbol{x}_1(t_{\mathrm{s}}^1) = \Delta_2(\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.18d}$$

$$0 \le \boldsymbol{c}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t\in\mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.18e}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{x}_1(t_{\mathrm{s}}^1),\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.18f}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.18g}$$

$$\underline{\boldsymbol{x}_{\mathrm{j}}} \le \boldsymbol{x}_{\mathrm{j}}(t) \le \overline{\boldsymbol{x}_{\mathrm{j}}}, \qquad t\in\mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.18h}$$

$$\underline{\boldsymbol{u}_{\mathrm{j}}} \le \boldsymbol{u}_{\mathrm{j}}(t) \le \overline{\boldsymbol{u}_{\mathrm{j}}}, \qquad t\in\mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.18i}$$

$$d_{\mathrm{j}}^{\mathrm{m}} = d_{\mathrm{j}}, \qquad \mathrm{j}=1,2, \tag{6.18j}$$

$$\underline{\hat{\boldsymbol{p}}} \le \hat{\boldsymbol{p}} \le \overline{\hat{\boldsymbol{p}}}. \tag{6.18k}$$

It is defined on fixed and *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [t_{\mathrm{s}}^{\mathrm{j}}, t_{\mathrm{f}}^{\mathrm{j}}] = [0,1]$ with fixed stage duration parameters $d_{\mathrm{j}}$ to given measurements $d_{\mathrm{j}}^{\mathrm{m}}$ for each model stage $\mathrm{j}=1,2$ and stage transition times before and after collision, $t_{\mathrm{f}}^1 = t_{\mathrm{f}}^2 = 1$ and $t_{\mathrm{s}}^2 = t_{\mathrm{s}}^1 = 0$, respectively, as introduced in subsection 2.1.2 together with the remaining variables and functions. Some of the arising model parameters $\boldsymbol{p}$ defined in subsection 6.2.4 are fixed to estimated values based on measurements provided by the HEIDELBERG MOTIONLAB [173], $\underline{\beta}_{\mathrm{i}}, \overline{\beta}_{\mathrm{i}}, \mathrm{i} = \{\text{knee\_r, knee\_r}\}$. The remaining damping parameters $\boldsymbol{\delta}$ and curvature parameters $\underline{\kappa}_{\mathrm{i}}, \overline{\kappa}_{\mathrm{i}}, \mathrm{i} = \{\text{knee\_r, knee\_r}\}$ are combined in the parameter vector $\hat{\boldsymbol{p}}$, which can be estimated within the reconstruction by solving (6.18). In the following we give a more detailed explanation of all other quantities that occur in OCP (6.18).

### 6.3.1 Least-Squares Objective Function

For the reconstruction of a measured gait cycle we use a least-squares objective. With this choice the deviation of the model response to given motion capture data is minimized, such that all constraints are fulfilled to perform two full steps under consideration of the dynamics for the CP multibody system model from section 6.1 and its transitions between. The least-squares objective function (6.18a) comprises measurements $\boldsymbol{\eta}^{\mathrm{P}} \in \mathbb{R}^{\mathrm{n_{meas}}}$ from a real-world process observed at specific time points, which was processed as described in subsection 6.2.6 and subsection 6.2.7. After time transformations on fixed *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [0,1]$ for each model stage these time points are defined as

$$0 = t_{\mathrm{j},0}^{\mathrm{m}} < t_{\mathrm{j},1}^{\mathrm{m}} < \ldots < t_{\mathrm{j},\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1}^{\mathrm{m}} = 1, \quad \mathrm{j}=1,2. \tag{6.19}$$

The corresponding model response function $\boldsymbol{h}_{\mathrm{j}} : \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_p}} \to \mathbb{R}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}} \times \mathrm{n}_{\mathrm{h}}^{\mathrm{j}}}$ is evaluated at these time points. We assume that the measurements $\eta_{\mathrm{jnk}}^{\mathrm{P}}$ are afflicted with independent, additive, and normally distributed errors $\varepsilon_{\mathrm{jnk}}$ with zero mean and variances $\sigma_{\mathrm{jnk}}^2$. Then for a correct model with true parameters $\boldsymbol{p}^*$ we get

$$\eta_{\mathrm{jnk}} = h_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{jn}}^{\mathrm{m}}),\boldsymbol{p}^*) + \varepsilon_{\mathrm{jnk}}, \qquad \mathrm{j}=1,2,\ \mathrm{n}=0,\ldots,\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1,\ \mathrm{k}=0,\ldots,\mathrm{n}_{\mathrm{h}}^{\mathrm{j}}-1.$$

Furthermore, we add a regularization term to the least-squares objective to avoid oscillations in the controls. It is weighted by a diagonal matrix $\boldsymbol{W}^{\boldsymbol{t}} \in \mathbb{R}^{\mathrm{n_{act}} \times \mathrm{n_{act}}}$ to account for different order of magnitude and scaled appro-

priately by $\gamma^{\mathrm{R}} \in \mathbb{R}$ . In sum, the objective is then defined by

$$\frac{1}{2}\sum_{\mathrm{j}=1}^{2}\sum_{\mathrm{n}=0}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1}\sum_{\mathrm{k}=0}^{\mathrm{n}_{\mathrm{h}}^{\mathrm{j}}-1}\frac{\left(h_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{jn}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{jnk}}^{\mathrm{P}}\right)^2}{\sigma_{\mathrm{jnk}}^2}+\gamma^{\mathrm{R}}\sum_{\mathrm{j}=1}^{2}\int_{t_{\mathrm{s}}^{\mathrm{j}}}^{t_{\mathrm{f}}^{\mathrm{j}}}\left(\boldsymbol{W}^{\dot{\boldsymbol{\tau}}}\|\boldsymbol{u}_{\mathrm{j}}(t)\|_2^2\right)\mathrm{d}t. \tag{6.20}$$

### 6.3.2 Dynamics and its Transitions

The dynamics on each model stage j = 1,2 are summarized in (6.18b) in the multi-stage OCP formulation and include the following ODE system

$$\dot{\boldsymbol{x}}_{\mathrm{j}}(t) = d_{\mathrm{j}}\cdot\boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}) \tag{6.21a}$$

$$= d_{\mathrm{j}}\cdot\begin{pmatrix}\dot{\boldsymbol{q}}_{\mathrm{j}}(t)\\ \ddot{\boldsymbol{q}}_{\mathrm{j}}(t)\\ \boldsymbol{u}_{\mathrm{j}}(t)\end{pmatrix}, \qquad t\in\mathcal{T}^{\mathrm{j}}=[t_{\mathrm{s}}^{\mathrm{j}},t_{\mathrm{f}}^{\mathrm{j}}], \tag{6.21b}$$

where the generalized acceleration $\ddot{\boldsymbol{q}}_{\mathrm{j}}$ is part of the solution of the equations of motion (3.15) for the rigid multi-body system model of a CP patient from section 6.1 with the quantities defined in section 6.2. Its transitions are denoted by

$$\boldsymbol{x}_2(t_{\mathrm{s}}^2)=\Delta_1(\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{p}) \quad\text{and}\quad \boldsymbol{x}_1(t_{\mathrm{s}}^1)=\Delta_2(\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.22}$$

where the first condition enters the OCP in (6.18c) and couples the differential states $\boldsymbol{x}_1$ of model stage 1 at final time $t_{\mathrm{f}}^1$ with the differential states $\boldsymbol{x}_2$ of model stage 2 at initial time $t_{\mathrm{s}}^2$. The second condition in (6.18d) maps the differential states $\boldsymbol{x}_2$ of model stage 2 at final time $t_{\mathrm{f}}^2$ to the differential states $\boldsymbol{x}_1$ of model stage 1 at initial time $t_{\mathrm{s}}^1$. This is a small difference to the general problem formulation of (3.8) but simplifies the formulation of periodicity constraints. For the first transition $\Delta_1(\cdot)$ the generalized coordinates $q_1(t_{\mathrm{f}}^1)=q_2(t_{\mathrm{s}}^2)$ stay the same at collision. Whereas in the second transition condition $\Delta_2(\cdot)$, terminal condition (6.32) and periodicity condition (6.33) are included, see subsection 6.3.3. The instantaneous change in the generalized velocities $\dot{\boldsymbol{q}}_{\mathrm{j}}$ can be calculated using the general formulation (3.16) with the given quantities of the previous sections.

### 6.3.3 Constraints

In this subsection we formulate path constraints (6.18e) and multi-point boundary constraints (6.18f) and (6.18g) arising in the multi-stage OCP for reconstruction of the gait of a CP patient. Furthermore, as mentioned above, periodicity conditions and terminal conditions are given and combined in the second transition condition (6.18d). All introduced constraints here also arise in the multi-stage OCP formulation of section 6.4 minimizing a chosen weighted combination of optimization criteria to describe synthesized gaits of a CP patient. In the latter OCP formulation the constraints introduced at this point are furthermore extended by self-penetration constraints as defined in subsection 6.2.5.

We denote the two contact points k ∈ {hal,met5} per foot and the corresponding velocities in global coordinates by

$$\boldsymbol{\rho}_{\mathrm{i}}^{\mathrm{jk}}=\begin{pmatrix}\rho_{\mathrm{i,x}}^{\mathrm{jk}} & \rho_{\mathrm{i,y}}^{\mathrm{jk}} & \rho_{\mathrm{i,z}}^{\mathrm{jk}}\end{pmatrix}^T,$$
$$\dot{\boldsymbol{\rho}}_{\mathrm{i}}^{\mathrm{jk}}=\begin{pmatrix}\dot{\rho}_{\mathrm{i,x}}^{\mathrm{jk}} & \dot{\rho}_{\mathrm{i,y}}^{\mathrm{jk}} & \dot{\rho}_{\mathrm{i,z}}^{\mathrm{jk}}\end{pmatrix}^T,$$

at the left and right foot i ∈ {foot_l,foot_r} with model stage index j = 1,2. The same contact points are also used in subsection 6.2.5 to set up the constraint set (6.13). For the arising ground reaction forces at the left and right

foot we write

$$\boldsymbol{v}_{\mathrm{i}}^{\mathrm{jk}} = \begin{pmatrix} v_{\mathrm{i,x}}^{\mathrm{jk}} & v_{\mathrm{i,y}}^{\mathrm{jk}} & v_{\mathrm{i,z}}^{\mathrm{jk}} \end{pmatrix}^T.$$

The positions of joints i ∈ {hip_r, hip_l, knee_r, knee_l, ankle_r, ankle_l} in the global reference frame are denoted by $\boldsymbol{\rho}_{\mathrm{i}}^{\mathrm{j}} = \begin{pmatrix} \rho_{\mathrm{i,x}}^{\mathrm{j}} & \rho_{\mathrm{i,y}}^{\mathrm{j}} & \rho_{\mathrm{i,z}}^{\mathrm{j}} \end{pmatrix}^T$ and the corresponding velocities by $\dot{\boldsymbol{\rho}}_{\mathrm{i}}^{\mathrm{j}} = \begin{pmatrix} \dot{\rho}_{\mathrm{i,x}}^{\mathrm{j}} & \dot{\rho}_{\mathrm{i,y}}^{\mathrm{j}} & \dot{\rho}_{\mathrm{i,z}}^{\mathrm{j}} \end{pmatrix}^T$.

#### Initial Conditions

At initial time $t_{\mathrm{s}}^1 = 0$ the vertical position of both feet is fixed to the ground $\eta_{\mathrm{ground}}$ and the position of the pelvis joint $\boldsymbol{r}_{\mathrm{pelvis}}^1 = \begin{pmatrix} x_{\mathrm{pelvis}}^1 & y_{\mathrm{pelvis}}^1 & z_{\mathrm{pelvis}}^1 \end{pmatrix}$ in $x$ and $y$ direction is fixed to measurements $\eta_{\mathrm{pelvis,x}}^{\mathrm{s}}$ and $\eta_{\mathrm{pelvis,y}}^{\mathrm{s}}$, respectively,

$$\begin{aligned}
\rho_{\mathrm{foot_r,z}}^{1,\mathrm{hal}}(t_{\mathrm{s}}^1) &= \eta_{\mathrm{ground}}, \\
\rho_{\mathrm{foot_r,z}}^{1,\mathrm{met5}}(t_{\mathrm{s}}^1) &= \eta_{\mathrm{ground}}, \\
\rho_{\mathrm{foot_l,z}}^{1,\mathrm{hal}}(t_{\mathrm{s}}^1) &= \eta_{\mathrm{ground}}, \\
\rho_{\mathrm{foot_l,z}}^{1,\mathrm{met5}}(t_{\mathrm{s}}^1) &= \eta_{\mathrm{ground}}, \\
x_{\mathrm{pelvis}}^1(t_{\mathrm{s}}^1) &= \eta_{\mathrm{pelvis,x}}^{\mathrm{s}}, \\
y_{\mathrm{pelvis}}^1(t_{\mathrm{s}}^1) &= \eta_{\mathrm{pelvis,y}}^{\mathrm{s}}.
\end{aligned}$$

All remaining generalized positions are left free for optimization of the initial posture of the CP walker model. Furthermore, we ensure that the velocities in the right foot are set to 0 and that the vertical ground reaction forces at the right foot are positive with

$$\begin{aligned}
\dot{\rho}_{\mathrm{foot_r,x}}^{1,\mathrm{hal}}(t_{\mathrm{s}}^1) &= 0, \\
\dot{\rho}_{\mathrm{foot_r,y}}^{1,\mathrm{hal}}(t_{\mathrm{s}}^1) &= 0, \\
\dot{\rho}_{\mathrm{foot_r,z}}^{1,\mathrm{hal}}(t_{\mathrm{s}}^1) &= 0, \\
\dot{\rho}_{\mathrm{foot_r,x}}^{1,\mathrm{met5}}(t_{\mathrm{s}}^1) &= 0, \\
\dot{\rho}_{\mathrm{foot_r,z}}^{1,\mathrm{met5}}(t_{\mathrm{s}}^1) &= 0, \\
v_{\mathrm{foot_r,z}}^{1,\mathrm{hal}}(t_{\mathrm{s}}^1) &\geq 0.
\end{aligned}$$

It is sufficient to enforce the initial conditions to hold at the beginning of the first model stage incorporated in (6.18f) because of the transition conditions (6.18c) and (6.18d).

#### Constraints at Phase 1: Single Support Right Foot

During the first single support phase $t \in \mathcal{T}^1$ of the right foot in model stage 1, we have to ensure that the left foot does not penetrate the ground and that both ankles stay above ground

$$\begin{aligned}
\rho_{\mathrm{foot_r,z}}^{1,\mathrm{hal}}(t) &\geq \eta_{\mathrm{ground}}, \\
\rho_{\mathrm{foot_r,z}}^{1,\mathrm{met5}}(t) &\geq \eta_{\mathrm{ground}}, \\
\rho_{\mathrm{ankle_r,z}}^{1}(t) &\geq \eta_{\mathrm{ground}}, \\
\rho_{\mathrm{ankle_l,z}}^{1}(t) &\geq \eta_{\mathrm{ground}},
\end{aligned}$$

Furthermore a positive vertical ground reaction force at the right contact point has to be fulfilled such that

$$v_{\mathrm{foot\_r,z}}^{1,\mathrm{hal}}(t) \geq 0,$$

holds at the first single support phase. Later in the multi-stage OCP formulation in section 6.4 avoidance of self-penetration is achieved by (6.14) as described in subsection 6.2.5.

#### Constraints at First Transition

At first transition time $t_{\mathrm{f}}^1 = 1$ right before the collision of the left foot takes place, the following conditions are posed:

$$\begin{aligned}
&\rho^{1,\mathrm{hal}}_{\mathrm{foot_l,z}}(t_{\mathrm{f}}^1) = \eta_{\mathrm{ground}},\\
&\rho^{1,\mathrm{met5}}_{\mathrm{foot_l,z}}(t_{\mathrm{f}}^1) = \eta_{\mathrm{ground}},\\
&\dot{\rho}^{1,\mathrm{hal}}_{\mathrm{foot_l,z}}(t_{\mathrm{f}}^1) \leq 0,\\
&\dot{\rho}^{1,\mathrm{met5}}_{\mathrm{foot_l,z}}(t_{\mathrm{f}}^1) \leq 0,\\
&v^{1,\mathrm{hal}}_{\mathrm{foot_l,z}}(t_{\mathrm{f}}^1) \geq 0,
\end{aligned}$$

where the left foot touches the ground with a negative velocity, which ensures that the foot enters the ground from above.

#### Constraints at Phase 2: Single Support Left Foot

Similar to the first phase during the second phase $t \in \mathcal{T}^2$ the right foot should swing above ground and the following constraints should hold:

$$\begin{aligned}
&\rho^{2,\mathrm{hal}}_{\mathrm{foot_l,z}}(t) \geq \eta_{\mathrm{ground}},\\
&\rho^{2,\mathrm{met5}}_{\mathrm{foot_l,z}}(t) \geq \eta_{\mathrm{ground}},\\
&\rho^{2}_{\mathrm{ankle_r,z}}(t) \geq \eta_{\mathrm{ground}},\\
&\rho^{2}_{\mathrm{ankle_l,z}}(t) \geq \eta_{\mathrm{ground}},\\
&v^{2,\mathrm{hal}}_{\mathrm{foot_l,z}}(t) \geq 0.
\end{aligned}$$

Similar to the first single support phase, in the second phase in the multi-stage OCP formulation in section 6.4 avoidance of self-penetration is achieved by (6.14) as described in subsection 6.2.5.

#### Constraints at Second Transition

At second transition time $t_{\mathrm{f}}^2 = 1$ right before collision the following conditions are posed:

$$\rho^{2,\mathrm{hal}}_{\mathrm{foot\_r,z}}(t_{\mathrm{f}}^2) = \eta_{\mathrm{ground}}, \tag{6.31a}$$
$$\rho^{2,\mathrm{met5}}_{\mathrm{foot\_r,z}}(t_{\mathrm{f}}^2) = \eta_{\mathrm{ground}}, \tag{6.31b}$$
$$\dot{\rho}^{2,\mathrm{hal}}_{\mathrm{foot\_r,z}}(t_{\mathrm{f}}^2) \leq 0, \tag{6.31c}$$
$$\dot{\rho}^{2,\mathrm{met5}}_{\mathrm{foot\_r,z}}(t_{\mathrm{f}}^2) \leq 0, \tag{6.31d}$$
$$v^{2,\mathrm{hal}}_{\mathrm{foot\_r,z}}(t_{\mathrm{f}}^2) \geq 0, \tag{6.31e}$$

which are similar to the ones at first transition time.

#### Terminal Condition, Periodicity Constraints and Bounds on Differential States and Controls

The multi-stage OCP formulation for reconstruction with a least-squares objective described in this section fits the gait model to measurement data. Because of the given data a fixed terminal condition can be set by

$$x^2_{\text{pelvis}}(t^2_{\text{f}}) = \eta^{\text{f}}_{\text{pelvis,x}}, \tag{6.32}$$

to some measured value $\eta^{\text{f}}_{\text{pelvis,x}}$. Furthermore, for the gait model of a CP walker multibody system we want periodicity constraints only on selected generalized coordinates and velocities to hold, such that redundancies in the constraints are avoided:

$$y^1_{\text{pelvis}}(t^1_{\text{s}}) = y^2_{\text{pelvis}}(t^2_{\text{f}}), \tag{6.33a}$$
$$z^1_{\text{pelvis}}(t^1_{\text{s}}) = z^2_{\text{pelvis}}(t^2_{\text{f}}), \tag{6.33b}$$
$$\boldsymbol{\varphi}^1_{\text{pelvis}}(t^1_{\text{s}}) = \boldsymbol{\varphi}^2_{\text{pelvis}}(t^2_{\text{f}}), \tag{6.33c}$$
$$\dot{x}^1_{\text{pelvis}}(t^1_{\text{s}}) = \dot{x}^2_{\text{pelvis}}(t^2_{\text{f}}), \tag{6.33d}$$
$$\boldsymbol{\varphi}^1_{\text{i}}(t^1_{\text{s}}) = \boldsymbol{\varphi}^2_{\text{i}}(t^2_{\text{f}}), \qquad \text{i} \in \{\text{hip\_r}, \text{hip\_l}, \text{knee\_r}, \text{knee\_l}, \text{ankle\_r}, \text{ankle\_l}\}, \tag{6.33e}$$
$$\dot{\boldsymbol{\varphi}}^1_{\text{i}}(t^1_{\text{s}}) = \dot{\boldsymbol{\varphi}}^{\boldsymbol{2}}_{\text{i}}(t^2_{\text{f}}), \qquad \text{i} \in \{\text{hip\_r}, \text{hip\_l}, \text{knee\_r}, \text{knee\_l}, \text{ankle\_r}, \text{ankle\_l}\}. \tag{6.33f}$$

Together with the conditions (6.31) at the second phase transition between the end of model stage 2 and the beginning of model stage 1 we include the constraints (6.32) and (6.33) in the second transition condition (6.18d) in the OCP formulation.

Considering given measurements in OCP (6.18) allows to use the following bounds in (6.18h) for the differential states

$$\begin{pmatrix} -10 \\ -10 \\ -10 \end{pmatrix} \leq \boldsymbol{r}^{\text{j}}_{\text{pelvis}} \leq \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix},$$
$$\begin{pmatrix} -0.5 \\ -0.25 \\ -0.5 \end{pmatrix} \leq \boldsymbol{\varphi}^{\text{j}}_{\text{pelvis}} \leq \begin{pmatrix} 0.3 \\ 0.3 \\ 0.4 \end{pmatrix},$$
$$\begin{pmatrix} -1.4 \\ -0.5 \\ -0.7 \end{pmatrix} \leq \boldsymbol{\varphi}^{\text{j}}_{\text{i}} \leq \begin{pmatrix} 0.75 \\ 0.5 \\ 0.7 \end{pmatrix}, \qquad \text{i} \in \{\text{hip\_r}, \text{hip\_l}\},$$
$$\begin{pmatrix} -0.01 \end{pmatrix} \leq \boldsymbol{\varphi}^{\text{j}}_{\text{i}} \leq \begin{pmatrix} 1.9 \end{pmatrix}, \qquad \text{i} \in \{\text{knee\_r}, \text{knee\_l}\},$$
$$\begin{pmatrix} -2.7 \\ -1.2 \\ -0.8 \end{pmatrix} \leq \boldsymbol{\varphi}^{\text{j}}_{\text{i}} \leq \begin{pmatrix} 0.45 \\ 1.2 \\ 0.8 \end{pmatrix}, \qquad \text{i} \in \{\text{ankle\_r}, \text{ankle\_l}\},$$
$$-50 \leq \dot{q}_{\text{ji}} \leq 50, \qquad \text{i} \in \{0, \ldots, \text{n}_{\text{dofs}} - 1\},$$
$$-500 \leq \boldsymbol{\tau}^{\text{a}}_{\text{ji}} \leq 500, \qquad \text{i} \in \{0, \ldots, \text{n}_{\text{act}} - 1\},$$

which are estimated using the *digital twin* described in subsection 6.2.7 and can be chosen in such a way that they are not reached in the solution. In the same way, extended bounds (6.18i) for the controls are chosen: lower bounds $\underline{\boldsymbol{u}_{\text{ji}}} = -10^4$ and upper bounds $\overline{\boldsymbol{u}_{\text{ji}}} = 10^4$ were sufficient for $\text{i} \in \{0, \ldots, \text{n}_{\text{act}} - 1\}$.

In addition to these loose bounds, we formulate tighter lower bounds on $\boldsymbol{\varphi}^{\text{j}}_{\text{i}}, \text{i} \in \{\text{knee\_r}, \text{knee\_l}\}$ at initial and transition times to avoid *"free energy"* contributions from the passive reset forces. In our calculations for $\boldsymbol{\varphi}^{\text{j}}_{\text{i}} \geq \underline{\beta}_{\text{i}} + \epsilon_{\text{pass}}$ we set $\epsilon_{\text{pass}} = 0.1$ following given motion capture data for the dynamics reconstruction.

## 6.4 A Multi-Stage OCP for the Gait of a Patient with CP

To formulate an OCP for the gait of the CP walker multibody system from section 6.1 with its dynamics from section 6.2, we consider the general multi-stage OCP formulation (3.8) from subsection 3.1.2 describing the human gait. With all in section 6.3 introduced quantities and functions such as differential states (6.16) and controls (6.17), the constraints in subsection 6.3.3 together with the dynamics (6.21), transition conditions (6.22), and the objective function (6.36) defined in the following in subsection 6.4.1, the multi-stage OCP is set up completely to describe two synthesized steps of the rigid multibody system model for a CP patient and reads as follows:

$$\min_{\substack{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{u}_1,\boldsymbol{u}_2,\\ d_1,d_2}} \quad \sum_{\mathrm{j}=1}^{2} \boldsymbol{\Phi}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),d_{\mathrm{j}},\boldsymbol{\alpha},\boldsymbol{p}) \tag{6.35a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_{\mathrm{j}}(t) = d_{\mathrm{j}} \cdot \boldsymbol{f}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.35b}$$

$$\boldsymbol{x}_2(t_{\mathrm{s}}^2) = \Delta_1(\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{p}), \tag{6.35c}$$

$$\boldsymbol{x}_1(t_{\mathrm{s}}^1) = \Delta_2(\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.35d}$$

$$0 \le \boldsymbol{c}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{p}), \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.35e}$$

$$0 = \boldsymbol{r}^{\mathrm{ec}}(\boldsymbol{x}_1(t_{\mathrm{s}}^1),\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.35f}$$

$$0 \le \boldsymbol{r}^{\mathrm{ic}}(\boldsymbol{x}_1(t_{\mathrm{f}}^1),\boldsymbol{x}_2(t_{\mathrm{f}}^2),\boldsymbol{p}), \tag{6.35g}$$

$$\underline{\boldsymbol{x}_{\mathrm{j}}} \le \boldsymbol{x}_{\mathrm{j}}(t) \le \overline{\boldsymbol{x}_{\mathrm{j}}}, \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.35h}$$

$$\underline{\boldsymbol{u}_{\mathrm{j}}} \le \boldsymbol{u}_{\mathrm{j}}(t) \le \overline{\boldsymbol{u}_{\mathrm{j}}}, \qquad t \in \mathcal{T}^{\mathrm{j}},\ \mathrm{j}=1,2, \tag{6.35i}$$

$$0 \le d_{\mathrm{j}}, \qquad \mathrm{j}=1,2. \tag{6.35j}$$

It is defined on fixed and *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [t_{\mathrm{s}}^{\mathrm{j}}, t_{\mathrm{f}}^{\mathrm{j}}] = [0,1]$ with stage duration parameters $d_{\mathrm{j}}$ for each model stage $\mathrm{j}=1,2$ and stage transition times before and after collision, $t_{\mathrm{f}}^1 = t_{\mathrm{f}}^2 = 1$ and $t_{\mathrm{s}}^2 = t_{\mathrm{s}}^1 = 0$, respectively, as introduced in subsection 2.1.2 together with the remaining variables and functions. In contrast to the least-squares OCP (6.18) the duration parameters $d_{\mathrm{j}}$ are left free for optimization, whereas all arising model parameters $\boldsymbol{p}$ defined in subsection 6.2.4 are fixed to the solution of the reconstruction. The most quantities and functions that occur in OCP (6.35) are the same as defined in the previous section 6.3. In the synthesis of varying gaits for the rigid multibody system model of a CP patient the main differences to the previous section, where a given motion is reconstructed, is the objective function as defined in more detail in subsection 6.4.1. Furthermore, the constraints as described in subsection 6.3.3 have to be extended by self-penetration constraints (6.14) as described in subsection 6.2.5 to avoid penetration of the segments. This is needed because the deviation to given motion capture data is no longer minimized. In fact, here we are interested to compare various gaits, which are synthesized under consideration of differently weighted optimization criteria.

### 6.4.1 Objective Function

As introduced at the beginning of subsection 3.1.2 we interpret the human gait as optimal with respect to a well chosen combination of different optimization criteria. In general, the objective function is of Bolza-type and represented by

$$\begin{aligned}\sum_{\mathrm{j}=1}^{2} \boldsymbol{\Phi}_{\mathrm{j}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),d_{\mathrm{j}},\boldsymbol{\alpha},\boldsymbol{p}) &= \sum_{\mathrm{k}=1}^{\mathrm{n_M}} \alpha_{\mathrm{k}} \cdot \sum_{\mathrm{j}=1}^{2} \boldsymbol{\phi}_{\mathrm{jk}}^{\mathrm{M}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{f}}^{\mathrm{j}}),d_{\mathrm{j}},\boldsymbol{p}) \\ &\quad + \sum_{\mathrm{k}=\mathrm{n_M}+1}^{\mathrm{n_M}+\mathrm{n_L}} \alpha_{\mathrm{k}} \cdot \sum_{\mathrm{j}=1}^{2} \int_{t_{\mathrm{s}}^{\mathrm{j}}}^{t_{\mathrm{f}}^{\mathrm{j}}} d_{\mathrm{j}} \cdot \boldsymbol{\phi}_{\mathrm{jk}}^{\mathrm{L}}(\boldsymbol{x}_{\mathrm{j}}(t),\boldsymbol{u}_{\mathrm{j}}(t),\boldsymbol{p})\,\mathrm{d}t,\end{aligned} \tag{6.36}$$

with weights $\alpha_k \in \mathbb{R}^{n_M+n_L}$. For the gait of patients with CP we formulate optimization criteria described in the following. If we are interested in meeting given measurements these criteria have to be chosen in such a way that with a selected combination of $n_M + n_L$ criteria and individual weights $\boldsymbol{\alpha} = \begin{pmatrix}\alpha_1 & \dots & \alpha_{n_M+n_L}\end{pmatrix}^T$ the gait of the patient can be reproduced within an acceptable accuracy. The identification of the usually unknown quantities $\boldsymbol{\alpha}$ and $\boldsymbol{p}$ can be achieved by solving a Bilevel Inverse OCP of type (2.21) from section 2.4 based on given motion capture data.

The $n_L = 4$ individual optimization criteria we choose are of Lagrange-type and defined in the following.

#### Optimization Criterion: Maximize Stability

The first optimization criterion serves as a stability criterion by minimizing the deviation of the $y$-components of the position of the hip joint $\rho^j_{i,y}, i \in \{\text{hip\_r, hip\_r}\}$ of the supported leg and the corresponding contact point $\rho^{j,hal}_{i,y}, i \in \{\text{foot\_r, foot\_r}\}$, respectively, at each model stage $j = 1,2$. It is weighted by the euclidean distance squared of both contact points defined by

$$\boldsymbol{\delta}^{\text{feet,j}}(\boldsymbol{q}_j(t)) = \left\| \boldsymbol{\rho}^{j,hal}_{\text{foot\_r}}(\boldsymbol{q}_j(t)) - \boldsymbol{\rho}^{j,hal}_{\text{foot\_l}}(\boldsymbol{q}_j(t)) \right\|_2^2,$$

which is always positive because of stated constraints for self-penetration avoidance in the OCP formulation of (6.35). This ensures that the contribution of the stability criterion is higher if the swinging leg is close to the supporting leg, and less important near the change of two single support phases. At the first single support phase of the right foot the criterion is formulated as

$$\boldsymbol{\phi}^L_{11}(\boldsymbol{x}_1(t), \boldsymbol{u}_1(t), \boldsymbol{p}) := \frac{1}{\boldsymbol{\delta}^{\text{feet,1}}(\boldsymbol{q}_1(t))} \cdot \left( \rho^{1,hal}_{\text{foot\_r,y}}(\boldsymbol{q}_1(t)) - \rho^1_{\text{hip\_r,y}}(\boldsymbol{q}_1(t)) \right)^2, \tag{6.37}$$

and at second single support phase of the left foot we have

$$\boldsymbol{\phi}^L_{21}(\boldsymbol{x}_2(t), \boldsymbol{u}_2(t), \boldsymbol{p}) := \frac{1}{\boldsymbol{\delta}^{\text{feet,2}}(\boldsymbol{q}_2(t))} \cdot \left( \rho^{2,hal}_{\text{foot\_l,y}}(\boldsymbol{q}_2(t)) - \rho^2_{\text{hip\_l,y}}(\boldsymbol{q}_2(t)) \right)^2. \tag{6.38}$$

The stability criterion stated here differs from other criteria found, e.g. in [87, 128] and references therein. Some of them rely on the so-called *zero moment point* or the *angular momentum* over the CoM. However, in our dynamic rigid multibody system model for a CP patient only two contact points per foot are chosen and the upper body joints are fixed, such that we use the stability criterion proposed by Hatz [80] and modify it by a weighted factor to account for the varying contribution during walking.

#### Optimization Criterion: Minimize Absolute Mechanical Work

The second criterion minimizes the sum over the component-wise and weighted absolute mechanical works, which must be generated at each actuated joint around the corresponding axes. It is defined by

$$\boldsymbol{\phi}^L_{j2}(\boldsymbol{x}_j(t), \boldsymbol{u}_j(t), \boldsymbol{p}) := \frac{1}{\boldsymbol{\delta}^{\text{step,j}}} \left\| \boldsymbol{W}^{\boldsymbol{\tau}} \boldsymbol{\tau}^a_j(t) \circ \dot{\boldsymbol{q}}^{\text{act}}_j(t) \right\|_1 = \frac{1}{\boldsymbol{\delta}^{\text{step,j}}} \sum_{k=0}^{n_{act}-1} \left| w^\tau_k \tau^a_{jk}(t) \dot{q}_{jk+6}(t) \right|, \tag{6.39}$$

where $\dot{\boldsymbol{q}}^{\text{act}}_j(t)$ is a subvector of $\dot{\boldsymbol{q}}_j(t)$ with entries corresponding to the actuated joints and $\circ$ denotes the element-wise vector multiplication. Furthermore, $\boldsymbol{\delta}^{\text{step,j}}$ is the step length at the corresponding model stage $j = 1,2$, and $\boldsymbol{W}^{\boldsymbol{\tau}} \in \mathbb{R}^{n_{act} \times n_{act}}$ is a diagonal matrix with entries $w^\tau_k \in \mathbb{R}$ for $k = 0, \dots, n_{act} - 1$. These weights $w^\tau_k > 0$ account for the different order of magnitude of the active torques acting at the joints. In the actual implementation we do not use the $L_1$ norm, but a smooth approximation given by

$$\|\boldsymbol{v}\|_1 \approx \sum_k \sqrt{v_k^2 + \epsilon_{\text{smooth}}},$$

where the vector is set to $\boldsymbol{v} := \boldsymbol{W}^{\boldsymbol{\tau}} \boldsymbol{\tau}_{\mathrm{j}}^{\mathrm{a}}(t) \circ \dot{\boldsymbol{q}}_{\mathrm{j}}^{\mathrm{act}}(t)$ and with $\epsilon_{\mathrm{smooth}} > 0$ sufficiently small to ensure differentiability.

#### Optimization Criterion: Minimize Squared Torques

As a criterion for the required effort to perform two steps we choose a weighted sum over the squared active torques. We define the third Lagrange-term by

$$\boldsymbol{\phi}_{\mathrm{j3}}^{\mathrm{L}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), \boldsymbol{p}) := \frac{1}{\boldsymbol{\delta}^{\mathrm{step,j}}} \cdot \| \boldsymbol{W}^{\boldsymbol{\tau}} \boldsymbol{\tau}_{\mathrm{j}}^{\mathrm{a}}(t) \|_2^2, \tag{6.40}$$

where $\boldsymbol{\delta}^{\mathrm{step,j}}$ and $\boldsymbol{W}^{\boldsymbol{\tau}} \in \mathbb{R}^{\mathrm{n_{act}} \times \mathrm{n_{act}}}$ are the same as above.

#### Optimization Criterion: Minimize Squared Torque Derivatives

Furthermore, the last criterion minimizes a weighted sum over the squared controls described by

$$\boldsymbol{\phi}_{\mathrm{j4}}^{\mathrm{L}}(\boldsymbol{x}_{\mathrm{j}}(t), \boldsymbol{u}_{\mathrm{j}}(t), \boldsymbol{p}) := \frac{1}{\boldsymbol{\delta}^{\mathrm{step,j}}} \| \boldsymbol{W}^{\dot{\boldsymbol{\tau}}} \cdot \boldsymbol{u}_{\mathrm{j}}(t) \|_2^2 = \frac{1}{\boldsymbol{\delta}^{\mathrm{step,j}}} \| \boldsymbol{W}^{\dot{\boldsymbol{\tau}}} \cdot \dot{\boldsymbol{\tau}}_{\mathrm{j}}^{\mathrm{a}}(t) \|_2^2, \tag{6.41}$$

which is divided by the step length at model stages j = 1,2, and weighted by a diagonal matrix $\boldsymbol{W}^{\dot{\boldsymbol{\tau}}} \in \mathbb{R}^{\mathrm{n_{act}} \times \mathrm{n_{act}}}$, where each entry $w_{\mathrm{k}}^{\dot{\tau}} \in \mathbb{R}$ for $\mathrm{k} = 0, \ldots, \mathrm{n_{act}} - 1$ accounts for the different order of magnitude.

In chapter 10 the discussion on modeling of CP patients' gait is continued. Therein, the numerical results of the reconstruction of the dynamics of a patient with CP to motion capture data provided by the HEIDELBERG MOTIONLAB [173] are given by solving the least-squares multi-stage OCP (6.18) from the previous section 6.3. Furthermore, varying synthesized gaits of a patient with CP are analyzed under consideration of the multi-stage OCP (6.35) from this section with differently weighted optimization criteria.

## 6.5 Comparison to CP Gait Model by Hatz [80] and other Existing Models

In this work we decided to solve the equations of motion with RBDL by Felis [53] with an interface to LUAMODELs, as described in section 6.2. We implemented the previously introduced multibody system model of a CP patient from section 6.1 as a LUAMODEL. Originally, the CP model by Hatz was built using the proprietary software package HUMANS [1] for the generation of the corresponding equations of motion. For the most part, the structure of our multibody systems model's segments and their connections through joints are the same as in the CP model by Hatz. The main differences in the rigid multibody system model, apart from the implementation framework, are the patient-specific knee axes determined under consideration of motion capture data as described in subsection 6.2.8. This patient-specific knee axes enable us to incorporate only one DOF around the particular axis in each knee joint, to cope with the degenerated *"physical"* joints of the CP patient which result in the characteristic *crouched gait*. Three DOFs in each knee joint as in the CP model by Hatz might lead to numerical instability when solving OCPs describing the gait as in section 6.4. In other rigid multibody system models, such as the HEIMAN model by Felis, the three dimensional gait with the additional asymmetrical stature of the patient cannot be captured appropriately. Furthermore, in the dynamic model of section 6.2 the contact model differs to the one in [80]. On the one side, as in the thesis of Hatz [80], we also incorporated the need of an forefoot contact to cope with the typically appearing so-called *club foot* or *pes equinus* gait of CP patients, where the heel never touches the ground. However, on the other side, instead of only one contact point at the tip of each foot, we include two contact points at hallux and metatarsal 5 positions. Furthermore, we enforce a positive reaction force during single support phase at the corresponding foot at hallux point. There are other, more detailed foot contact models available in the literature, e.g. finite-element approximations of continuum mechanics models, as in Halloran et al. [78], surrogate models like (3D-) volumetric contact models, as in Brown and McPhee [28], and rigid foot-ground contact models, as in Felis and Mombaur [54], Ren et al. [144]. We decided to stick with the simpler contact model because of our ultimate goal to use the multi-

stage OCP formulation as lower level in the Bilevel Inverse OCP and the need to reduce the complexity of our gait model to an acceptable level. Beside the contact model in [80], a similar model has proven its successful integration in an optimization-based approach for predicting sprinting and long jump motions, where also a forefoot contact takes place. It has been published recently in the thesis of Emonds [49]. In our multi-stage OCP formulation to describe the gait of a patient with CP, we propose a well-chosen combination of optimization criteria under consideration of results by [12, 49, 52, 80] and use active torques derivatives as controls to obtain more smooth human-like motions, as e.g. in [49]. Other than in the OCP gait model of Hatz, where only active torques are considered, we distinguish between active and passive torque contributions in the joint actuation. In the latter torque contribution, damping terms are incorporated in the actuated joints to reduce vibrations, as e.g. in [79]. Furthermore, to be able to capture the typical *crouched* gait of patients with CP, we add passive reset forces to the passively actuated torques of the knee joints. This modeling approach was proposed in the work of Schlöder [150] to model treatment planning for patients with CP and applied in a case study to a basic 2−D model with 7 DOFs. In the numerical results of chapter 10, the limited range of motion of a patient with CP can be reconstructed to given real world data. Hence, in future research our more complex personalized CP gait model should also be investigated for application of model based treatment planning. Before this can be performed model parameters and objective weights in the multi-stage OCP (6.35) have to be identified by solving a Bilevel Inverse OCP, where a PE Problem is constraint by the derived OCP. An interesting extention of our proposed CP gait model would be to parametrize the individual knee axes. Then the arising additional model parameters could be identfied by solving a corresponding Bilevel Inverse OCP as the one stated in the following section 6.6. As a result, this provides a calibrated patient-specific gait model that may be used in model based treatment planning additionally covering interventions related to changes in the rotational range of knee joints.

## 6.6 Outlook: Bilevel Inverse OCP for Identification of Unknowns in CP Gait

In this section we formulate a Bilevel Inverse OCP of the form (2.21) for the identification of individual weights $\boldsymbol{\alpha}$ and model parameters $\boldsymbol{p}$ in the underlying multi-stage OCP for the gait of the rigid multibody system model of a CP patient described in the previous sections. On the upper level we have a PE problem with given measurements constrained by the lower level OCP (6.35) of section 6.4, and, furthermore, conditions on the unknown objective weights and model parameters. At this point we assume, that all model parameters are structurally and practically identifiable for the available measurement data, see subsection 6.2.4. If this is not the case in practice, a suitable subvector of the model parameter vector can be identified via the Bilevel Inverse OCP, and the remaining parameters might be determind directly from given measurements. For this set-up, where we have measurements $\boldsymbol{\eta}^{\mathrm{P}}$ processed as described in subsection 6.2.6 and subsection 6.2.7, and a least-squares objective (6.20) on the upper level as defined in subsection 6.3.1 we can formulate the following Bilevel Inverse OCP

$$\min_{\substack{\boldsymbol{\alpha},\boldsymbol{p},\\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \frac{1}{2}\sum_{\mathrm{j}=1}^{2}\sum_{\mathrm{n}=0}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1}\sum_{\mathrm{k}=0}^{\mathrm{n}_{\mathrm{h}}^{\mathrm{j}}-1}\frac{\left(h_{\mathrm{jk}}(\boldsymbol{x}_{\mathrm{j}}(t_{\mathrm{jn}}^{\mathrm{m}}),\boldsymbol{p})-\eta_{\mathrm{jnk}}^{\mathrm{P}}\right)^2}{\sigma_{\mathrm{jnk}}^2}+\gamma^{\mathrm{R}}\sum_{\mathrm{j}=1}^{2}\int_{t_{\mathrm{s}}^{\mathrm{j}}}^{t_{\mathrm{f}}^{\mathrm{j}}}\left(\boldsymbol{W}^{\dot{\boldsymbol{t}}}\|\boldsymbol{u}_{\mathrm{j}}(t)\|_2^2\right)\mathrm{d}t \tag{6.42a}$$

$$\text{s.t.} \quad (\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}) \text{ solve OCP (6.35)}, \tag{6.42b}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{n}_{\mathrm{M}}+\mathrm{n}_{\mathrm{L}}}\alpha_{\mathrm{k}}=1,\ \boldsymbol{\alpha}\geq 0, \tag{6.42c}$$

$$\underline{\boldsymbol{p}}\leq\boldsymbol{p}\leq\overline{\boldsymbol{p}}. \tag{6.42d}$$

All other arising quantities and functions in (6.42) are already introduced and defined in the previous sections. Self-penetration avoidance can be neglected in the constraints of the lower level OCP, because of the fitting to given measurements. Our DISIMFAS from chapter 4 to solve these kinds of problems was implemented

in the software package ParDynOpt as described in chapter 7. However, so far the actual implementation of RBDL [53] is not compatible to ADOL-C [171], which is used in the SolvIND [108] interface for derivative generation. It is left for future research to investigate the proposed DiSimFAS from chapter 4 with regard to the patient-specific Bilevel Inverse OCP stated above. Unfortunately, so far for the Direct All-at-Once Approach in ParaOCP [80] already for the Bilevel Inverse OCP of the basic walker model described in chapter 5 no convergence could be achieved because of arising infeasible quadratic programs within the solution procedure. Also a replacement of the underlying SQP method by an interior-point approach with selected regularization and penalization approaches, such as Scholtes-regularization [153] or lifting approach [80], as described in section 1.3 could not tackle the infeasibility problems arising from the resulting MPCCs. With the omission of the arising inequality constraints in the resulting NLP by applying the DiSimFAS and, hence, a reduction of the complexity of the problem, we are optimistic that we can achieve better results for Bilevel Inverse OCP (6.42) with ParDynOpt after adaptation of RBDL in the future. In the case study in chapter 9 of the basic walker model described in chapter 5 the DiSimFAS was successfully applied on a basic human-like locomotion.

## 6.7 Pilot Study: Identification of Optimal Weights by Deep Neural Networks

In this section we follow another approach for the identification of optimal weights in the previously derived CP gait model. In contrast to the optimization based mathematical methods considered so far, in this study we use Deep Neural Networks (DNNs). The approach using DNNs is based on the work of Afkham et al. [4] and the references therein. In the next sections we follow the description in [4] and refer to Schmidhuber [152] for a comprehensive overview on DNNs.

### 6.7.1 Basic Concept and Motivation

In contrast to chapter 4, where the Bilevel Inverse OCP (6.42) is solved by applying the DiSimFAS, in this section we consider a so-called *supervised learning approach*. In Russell and Norvig [148] the following definition for the latter approach is given: "In supervised learning the agent observes some example input-output pairs and learns a function that maps from input to output." This approach might also lead to bilevel optimization problems, as in the work of Haber and Tenorio [77]. However, in the supervised learning approach described here, which is based on [4], this is not the case. In a first step a DNN is trained with simulated measurements by solving a set of lower level OCPs describing the CP gait for differently weighted optimization criteria. Therein, a mapping from simulated measurements to corresponding objective weights is approximated. If the DNN is trained appropriately, in a second step optimal weights for newly obtained measurements can be computed by forward propagation through the network.

This approach does not fall into the categories of the previously introduced solution methods for Bilevel Inverse OCPs described in subsection 2.4.2, where the problem at hand is solved numerically with indirect or direct methods. The mathematical challenges, which are accompanied by the approaches for the solution of Bilevel Inverse OCPs considered so far, on the one hand, and the possibility of providing a high amount of simulated data by solving OCPs for differently weighted objective functions, on the other hand, motivated us to consider machine learning approaches, which have gained increasing popularity in recent years, and, in particular, the supervised learning approach by Afkham et al. [4]. In the following section we develop a learning approach for optimal weights for the CP gait model described previously via DNNs based on the latter approach as a first study in this direction. This section 6.7 is then concluded with a case study in section 10.4. Note that in the following we partially use different notations as in the previous sections.

### 6.7.2 Learning weights via DNNs

We start with a measurement model that comprises measurements $\boldsymbol{\eta} \in \mathbb{R}^{n_{\boldsymbol{\eta}}}$ observed at specific time points as described in subsection 6.3.1, such that for true model parameters $\boldsymbol{p}^*$ we have

$$\eta_{jnk} = h_{jk}(\boldsymbol{x}_j(t_{jn}^m), \boldsymbol{p}^*) + \varepsilon_{jnk}, \qquad j = 1,2,\ n = 0,\dots,n_m^j - 1,\ k = 0,\dots,n_h^j - 1, \tag{6.43}$$

where the indices j correspond to model stages, n to the time points, and k to entries of the measurement functions. In the following without loss of generality, we consider the model response defined as

$$h_{jk}(\boldsymbol{x}_j(t_{jn}^m), \boldsymbol{p}^*) := q_{jk}(t_{jn}^m), \qquad \text{for } k = 0,\dots,n_{dofs} - 1, \tag{6.44}$$

where the corresponding generalized coordinates are evaluated at time points $t_{jn}^m$ and combined into vector $\boldsymbol{q} \in \mathbb{R}^{n_{\boldsymbol{\eta}}}$. Note that the resulting measurements are not the same as described in subsection 6.2.7 by creating a digital twin. In contrast to the Bilevel Inverse OCP formulation (6.42) of the last section, in the supervised learning approach described here, we only seek for unknown objective weights $\boldsymbol{\alpha}$ and assume that all arising model parameters $\boldsymbol{p}$ are already known.

In [4] the regularization parameters of inverse problems are identified. These parameters are part of the objective function minimizing a *model-data misfit* enhanced by a regularization term. In our case the distinctive feature to the approach in [4] is that objective weights, that are part of the OCP (6.35) describing the CP gait need to be identified. This means that the considered generalized coordinates in (6.44) correspond to the solution of OCP (6.35) for individual weights $\boldsymbol{\alpha}$, that need to be determined, such that given measurements (6.43) are met.

In the following considerations we suppose that the solution of the OCP (6.35) is unique for given objective weights, and avoid redundancies by ensuring that $\sum_{k=1}^{4} \alpha_k = 1, \boldsymbol{\alpha} \geq 0$. Furthermore, we assume that there exists a continuous function $\boldsymbol{\Phi}_{DNN} : \mathbb{R}^{n_{\boldsymbol{\eta}}} \to \mathbb{R}^{n_{\boldsymbol{\alpha}}}$ that maps measurements $\boldsymbol{\eta}$ comprising observed generalized coordinates at specific time points onto objective weights $\boldsymbol{\alpha} \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$,

$$\boldsymbol{\Phi}_{DNN}(\boldsymbol{\eta}) = \boldsymbol{\alpha}. \tag{6.45}$$

The nonlinear function $\boldsymbol{\Phi}_{DNN}$ is supposed to be well-defined. The basic idea in the supervised machine learning approach, is to approximate this function $\boldsymbol{\Phi}_{DNN}$ by a DNN denoted by $\hat{\boldsymbol{\Phi}}_{DNN}$ and use the estimated DNN for identification of objective weights for newly obtained measurements.

**Algorithm 4** Learning objective weights via DNNs

*Offline Phase*

1: Require CP gait model described by OCP (6.35) with known model parameters
2: Solve $n_S$ OCPs of the form (6.35) for varied $\boldsymbol{\alpha}_{(s)}^*, s = 1,\dots,n_S$
3: Obtain optimal generalized coordinates $\boldsymbol{q}_{(s)}^*, s = 1,\dots,n_S$ as training signals
4: Set-up DNN $\hat{\boldsymbol{\Phi}}_{DNN} = \hat{\boldsymbol{\Phi}}_{DNN}(\boldsymbol{\eta}, \boldsymbol{\theta})$
5: Use optimal pairs (6.46) as training data to compute network parameters $\hat{\boldsymbol{\theta}}$

*Online Phase*

6: Obtain new data $\boldsymbol{\eta}$ (e.g. from HEIDELBERG MOTIONLAB)
7: Propagate $\boldsymbol{\eta}$ through trained DNN to get $\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\Phi}}_{DNN}(\boldsymbol{\eta}; \hat{\boldsymbol{\theta}})$
8: Solve OCP (6.35) with $\boldsymbol{\alpha} = \hat{\boldsymbol{\alpha}}$ to obtain optimal differential states and controls

In algorithm 4, an overview of the approach for the identification of optimal weights via DNNs is illustrated. Therein we distinguish between an *offline phase* and an *online phase*. In the offline phase the DNN $\hat{\boldsymbol{\Phi}}_{DNN}$ is

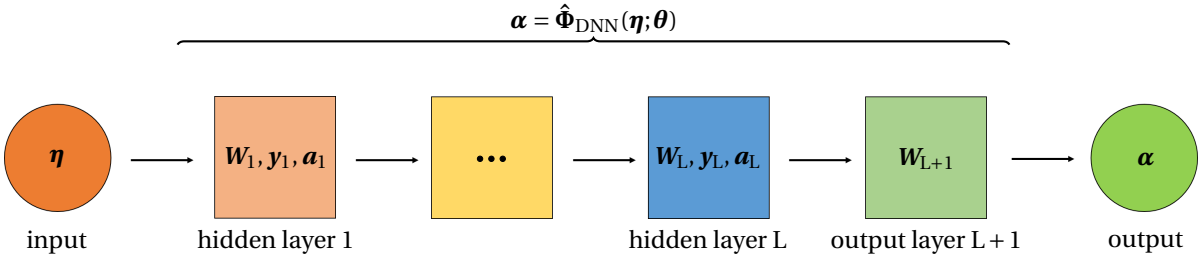


**Figure 6.7:** This figure depicts a schematic representation of a DNN $\hat{\boldsymbol{\Phi}}_{\mathrm{DNN}}(\boldsymbol{\eta};\boldsymbol{\theta})$ with L hidden layers. Each hidden layer $\ell$ is determind by an activation function $\boldsymbol{a}_\ell$, weights $\boldsymbol{W}_\ell$, and biases $\boldsymbol{y}_\ell$. The DNN mapps an input $\boldsymbol{\eta}$ onto an output $\boldsymbol{\alpha}$ defined for given network parameter vector $\boldsymbol{\theta}$, which combines all weights and biases.

trained on $\mathrm{n_S}$ simulated data pairs

$$(\boldsymbol{\eta}_{(\mathrm{s})}, \boldsymbol{\alpha}_{(\mathrm{s})}) = (\boldsymbol{q}^*_{(\mathrm{s})}, \boldsymbol{\alpha}^*_{(\mathrm{s})}), \qquad \mathrm{s} = 1, \ldots, \mathrm{n_S}, \tag{6.46}$$

where $\boldsymbol{q}^*_{(\mathrm{s})}$ are the solutions of OCPs (6.35) for given varying objective weights $\boldsymbol{\alpha}^*_{(\mathrm{s})}$. For training we choose the solutions $\boldsymbol{q}^*_{(\mathrm{s})}$ directly without adding noise. In the offline phase the trained DNN $\hat{\boldsymbol{\Phi}}_{\mathrm{DNN}}$ is used to obtain estimated objective weights $\hat{\boldsymbol{\alpha}}$ for given measurements $\boldsymbol{\eta}$. Finally, optimal differential states and controls can be calculated by solving OCP (6.35) for the estimated weights. In the following we describe the online phase in more detail, where we consider a fully-connected neural network.

With the assumption that there exists a function $\boldsymbol{\Phi}_{\mathrm{DNN}}$ (6.45) that can be approximated by a DNN $\hat{\boldsymbol{\Phi}}_{\mathrm{DNN}}$, we define a so-called *fully-connected feedforward neural network* as $\hat{\boldsymbol{\Phi}}_{\mathrm{DNN}} : \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}}} \times \mathbb{R}^{\mathrm{n}_{\boldsymbol{\theta}}} \to \mathbb{R}^{\mathrm{n}_{\boldsymbol{\alpha}}}$ with

$$\hat{\boldsymbol{\Phi}}_{\mathrm{DNN}}(\boldsymbol{\eta};\boldsymbol{\theta}) = \left(\boldsymbol{\phi}_{\mathrm{n_L}+1}(\boldsymbol{\theta}_{\mathrm{n_L}+1}) \circ \ldots \circ \boldsymbol{\phi}_1(\boldsymbol{\theta}_1)\right)(\boldsymbol{\eta}) = \boldsymbol{\alpha}, \tag{6.47}$$

where $\circ$ denotes the component-wise composition of functions $\boldsymbol{\phi}_\ell : \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}_{\ell-1}}} \times \mathbb{R}^{\mathrm{n}_{\boldsymbol{\theta}_\ell}} \to \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}_\ell}}$ for network layers $\ell = 1, \ldots, \mathrm{n_L} + 1$. The dimension of the input is set to the dimension of the measurement vector $\boldsymbol{\eta}$ with $\mathrm{n}_{\boldsymbol{\eta}_0} = \mathrm{n}_{\boldsymbol{\eta}}$. For the output layer $\ell = \mathrm{L} + 1$ the dimension is $\mathrm{n}_{\boldsymbol{\eta}_{\mathrm{n_L}+1}} = \mathrm{n}_{\boldsymbol{\alpha}}$, which corresponds to the weight $\boldsymbol{\alpha}$ as output. DNN (6.47) describes a parametrized mapping between measurements $\boldsymbol{\eta}$ and objective weights $\boldsymbol{\alpha}$, with network parameter vector $\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}_1^T & \ldots & \boldsymbol{\theta}_{\mathrm{n_L}+1}^T \end{pmatrix}^T \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\theta}}}$ comprising parameters $\boldsymbol{\theta}_\ell \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\theta}_\ell}}$ of each layer $\ell = 1, \ldots, \mathrm{n_L} + 1$. These network parameters $\boldsymbol{\theta}_\ell$ combine entries of so-called network *weights* $\boldsymbol{W}_\ell$ and *biases* $\boldsymbol{y}_\ell$. The parameter vector for layer $\ell$ can be, e.g. defined as $\boldsymbol{\theta}_\ell = \begin{pmatrix} \boldsymbol{y}_\ell^T & \mathrm{Vec}(\boldsymbol{W}_\ell)^T \end{pmatrix}^T$, with $\boldsymbol{y}_\ell \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}_\ell}}$ and $\boldsymbol{W}_\ell \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}_\ell} \times \mathrm{n}_{\boldsymbol{\eta}_{\ell-1}}}$, where $\mathrm{Vec}(\boldsymbol{W}_\ell)$ denotes a vector comprising all entries of the weight matrix. For the so-called *hidden network layers* $\ell = 1, \ldots, \mathrm{n_L}$ the functions $\boldsymbol{\phi}_\ell$ of each layer are defined as

$$\boldsymbol{\phi}_\ell(\boldsymbol{\theta}_\ell)(\boldsymbol{\eta}_{\ell-1}) = \boldsymbol{a}_\ell(\boldsymbol{W}_\ell \boldsymbol{\eta}_{\ell-1} + \boldsymbol{y}_\ell),$$

where $\boldsymbol{a}_\ell : \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}_\ell}} \to \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}_\ell}}$ are typically nonlinear *activation functions*. One prominent example for an activation function is the *rectified linear unit* function $\boldsymbol{ReLU} : \mathbb{R} \to \mathbb{R}$ with $\boldsymbol{ReLU}(x) = \max(0, x)$, such that $\boldsymbol{a}_\ell$ is defined, e.g., as

$$\boldsymbol{a}_\ell(\boldsymbol{x}_\ell) = \begin{pmatrix} \boldsymbol{ReLU}(x_{\ell 1}) & \ldots & \boldsymbol{ReLU}(x_{\ell \mathrm{n}_{\boldsymbol{\eta}_\ell}}) \end{pmatrix}^T \quad \text{for} \quad \ell = 1, \ldots, \mathrm{n_L}, \tag{6.48}$$

where $\boldsymbol{x}_\ell = \boldsymbol{W}_\ell \boldsymbol{\eta}_{\ell-1} + \boldsymbol{y}_\ell$. The last layer $\ell = \mathrm{n_L} + 1$ is called the *output layer* and is supposed to be a linear transformation with no bias term defined by $\boldsymbol{\phi}_{\mathrm{n_L}+1}(\boldsymbol{\theta}_{\mathrm{n_L}+1})(\boldsymbol{\eta}_{\mathrm{n_L}}) = \boldsymbol{W}_{\mathrm{n_L}+1} \boldsymbol{\eta}_{\mathrm{n_L}}$. In the design of the DNN (6.47) several choices can be made. For instance the number of hidden network layers $\mathrm{n_L}$, the width of each hidden layer $\mathrm{n}_{\boldsymbol{\eta}_\ell}$, and the type of activation function (6.48) at each layer $\ell = 1, \ldots, \mathrm{n_L}$ can be varied, e.g., according to universal approximation properties as e.g., in [89, 90].

In the online phase the goal is to determine the network parameters $\boldsymbol{\theta}$ with an estimated parameter set $\hat{\boldsymbol{\theta}}$ providing an accurate approximation of the DNN $\hat{\boldsymbol{\Phi}}_{\mathrm{DNN}}(\boldsymbol{\eta};\hat{\boldsymbol{\theta}}) \approx \boldsymbol{\Phi}_{\mathrm{DNN}}(\boldsymbol{\eta})$. With $\mathrm{n_S}$ simulated data (6.46) the DNN (6.47) is trained as described in the following. The solutions $\boldsymbol{q}^*_{(\mathrm{s})}$ of OCPs (6.35) for given varying objective weights $\boldsymbol{\alpha}^*_{(\mathrm{s})}$ are used as inputs $\boldsymbol{\eta}_{(\mathrm{s})} \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\eta}}}$ in the DNN and the corresponding objective weights $\boldsymbol{\alpha}^*_{(\mathrm{s})}$ as outputs $\boldsymbol{\alpha}_{(\mathrm{s})} \in \mathbb{R}^{\mathrm{n}_{\boldsymbol{\alpha}}}$. Note that because of the assumtion that the model response in the measurement model (6.43) is defined by (6.44) and without consideration of noise on the training data, the solutions $\boldsymbol{q}^*_{(\mathrm{s})}$ can be used directly for approximation of the DNN. Then the network parameters $\hat{\boldsymbol{\theta}}$ can be estimated, e.g., by solving the following optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\arg\min} \qquad \frac{1}{\mathrm{n_S}} \sum_{\mathrm{s}=1}^{\mathrm{n_S}} \| \hat{\boldsymbol{\Phi}}_{\mathrm{DNN}}(\boldsymbol{q}^*_{(\mathrm{s})};\boldsymbol{\theta}) - \boldsymbol{\alpha}^*_{(\mathrm{s})} \|_2^2. \tag{6.49}$$

In (6.49) the sum over all mean squared errors between the parametrized DNN (6.47) evaluated at $\boldsymbol{q}^*_{(\mathrm{s})}$ and the objective weights $\boldsymbol{\alpha}^*_{(\mathrm{s})}$ for all training pairs $\mathrm{s} = 1,\dots,\mathrm{n_S}$ is minimized and the estimated network parameters are set to the solution $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$. Various numerical methods for solving these kinds of optimization problems are available. Among these methods stochastic approximation methods as introduced, e.g., in [145, 159], are commonly used, such as the stochastic gradient descent method and variants like ADAM [101]. The resulting trained DNN can then be used in the offline phase of algorithm 4 to obtain estimated objective weights $\hat{\boldsymbol{\alpha}}$ for newly available measurement data $\boldsymbol{\eta}$. Finally, the OCP (6.35) for the estimated weights can be solved to get optimal differential states and controls. In section 10.4 the investigation of the identification of optimal weights in the CP gait model via the DNN derived in this section is concluded in a case study.

# Part III

# Implementations and Numerical Results

# Chapter 7

# The Software Package ParDynOpt

The software package ParDynOpt was developed and implemented in the course of this thesis with support from Andreas Sommer and Leonard Wirsching and initial contributions by Andreas Meyer, within the Scientific Computing Sustainable Software Collaboratory under the supervision of Ekaterina Kostina at the Interdisciplinary Center for Scientific Computing. In the following a brief overview of its functionality is given.

## 7.1 Introduction and Software Structure

Apart from general OCPs, our software package ParDynOpt allows us to solve Bilevel Inverse OCPs of the form (2.21) efficiently. It realizes the proposed DiSimFAS as described in chapter 4 and provides interfaces to the software package Ipopt [169] for the solution of the resulting structured one-level NLP (4.13) with an efficient interior-point method. Furthermore, interfaces for SNOPT [71] and filterSQP [59] are prepared for implementations of SQP methods. For setting up a dynamic model in a comfortable way, an interface to the software package SolvIND [5] is integrated. SolvIND was developed by Albersmeyer in the group of Hans Georg Bock and provides ODE and DAE solvers in an IND framework for exact derivative generation using AD with an interface to the software package ADOL-C [170]. This powerful feature of the software suite SolvIND is also supported by ParDynOpt.

ParDynOpt is a `C++` software package constructed in a modular way. It implements interfaces to dynamic model descriptions, discretization methods, evaluations of model functions and corresponding derivative generations, and integrates various NLP solvers. It is constructed in such a way, that its modules can easily be exchanged and extended in the future. Furthermore, ParDynOpt also provides the possibility to solve other problems than the ones already implemented in a convenient way. In the future this can be realized by new implementation of the corresponding problem formulation and its translation into the resulting NLP, under consideration of already existing modules such as, e.g., modules for model description, discretization and function evaluation with derivative generation.

In Figure 7.6 the basic structure of ParDynOpt is depicted. In the following we give an overview by referring to the shown class diagram on a conceptual level and describe each module marked with a frame in the figure. A more detailed description how to set up a OCP and a Bilevel Inverse OCP within the user interface is described in section 7.2.

In sum, we give only a brief overview of the structure in ParDynOpt and features implemented therein. For a more detailed insight and better understanding of useful concepts for the realization of the modular structure we refer the reader to the source code.

#### *Problem Description* Module

To set up a problem in ParDynOpt, in a first step an object of the `ProblemDescription` class has to be instantiated by the user. Hence, we start with the *problem description* module in Figure 7.1. Therein, the number of model stages, the chosen discretization methods for the dynamics and the control functions with the possibility of user defined discretization grids per stage can be specified. Furthermore, the `ProblemDescription` class incorporates the dynamic models of all model stages. These dynamic models are incorporated in a derived class of the abstract `Model` class with a chosen model description. Within the implemented `ModelSolvInd` class the model description of the software package SolvIND [5] is integrated. Hence, the dynamic model can be set

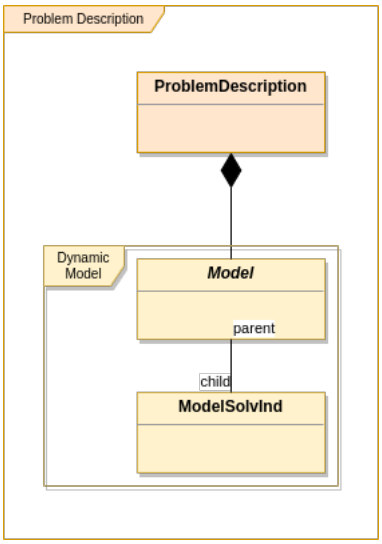


**Figure 7.1:** This figure depicts the PARDYNOPT structure of the *problem description* module on a conceptual level.

up in a very comfortable way. Furthermore, in the `ProblemDescription` class discontinuities can be modeled with a transition stage of duration 0 by defining the corresponding model stage as `StageType::transition`, whereas dynamic model stages are defined by `StageType::dynamic`. Setting `NodeEvaluatorType::solvind` when constructing a `ProblemDescription` object enables SOLVIND's function evaluator and ODE and DAE solvers in an IND framework to evaluate model functions, integrate the initial value problems, and generate exact derivatives using AD with an interface to the software package ADOL-C [170]. Furthermore, the NLP solver of choice can be specified within the `ProblemDescription` class. Setting `NlpSolverType::ipopt` uses the interface to the IPOPT [169] software package with an efficient implementation of an interior-point method. Other interfaces can be provided for SQP methods such as SNOPT [71] or FILTERSQP [59].

### *Problem* Module

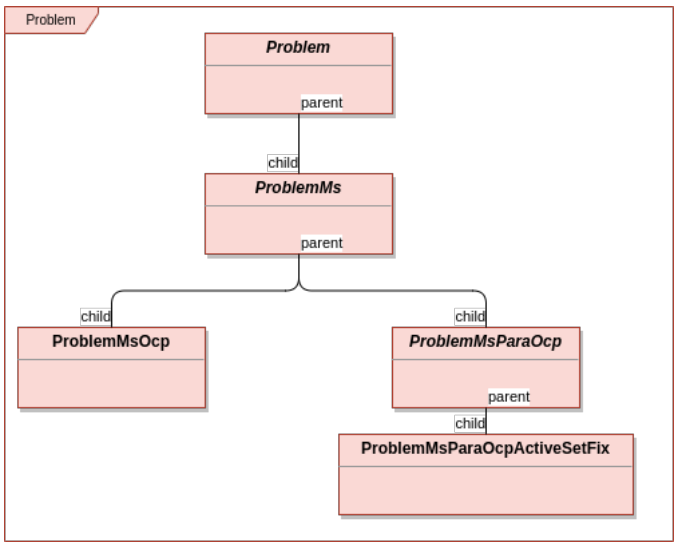


**Figure 7.2:** This figure depicts the PARDYNOPT structure of the *problem* module on a conceptual level.

The main module in PARDYNOPT is described by the abstract `Problem` class and its derived classes in Figure 7.2. Such a special `Problem` object is instantiated by the user for a given `ProblemDescription`. After finalized ini-

tialization it is then solved according to the provided description by calling the method `solve()` of the corresponding derived `Problem` class. So far, a `ProblemMs` class is integrated which implements the Direct Multiple Shooting Method and, e.g., provides evaluation of matching conditions and generation of sensitivities. It serves as a base class for `ProblemMsOcp` class and `ProblemMsParaOcp` class to set up an OCP or a Bilevel Inverse OCP, respectively. Our DISIMFAS is implemented in the derived `ProblemMsParaOcpActiveSetFix` class of the abstract class `ProblemMsParaOcp`.

#### *NLP Translator* Module

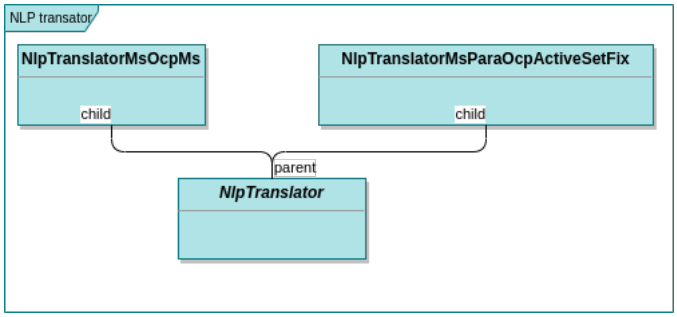


**Figure 7.3:** This figure depicts the PARDYNOPT structure of the *NLP translator* module on a conceptual level.

Each specific problem can be translated into the corresponding NLP within the *NLP translator* module in Figure 7.3. Therein, the abstract class `NlpTranslator` serves as a base class for the derived classes `NlpTranslatorMsOcp` and `NlpTranslatorMsParaOcpActiveSetFix`, which are related to the classes of the *problem* module `ProblemMsOcp` and `ProblemMsParaOcpActiveSetFix`, respectively. Within this structure an extension for other solution methods can easily be implemented.

#### *NLP Solver* Module

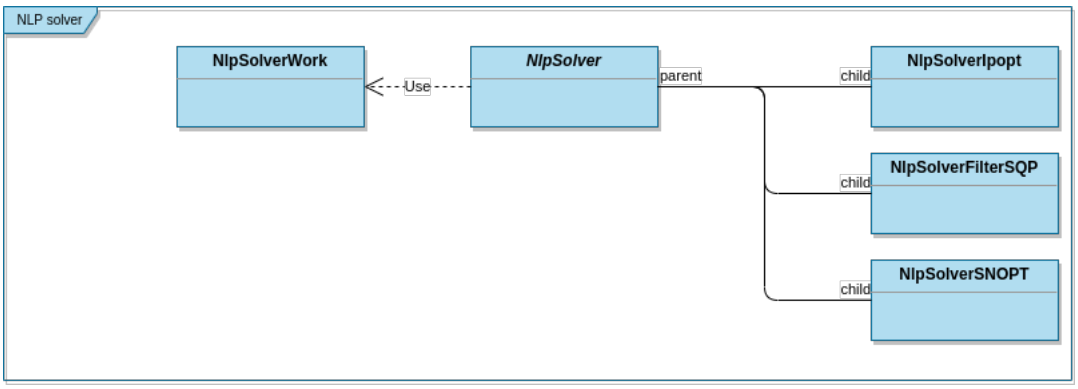


**Figure 7.4:** This figure depicts the PARDYNOPT structure of the *NLP solver* module on a conceptual level.

The `NlpSolverWork` struct serves as a collection of all data needed to set up a NLP within the module *NLP solver* in Figure 7.4. Therein, interfaces to various NLP solvers are provided, e.g. an interior-point method is implemented in the IPOPT framework within the `NlpSolverIpopt` class. In an actual implementation of the virtual method `Problem::solve()` an object of the corresponding `NlpTranslator` class is instantiated which operates on the data stored in a `NlpSolverWork` object. All needed quantities by the chosen NLP solver such as objective, constraints, Jacobians, Hessians of the Lagrangian as well as primal and dual variables with all dimensions are provided within methods implemented in the `NlpTranslator` class, which in turn uses attributes and methods of the corresponding `Problem` instance.

***Internal Structure* and *Function Evaluation/Derivative Generation* Modules**

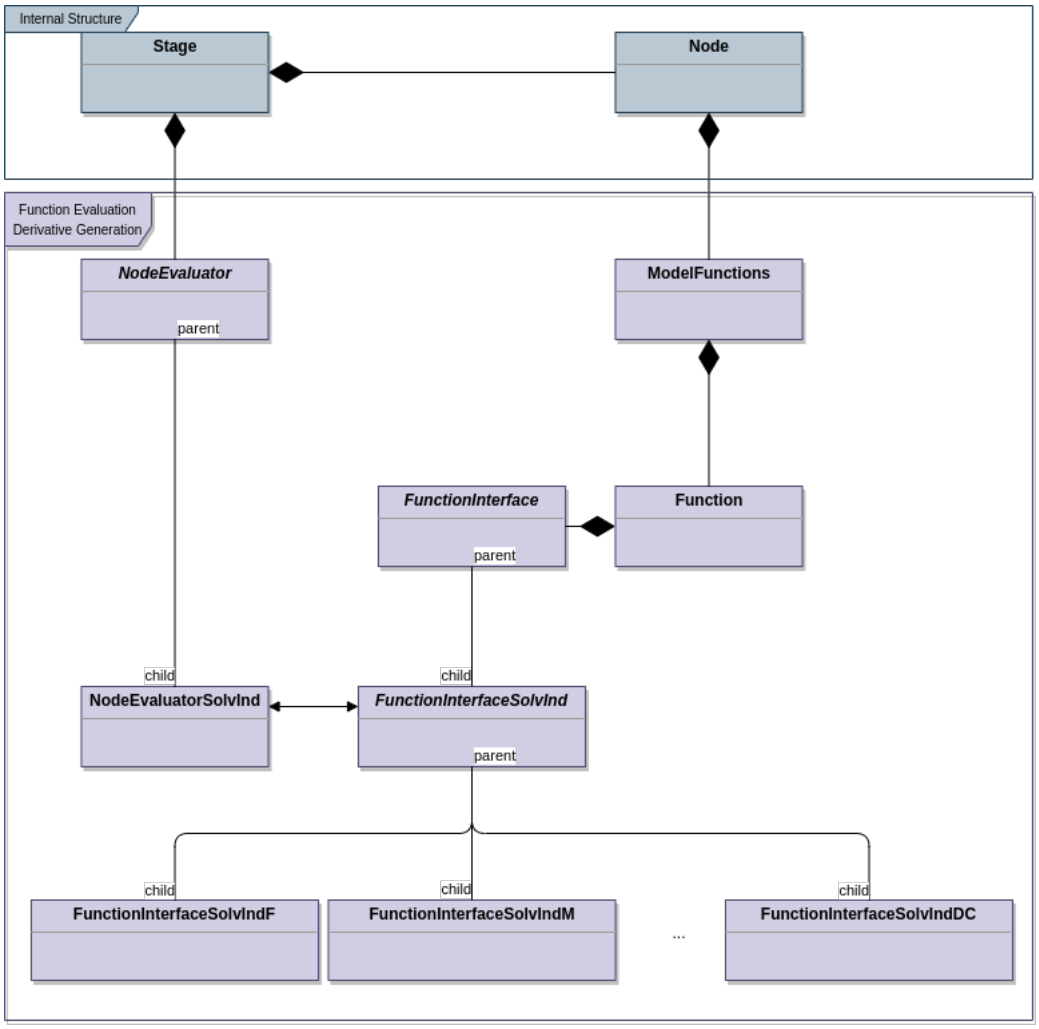


**Figure 7.5:** This figure depicts the PARDYNOPT structure of the *internal structure* module and the *function evaluation/derivative generation* module on a conceptual level.

Each `Problem` object incorporates arrays of pointers to `Stage` instances related to given model stages which itself comprise arrays of pointers to `Node` objects related to the discretization grid. This setup is depicted in the module *internal structure* in Figure 7.5. Within this structure the model variables and function evaluations with derivative generation can be spread out on `Node` level with an interface to the module *function evaluation/derivative generation*. So-called *mappers* are implemented for an efficient way to update `Node` variables with model variables on `Problem` level on the one side, and update results of function evaluations or derivatives from `Node` level to realizations on `Problem` level on the other side. All needed data by the NLP solver is first evaluated on `Node` level and later collected according to the problem at hand in a second step. This facilitates a parallelization of the source code in the future. Each `Stage` is described by its own set of model functions and each `Stage` object comprises an array of pointers to `NodeEvaluator` instances according to the number of `Node` objects with access to the `Model` object and the `ModelDescription` object therein. On `Node` level all model functions are provided in struct `ModelFunction`. They in turn comprise pointers to `Function` objects with `FunctionInterface` as actual interfaces to specific model functions and its derivatives enabled by the `NodeEvaluator` instance. In our special implementation of an `NodeEvaluatorSolvInd` class, the actual function evaluations and derivative generations are provided by derived classes of the base class

FunctionInterfaceSolvInd. Therein, also solutions of ODE systems and the corresponding sensitivities can be performed using the powerful software suite SOLVIND.

## 7.2 Framework in PARDYNOPT for OCPs and Bilevel Inverse OCPs

In this section we present how to set up an OCP and a Bilevel Inverse OCP in PARDYNOPT within the ProblemMsOcp class and the ProblemMsParaOcpActiveSetFix class, respectively. In appendix A.2 we provide a complete example - the rocket car example (described in more detail in chapter 8).

### 7.2.1 Setting up Problems in PARDYNOPT

In the software package PARDYNOPT a specific Problem can be instatiated and solved by first providing a ProblemDescription object. The user is advised to call a constructor of the ProblemDescription class as the one in Listing 7.1. The first parameter describes an array of pointers to Model instances per model stage. In the actual implementation of PARDYNOPT one ModelSolvInd object per stage can be instantiated by calling the constructor

```
ModelSolvInd(std::string model_file, std::string model_name).
```

SOLVIND models have to be set up by the user, see an example in Listing A.3 in appendix A.2. Here all model functions, like right-hand-side, objective functions and constraints with all corresponding function output dimensions as well as dimensions for states, controls, model parameters and duration parameters are specified. Furthermore, the second parameter in the ProblemDescription constructor in Listing 7.1 describes the number of model stages, and the following parameter an array of stage types. Therein, StageType::dynamic determines the corresponding stage to be dynamic, and StageType::transition declares a stage to be of duration 0 with a given function to describe a transition. In the last parameter of the constructor of the class ProblemDescription, an array of numbers of nodes per model stage are defined. In the Direct Multiple Shooting Method the number of nodes correspond to the number of multiple shooting nodes per model stage.

**Listing 7.1:** Constructor of ProblemDescription class.

```
1  /**
2   * \brief   Construct a new ProblemDescription object.
3   * \param   model           Array of models.
4   * \param   num_stages      Number of stages.
5   * \param   stage_type      Array of stage types.
6   * \param   num_nodes       Array of numbers of nodes.
7   */
8  ProblemDescription(Model** model,
9                     Dimension num_stages,
10                    StageType* stage_type,
11                    Dimension* num_nodes);
```

With several setting methods, as the ones in Listing 7.2, the predefined model discretization, node evaluator and NLP solver types can be changed by the user after instantiation of an ProblemDescription object. So far for model discretization the Direct Multiple Shooting Method is implemented and, hence, DynamicDiscretizationType::multiple_shooting is available with GridType::simple, GridType::equidistant_normalized or GridType::userdefined, where the user can specify the preferred multiple shooting time grid. By default GridType::simple is activated. In the actual implementation of PARDYNOPT we use the ControlDiscretizer::solvind, where ControlDiscretizationType::piecewise_const is set on the multiple shooting time grid for a piecewise constant control approximation. If nonlinear constraints appear in the problem formulations they are enforced on the multiple shooting grids according to the Direct Multiple Shooting Method as described in subsection 2.2.1. Nonlinear mixed control-state constraints or initial and final conditions can be defined in the SOLVIND model, see [5]. Usually, they are treated as decoupled equality

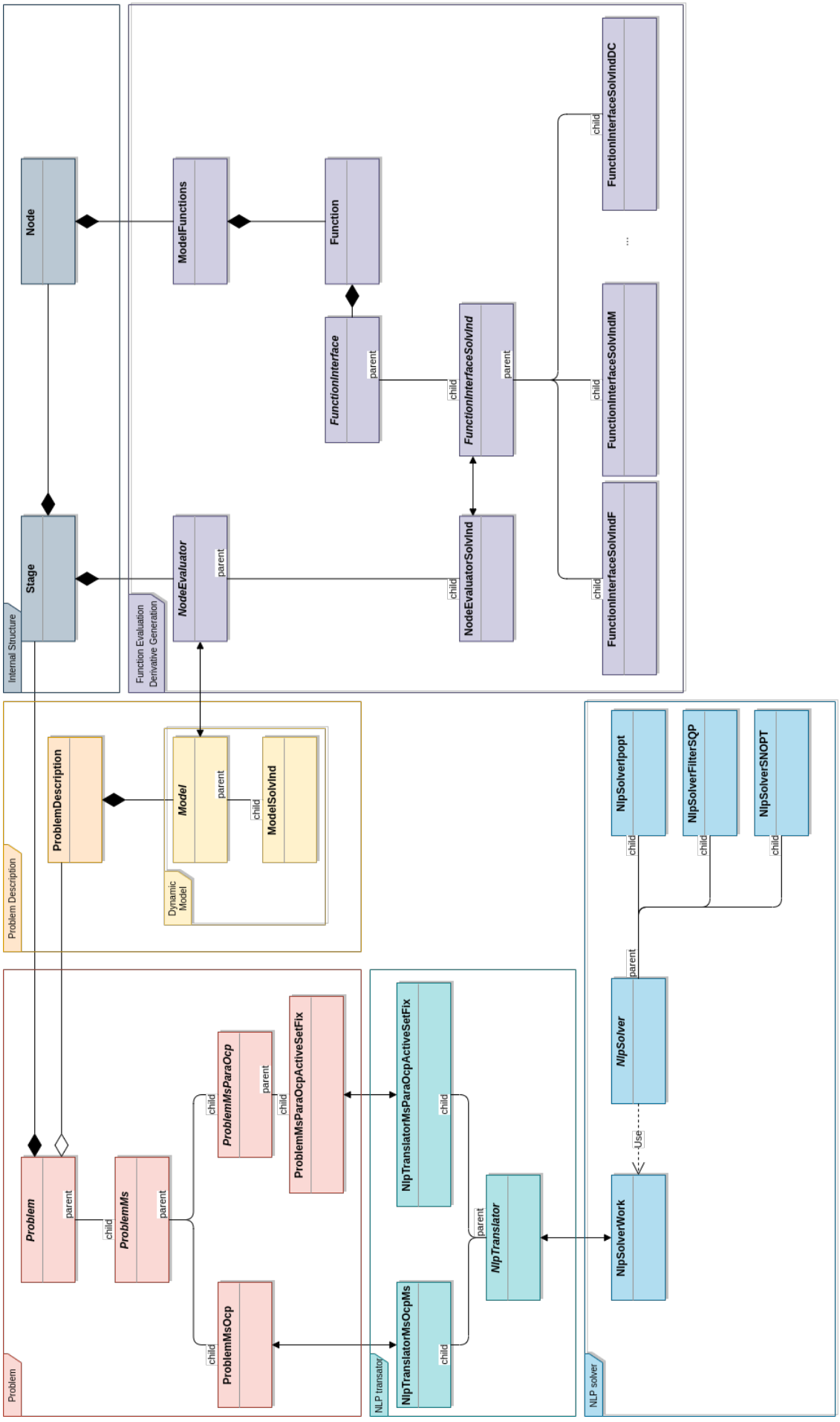


**Figure 7.6:** Total PARDYNOPT structure on a conceptual level.

and inequality constraints on the corresponding multiple shooting nodes. If periodicity constraints have to be taken into account, they can be realized with coupled equality and inequality constraints defined on the first and last shooting node. For a more detailed introduction how to set up a SOLVIND model we refer to the source code and the work of Albersmeyer in [5]. Note, that so far in PARDYNOPT all appearing inequality constraints in the resulting NLP beside simple bounds on variables are formulated as equality constraints by introducing slack variables. Furthermore, in PARDYNOPT `NodeEvaluatorType::solvind` for function evaluation and derivative generation is available, and `NlpSolverType::ipopt` can be used. In the future extensions for these specifications will be provided. Before a `ProblemDescription` object is passed to a specific `Problem` class, we first have to call the method `finalize_problem_description()` to indicate the finalization of the description of the problem at hand.

**Listing 7.2:** Setting methods in ProblemDescription class

```
/**
 * \brief   Set the model discretization types.
 * \param   dyn_disc_type    Dynamic discretization type.
 * \param   dyn_grid_type    Dynamic grid type.
 * \param   ctrl_disc_type   Array of control discretization types.
 */
void set_model_discretization_types(DynamicDiscretizationType dyn_disc_type,
                                    GridType dyn_grid_type,
                                    ControlDiscretizationType** ctrl_disc_type);

/**
 * \brief Set the node evaluator type.
 * \param node_eval_type Node evaluator type.
 */
void set_node_evaluator_type(NodeEvaluatorType node_eval_type);

/**
 * \brief Set the nlp solver type.
 * \param nlp_solver_type Nlp solver type.
 */
void set_nlp_solver_type(NlpSolverType nlp_solver_type);

/**
 * \brief   Finalize problem description.
 */
void finalize_problem_description();
```

Now we can instantiate the preferred `Problem` by calling a constructor of the derived class `ProblemMsOcp` for solving an OCP with the Direct Multiple Shooting Method as described in subsection 2.2.1, or a constructor of `ProblemMsParaOcpActiveSetFix` for solving an Bilevel Inverse OCP with the DISIMFAS derived in chapter 4, see Listing 7.3. Within this software design, the implementation of further problem formulations might be realized under consideration of already available modules.

**Listing 7.3:** Constructors of `ProblemMsOcp` and `ProblemMsParaOcpActiveSetFix`.

```
/**
 * \brief   Construct a new ProblemMsOcp object.
 * \details By calling this constructor an instance for solving
 *          an Optimal Control Problem using Multiple Shooting
 *          for discretization is created.
 * \param   problem_desc Problem description defined by user.
 */
ProblemMsOcp(ProblemDescription* problem_desc);

/**
```

```
11   * \brief   Construct a new ProblemMsParaOcpActiveSetFix object.
12   * \details By calling this constructor an instance for solving
13   *          a Parameter Estimation Problem constrained by an
14   *          Optimal Control Problem using the Direct Simultaneous
15   *          Optimization Approach with Fixed Active Set is created.
16   * \param   problem_desc Problem description defined by user.
17   */
18  ProblemMsParaOcpActiveSetFix(ProblemDescription* problem_desc);
```

After creating an object of the problem at hand, several initialization methods are available (some are depicted in Listing A.1 in appendix A.1). The methods which can be used for both problems, the `ProblemMsOcp` and the `ProblemMsParaOcpActiveSetFix`, are implemented in the parent class `ProblemMs`.

Finally, by calling the method `finalize_initialization()` the problem is indicated to be fully initialized such that the solution process of the resulting NLP can be initiated with the method `solve()`, see Listing 7.4. The results are written on screen and printed in text files for visualization by a provided python script.

**Listing 7.4:** Finalize initialization and solve a specific Problem.

```
1   /**
2    * \brief Finalize initialization.
3    * \details
4    */
5   virtual void finalize_initialization() override;
6
7
8   /**
9    * \brief Solve the Problem.
10   * \details
11   */
12  virtual void solve() override;
```

So far, technical settings for SOLVIND and the chosen NLP solver IPOPT can be specified in the source code of PARDYNOPT. A suitable user interface with more options will be provided in the future. Please note that in this chapter not all features implemented in PARDYNOPT are described and only a brief overview is given with selected options. For a more detailed description we refer the reader to the source code. of PARDYNOPT.

# Chapter 8

# Bilevel Inverse Optimal Control in Two Case Studies

In this chapter we present two case studies for assessing the applicability and performance of our DISIMFAS described in chapter 4, which is implemented in the software package PARDYNOPT, see chapter 7. In both studies we use simulated measurements for a better investigation of the proposed method to identify corresponding objective weights and model parameters. In section 8.1 a case study for a rocket car example is given. We investigate the solutions of OCPs for various cases and formulate single and multi-stage Bilevel Inverse OCPs, which incorporate the known structure of the control function. The intention of this study is to show that exploited structure information can be successfully integrated in our DISIMFAS. In section 8.2 we give an example for a rigid multibody system - the polar robot as deduced in [164]. Here, we first analyze the DISIMFAS with regard to differently structured initial guesses and second, compare our results with the solution given in [80], which was achieved with the Direct All-at-Once Approach implemented in PARAOCP.

## 8.1 Case Study: Rocket Car and Multi-Stage Formulations

We consider a rocket car example, as introduced, e.g., in [80, 81] and set up a single stage OCP of the form (2.1). It describes an optimal movement of a car from one position to another including friction and fuel consumption. The solutions are investigated for three selected settings and structures of the control functions are exploited. With this information and simulated measurements, we then formulate various single and multi-stage Bilevel Inverse OCPs of the form (2.21) tailored to the specific settings of the lower level OCPs for the rocket car example and solve them using our DISIMFAS to determine optimal weights and model parameters.

### 8.1.1 An OCP for the Rocket Car and its Solutions for Selected Settings

We start with a general problem formulation for a single stage OCP of the form (2.1) describing the optimal movement of a rocket car, where the end time is unknown a priori. As a result of the time transformation onto a fixed time horizon, $\mathcal{T} = [0,1]$, the stage duration parameter $d_1$ enters the following OCP together with the other quantities defined in Table 8.1 as follows:

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \boldsymbol{\Phi}(\cdot) = \alpha_1 \cdot d_1 - x_2(1) \tag{8.1a}$$

$$\text{s.t.} \quad \dot{x}_0(t) = d_1 x_1(t), \qquad t \in \mathcal{T}, \tag{8.1b}$$

$$\dot{x}_1(t) = d_1 \left(u_0(t) - p_0 x_1^2(t)\right) / \left(x_2(t) + \gamma\right), \qquad t \in \mathcal{T}, \tag{8.1c}$$

$$\dot{x}_2(t) = -d_1 \varrho u_0^2(t), \qquad t \in \mathcal{T}, \tag{8.1d}$$

$$x_0(0) = 0, x_1(0) = 0, x_2(0) = 1, \tag{8.1e}$$

$$x_0(1) = 10, x_1(1) = 1, \tag{8.1f}$$

$$-10 \le x_0(t) \le 10, \qquad t \in \mathcal{T}, \tag{8.1g}$$

$$-10 \le x_1(t) \le 10, \qquad t \in \mathcal{T}, \tag{8.1h}$$

$$0 \le x_2(t) \le 10, \qquad t \in \mathcal{T}, \tag{8.1i}$$

$$-1 \le u_0(t) \le 1, \qquad t \in \mathcal{T}, \tag{8.1j}$$

$$0 \le d_1, \tag{8.1k}$$

where the value of the control function $u_0(t)$ lies between $-1$ and 1. The objective function, $\Phi(\cdot)$, is a combination of end time, $d_1$, and fuel consumption, $-x_2(1)$. For comparison of solutions of (8.1) with differently weighted objective functions, we choose three settings for the weight $\alpha_1$ as shown in Table 8.2. With increasing value of $\alpha_1$ the contribution of the duration parameter in the objective function increases in order to reach the end position faster. The friction parameter arises in the term $p_0 x_1^2(t)/\left(x_2(t)+\gamma\right)$ which corresponds to Newton friction and is set to $p_0 = 0.1$ in all OCPs formulations, see Table 8.2.

**Table 8.1:** Definition of quantities which appear in OCP (8.1) for the rocket car example, as well as in the OCP and Bilevel Inverse OCP formulations in the following subsection. In the OCPs the friction parameter is set to $p_0 = 0.1$. In the Bilevel Inverse OCPs this parameter is determined together with the objective weight $\alpha_1$.

| Symbol | Description | Value |
|---|---|---|
| $x_0$ | position | $x_0(0) = 0, x_0(1) = 10$ |
| $x_1$ | velocity | $x_1(0) = 0, x_1(1) = 1$ |
| $x_2$ | fuel mass | $x_2(0) = 1$ |
| $u_0$ | control force | |
| $d_1$ | stage duration | |
| $p_0$ | friction | |
| $\gamma$ | car mass | 0.1 |
| $\varrho$ | fuel conversion rate | 0.1 |

For all cases the OCPs are solved with the software package ParDynOpt and its implementation of the Direct Multiple Shooting Method [26] as described in subsection 2.2.1 with an equidistant multiple shooting grid with 20 nodes and constant controls on each multiple shooting interval. Furthermore, for the arising initial value problems SolvIND is used with an adaptive BDF method in DAESOL-II [5] and its sensitivity and derivative generation with its interface to ADOL-C [171]. A interior-point method with its implementation in Ipopt [169] solves the resulting NLP with an exact Hessian approximation and a convergence tolerance set to $10^{-6}$ (defined in [169] as error tolerance), see also Table 8.4. The initial guesses for the state values are set to $\begin{pmatrix}0 & 0 & 0\end{pmatrix}^T$ at each multiple shooting node and the initial guesses for the values of all control parameters are set to 1. The results are illustrated in Figure 8.1 and summarized in Table 8.3. The violation of the constraints in the resulting NLPs as well as the infeasibility measure for the corresponding dual variables, see Ipopt [169], are comparably small.

**Table 8.2:** Three cases with selected settings for the objective weight $\alpha_1$ and the friction parameter $p_0$ in OCPs of the rocket car example.

| | $\alpha_1$ | $p_0$ |
|---|---|---|
| case 1 | 0.06 | 0.1 |
| case 2 | 0.2 | 0.1 |
| case 3 | 0.5 | 0.1 |

All three settings lead to different solutions and especially to different structures of the approximation of the control function $u_0(t)$ as shown in Figure 8.2. In case 1 the approximation of the control function lies completely in the interior of the interval $[-1,1]$, which means that no control bound is active at any time $t \in \mathcal{T}$. In contrast to the first case, for the remaining two settings the control function approximation reaches its maximum value in the first phase such that the upper bound 1 gets active. After a second phase, where the approximation of the control function lies in the interior for case 2 and case 3, only for the latter case the control function reaches its minimum value $-1$ in a third phase.

**Table 8.3:** Computational results of OCP (8.1) for the rocket car example in the three selected settings summarized in Table 8.2. The computations are performed with the Direct Multiple Shooting Method implemented in ParDynOpt.

| | case 1 ($\alpha_1 = 0.06$) | case 2 ($\alpha_1 = 0.2$) | case 3 ($\alpha_1 = 0.5$) |
|---|---|---|---|
| objective $\Phi^*$ | $-3.7796 \cdot 10^{-1}$ | $5.5224 \cdot 10^{-1}$ | 2.2570 |
| constraint violation | $4.40 \cdot 10^{-9}$ | $1.23 \cdot 10^{-8}$ | $1.20 \cdot 10^{-7}$ |
| dual infeasibility | $6.57 \cdot 10^{-10}$ | $2.08 \cdot 10^{-9}$ | $2.52 \cdot 10^{-8}$ |
| # iterations | 20 | 20 | 20 |
| $d_1^*$ | 7.8931 | 6.0072 | 5.5453 |

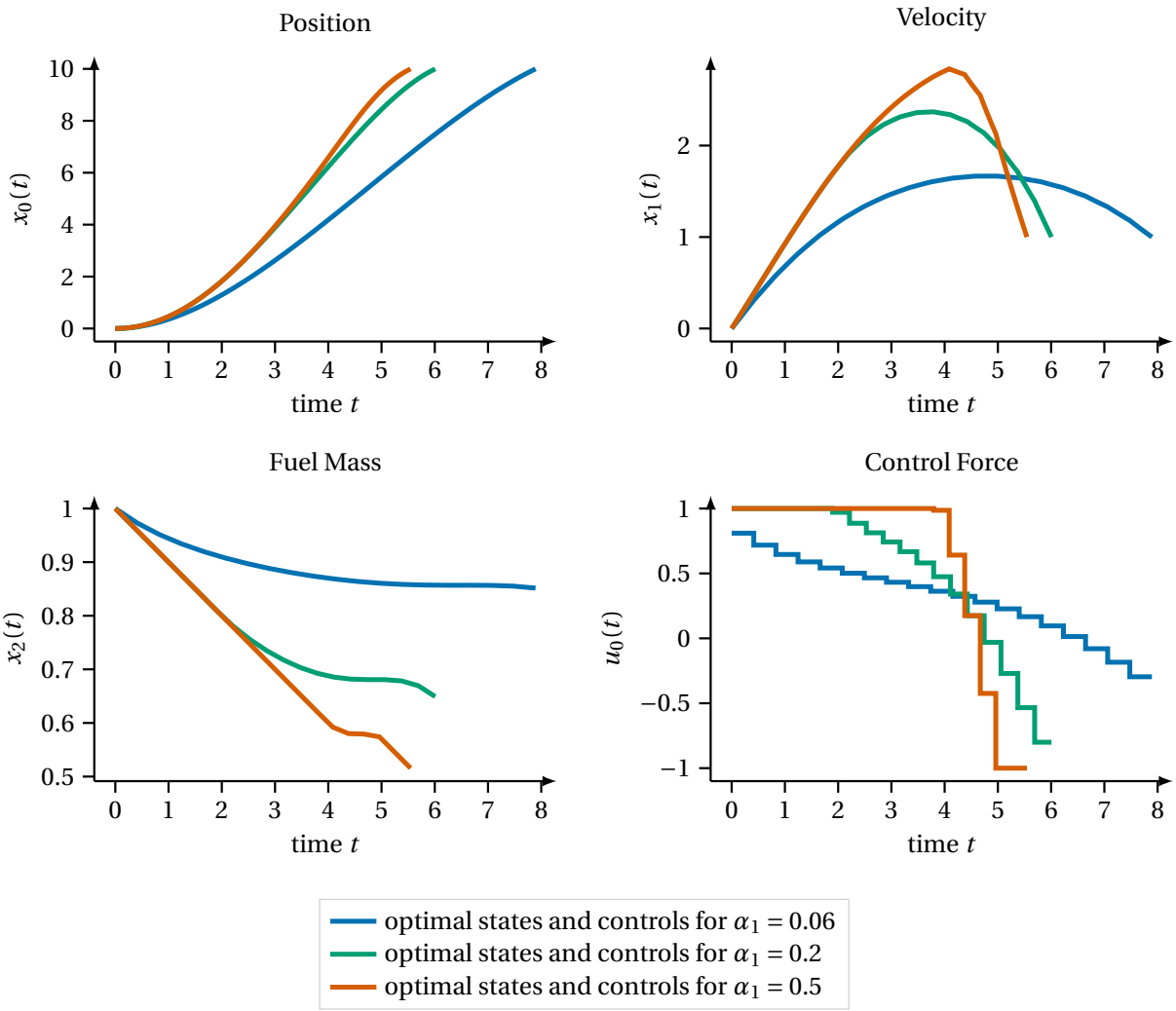


**Figure 8.1:** Optimal differential states and controls in the solutions of OCPs with differently weighted objective for three cases as summarized in Table 8.2. All results are computed with ParDynOpt by applying the Direct Multiple Shooting Method with 20 shooting nodes.

### 8.1.2 Multi-Stage Bilevel Inverse OCPs for Selected Settings

Now we incorporate the gained knowledge from subsection 8.1.1 and formulate suitable Bilevel Inverse OCPs of the form (2.21), which cover the different structures of the control function approximations in the three cases, see Figure 8.2. In the following we set up a single stage problem for case 1, a two-stage problem for case 2, and a three-stage problem for case 3. The resulting Bilevel Inverse OCPs are then solved with our DiSimFAS implemented in ParDynOpt as described in chapter 4.

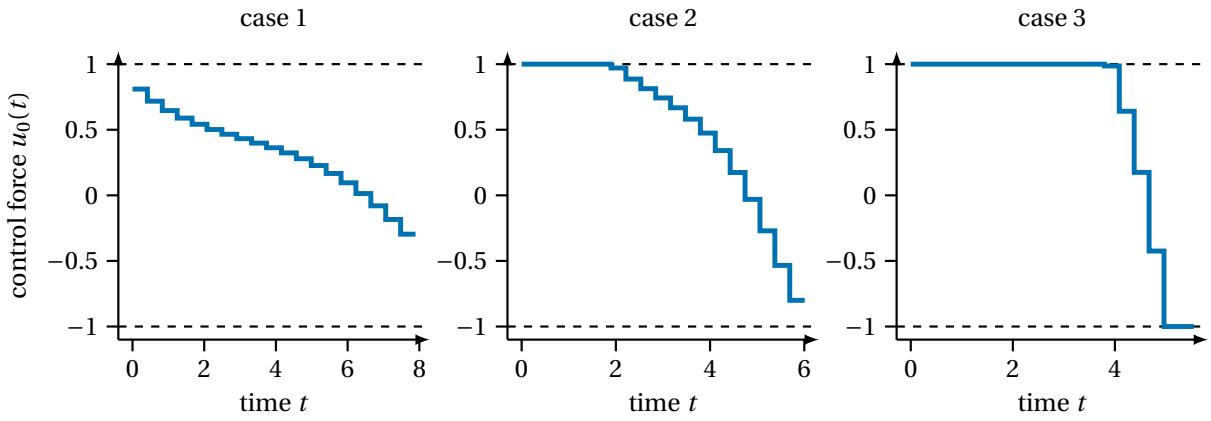


**Figure 8.2:** This figure depicts the structures of the approximation of the optimal control function in the solutions of the OCP (8.1) for selected settings from Table 8.2. In case 1 no bounds of the control parameters are active, whereas in case 2 and case 3 in a first phase the upper bound is active and then in a second phase no bounds are active. Only for case 3 the lower bound of the control parameters is active in a third phase.

In all cases we choose equidistant multiple shooting grids on all model stages and piecewise constant control approximations on the same time grids. Furthermore, SOLVIND is used with an adaptive BDF method DAESOL-II [5] for solving the initial value problems arising in the lower level OCPs and the implemented sensitivity and derivative generation with its interface to ADOL-C [171]. The resulting NLPs are then solved with IPOPT [169] with a convergence tolerance set to $10^{-6}$ (defined in [169] as error tolerance). For the constraint Jacobian we choose the structure exploiting implementation in PARDYNOPT and for the Hessian of the Lagrangian a limited-memory quasi-NEWTON method (L-BFGS update formula) in IPOPT.

**Table 8.4:** Setting used in PARDYNOPT to solve the one stage OCP (8.1), the two-stage OCP (8.3), the three-stage OCP (8.3) and their corresponding Bilevel Inverse OCPs (8.2), (8.4) and (8.6).

| | OCPs | Bilevel Inverse OCPs |
|---|---|---|
| solution method | Direct Multiple Shooting Method | Direct Simultaneous Approach with Fixed Active Set |
| # multiple shooting nodes | (20), (7,14), (14,5,3) | (20), (7,14), (14,5,3) |
| NLP solver | IPOPT | IPOPT |
| convergence tolerance | $10^{-6}$ | $10^{-6}$ |
| constraint Jacobian | sparse | sparse |
| Hessian | exact | BFGS update |
| integrator and derivatives | SOLVIND | SOLVIND |
| initial guess $\alpha_1$ | - | 1 |
| initial guess $p_0$ | - | 0.2 |

Pseudo-measurements are generated in all cases from solutions $\boldsymbol{x}^{*\mathrm{OCP}}$ of the corresponding lower level OCPs (8.1), (8.3), and (8.5). In case 1 independent, normally distributed noise $\boldsymbol{\varepsilon}$ is added with zero mean and a standard deviation $\boldsymbol{\sigma}$ with values $\sigma_0 = 0.05 \cdot 5, \sigma_1 = 0.05 \cdot 1.0, \sigma_2 = 0.05 \cdot 0.9$ by

$$\eta_{\mathrm{nk}} = x_{\mathrm{k}}^{*\mathrm{OCP}}(t_{\mathrm{n}}^{\mathrm{m}}) + \varepsilon_{\mathrm{nk}}, \qquad \mathrm{n} = 0, \ldots, \mathrm{n_m} - 1,\ \mathrm{k} = 0, 1, 2.$$

In case 2 and case 3 the pseudo-measurements are generated with

$$\eta_{\mathrm{jnk}} = x_{\mathrm{jk}}^{*\mathrm{OCP}}(t_{\mathrm{jn}}^{\mathrm{m}}) + \varepsilon_{\mathrm{jnk}}, \qquad \mathrm{n} = 0, \ldots, \mathrm{n}_{\mathrm{m}}^{\mathrm{j}} - 1,\ \mathrm{k} = 0, 1, 2,$$

by adding noise with zero mean and a standard deviation of $\sigma_0 = 0.05 \cdot 5, \sigma_1 = 0.05 \cdot 1.2, \sigma_2 = 0.05 \cdot 0.8$ for case 2 with the stage indices j = 1,2, and a standard deviation of $\sigma_0 = 0.05 \cdot 5, \sigma_1 = 0.05 \cdot 1.5, \sigma_2 = 0.05 \cdot 0.7$ for case 3 with j = 1,2,3. The measurement time grids are chosen to be the same as the multiple shooting grids. For case 1 the number of multiple shooting nodes is 20 as in the previous subsection, summarized in Table 8.4. The discretization of case 2 results in 7 multiple shooting nodes for the first model stage and 14 for the second. For case 3 with three model stages we choose a grid of 14 for the first stage, 5 for the second and 3 for the last model stage. In all cases, for measurement generation the OCPs are solved with PARDYNOPT in a similar setting as described in the previous subsection, see Table 8.4.

For solving the Bilevel Inverse OCPs, in all three cases the initial guess for the objective weight $\alpha_1$ is set to 1 and for the model parameter $p_0$ to 0.2. The initial state values are set to the simulated pseudo-measurements at each multiple shooting node and the initial values for all duration parameters and control parameters are chosen to be 1. Only the control parameters in the third stage of the third case are set to −1, where they are fixed to the lower bound. The setting in PARDYNOPT used to solve Bilevel Inverse OCPs for all cases is summarized in Table 8.4.

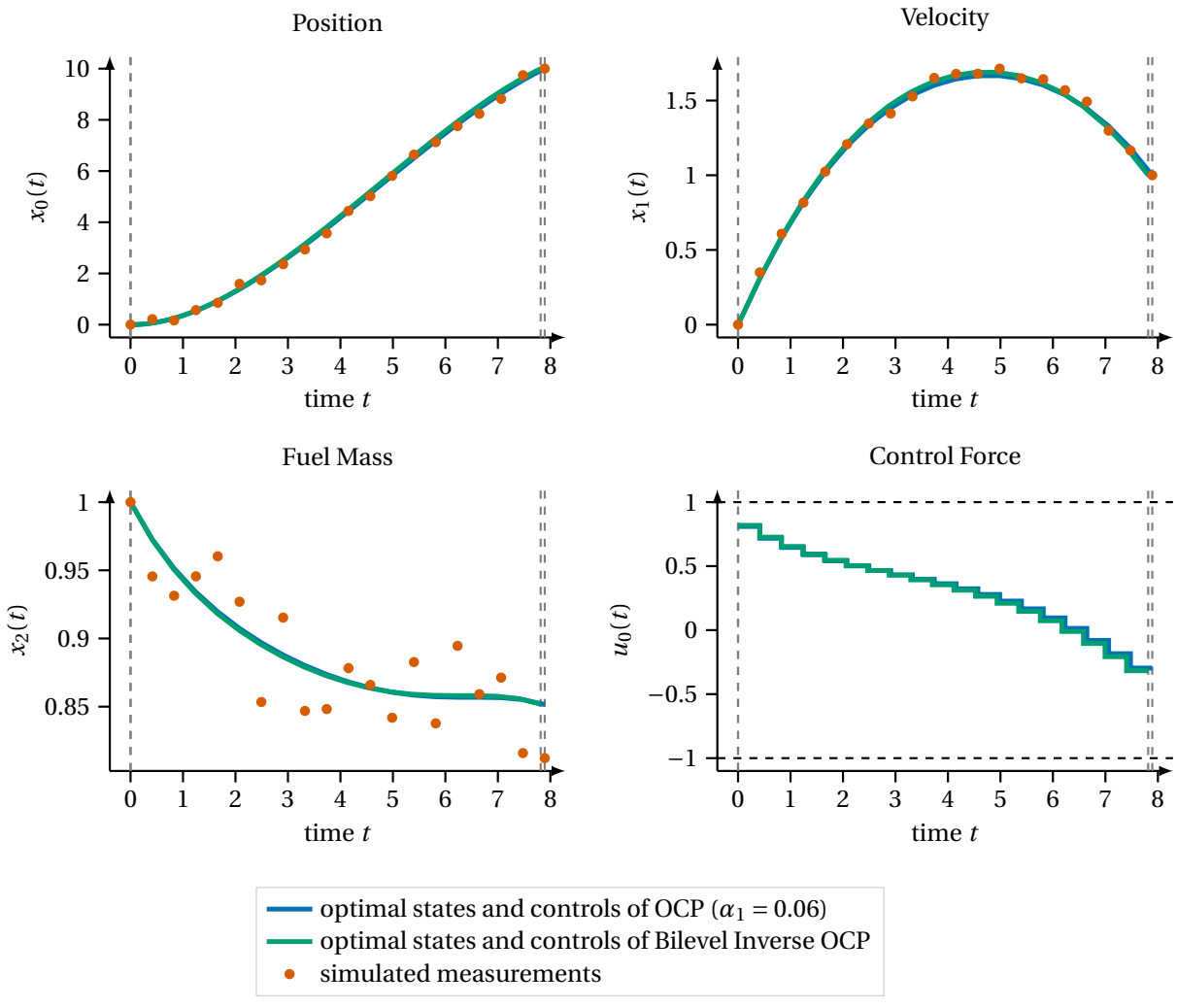


**Figure 8.3:** This figure depicts the optimal differential states and controls in the solution of Bilevel Inverse OCP (8.2) for case 1 computed with the DISIMFAS in PARDYNOPT with 20 multiple shooting nodes. This result is compared to the corresponding OCP (8.1) solution with PARDYNOPT by applying the Direct Multiple Shooting Method. Because the solutions are very similar the lines mostly overlap in the illustration. With the OCP solution pseudo-measurements are generated as described at the beginning of subsection 8.1.2. They are illustrated in the plots by dots.

### Single-Stage Bilevel Inverse OCP for Case 1

The control function for case 1, see Figure 8.2, lies completely in the interior of the interval $[-1,1]$, therefore, we choose a one stage formulation to solve a Bilevel Inverse OCP of the form

$$\min_{\substack{\alpha_1,p_0,\\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \boldsymbol{\Psi}(\cdot) = \frac{1}{2}\sum_{\mathrm{n}=0}^{\mathrm{n_m}-1}\sum_{\mathrm{k}=0}^{2}\frac{\left(x_{\mathrm{k}}(t_{\mathrm{n}}^{\mathrm{m}})-\eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{k}}^2} \tag{8.2a}$$

$$\text{s.t.} \quad (\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}) \text{ solve OCP (8.1)}, \tag{8.2b}$$

$$0 \le \alpha_1 \le 10, \tag{8.2c}$$

$$-20 \le p_0 \le 20, \tag{8.2d}$$

where the upper level PE is constrained by the lower level OCP (8.1). To apply our DISIMFAS described in chapter 4 only equality constraints of the discretized lower level OCP and bounds on the decision variables related to the fixed active set are considered in the resulting NLP (4.13). Because of the chosen loose bounds on the state variables and the known structure of the control approximation, in case 1 the working set $\mathcal{W}$ as a guess of the active set $\mathcal{A}^*$ is empty. Hence, all bounds on the NLP variables related to the constraints (8.1g)-(8.1k) are neglected. In the solution of Bilevel Inverse OCP (8.2) these bound constraints are still satisfied. In Figure 8.3 the optimal differential states and control parameters in the solution of (8.2) calculated with PARDYNOPT are illustrated. In Table 8.5 the results achieved with small constraint violation and infeasibility measure, see IPOPT [169], are summarized.

The DISIMFAS converges after 13 iterations with an objective value of 8.5863 and a stage duration of 7.8147. With a relative error of 1.20% for objective weight $\alpha_1^* = 6.0718 \cdot 10^{-2}$ and 7.40% for model parameter $p_0^* = 9.2603 \cdot 10^{-2}$, both quantities are reproduced with a relative error of less than 10%.

**Table 8.5:** Computational result of Bilevel Inverse OCP (8.2) for case 1 with our DISIMFAS in PARDYNOPT with settings summarized in Table 8.4 and comparison of the estimates $\alpha_1^*$ and $p_0^*$ to the true values and the phase duration $d_1^*$ to the corresponding OCP (8.1) solution.

| case 1 | Bilevel Inverse OCP (8.2) | |
|---|---|---|
| objective $\boldsymbol{\Psi}^*$ | 8.5863 | |
| constraint violation | $4.33 \cdot 10^{-12}$ | |
| dual infeasibility | $1.79 \cdot 10^{-10}$ | |
| # iterations | 13 | |
| | | |
| parameters | estimate (rel. Err%) | true value |
| $\alpha_1^*$ | $6.0718 \cdot 10^{-2}$ (1.20%) | 0.06 |
| $p_0^*$ | $9.2603 \cdot 10^{-2}$ (7.40%) | 0.1 |
| | | |
| stage duration | | OCP (8.1) |
| $d_1^*$ | 7.8147 | 7.8931 |

### Two-Stage Bilevel Inverse OCP for Case 2

In the solution of (8.1) with the settings of case 2 the upper bound of the control function is active in a first phase. After this phase it lies in the interior of the interval $[-1,1]$ as in the results of case 1, see Figure 8.2. Incorporating this structural information the single stage OCP (8.1) can be reformulated into a two-stage OCP of the following form

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \boldsymbol{\Phi}(\cdot) = \alpha_1 \cdot (d_1 + d_2) - x_{22}(1) \tag{8.3a}$$

$$\text{s.t.} \quad \dot{x}_{\mathrm{j}0}(t) = d_{\mathrm{j}} x_{\mathrm{j}1}(t), \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2, \tag{8.3b}$$

$$\dot{x}_{\mathrm{j}1}(t) = d_{\mathrm{j}} \left( u_{\mathrm{j}0}(t) - p_0 {x_{\mathrm{j}1}}^2(t) \right) / \left( x_{\mathrm{j}2}(t) + \gamma \right), \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2, \tag{8.3c}$$

$$\dot{x}_{\mathrm{j}2}(t) = -d_{\mathrm{j}} \varrho {u_{\mathrm{j}0}}^2(t), \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2, \tag{8.3d}$$

$$\boldsymbol{x}_1(1) = \boldsymbol{x}_2(0), \tag{8.3e}$$

$$x_{10}(0) = 0, x_{11}(0) = 0, x_{12}(0) = 1, \tag{8.3f}$$

$$x_{20}(1) = 10, x_{21}(1) = 1, \tag{8.3g}$$

$$-10 \leq x_{\mathrm{j}0}(t) \leq 10, \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2, \tag{8.3h}$$

$$-10 \leq x_{\mathrm{j}1}(t) \leq 10, \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2, \tag{8.3i}$$

$$0 \leq x_{\mathrm{j}2}(t) \leq 10, \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2, \tag{8.3j}$$

$$u_{10}(t) = 1, \qquad t \in \mathcal{T}^1, \tag{8.3k}$$

$$-1 \leq u_{20}(t) \leq 1, \qquad t \in \mathcal{T}^2, \tag{8.3l}$$

$$0 \leq \boldsymbol{d}, \tag{8.3m}$$

with the quantities defined in Table 8.1 extended by a stage subscript, j = 1,2, for the differential states and controls. Similar to OCP (8.1) we perform a time transformation on both stages onto fixed time horizons $\mathcal{T}^{\mathrm{j}} = [0,1]$, j = 1,2, such that stage duration parameters $\boldsymbol{d} = \begin{pmatrix} d_1 & d_2 \end{pmatrix}^T$ are added to the optimization variables in OCP (8.3).

With simulated measurements from the solution of (8.3) with the chosen objective weight $\alpha_1$ and model parameter $p_0$ for case 2 we set up a two-stage Bilevel Inverse OCP, where the upper level PE is constrained by the lower level OCP (8.3) as follows

$$\min_{\substack{\alpha_1, p_0, \\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \boldsymbol{\Psi}(\cdot) = \frac{1}{2} \sum_{\mathrm{j}=1}^{2} \sum_{\mathrm{n}=0}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1} \sum_{\mathrm{k}=0}^{2} \frac{\left( x_{\mathrm{jk}}(t_{\mathrm{jn}}^{\mathrm{m}}) - \eta_{\mathrm{jnk}} \right)^2}{\sigma_{\mathrm{k}}^2} \tag{8.4a}$$

$$\text{s.t.} \quad (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \text{ solve OCP (8.3)}, \tag{8.4b}$$

$$0 \leq \alpha_1 \leq 10, \tag{8.4c}$$

$$-20 \leq p_0 \leq 20. \tag{8.4d}$$

In the resulting NLP (4.13) of the DISIMFAS only equality constraints of the discretized lower level OCP and bounds on the decision variables related to the fixed active set arise. Similar to the previous case, under consideration of the known structure of the control approximation, all bounds on the NLP variables related to the constraints (8.3h)-(8.3j) and (8.3l) are neglected. However, in the solution of Bilevel Inverse OCP (8.4) these bound constraints are still satisfied. To incorporate the active upper bound of the control function in the solution of OCP (8.1) for case 2, the control function is fixed to its upper bound 1 in (8.3k) at the first model stage.

The DISIMFAS converges after 19 iterations with an objective value of 8.6764 and stage durations $d_1^* = 2.1839$ and $d_2^* = 3.7759$. With a relative error of 0.71% for objective weight $\alpha_1^* = 1.9859 \cdot 10^{-1}$ and 8.26% for model parameter $p_0^* = 9.1744 \cdot 10^{-2}$ both quantities are reproduced with a relative error of less than 10%.

In Figure 8.4 the optimal differential states and control parameters in the solution of Bilevel Inverse OCP (8.4) calculated with PARDYNOPT are illustrated and compared to the two-stage OCP (8.3) and the single-stage OCP (8.1) solutions for the second case. In Table 8.6 the results achieved with small constraint violation and infeasibility measure, see IPOPT [169], are summarized.

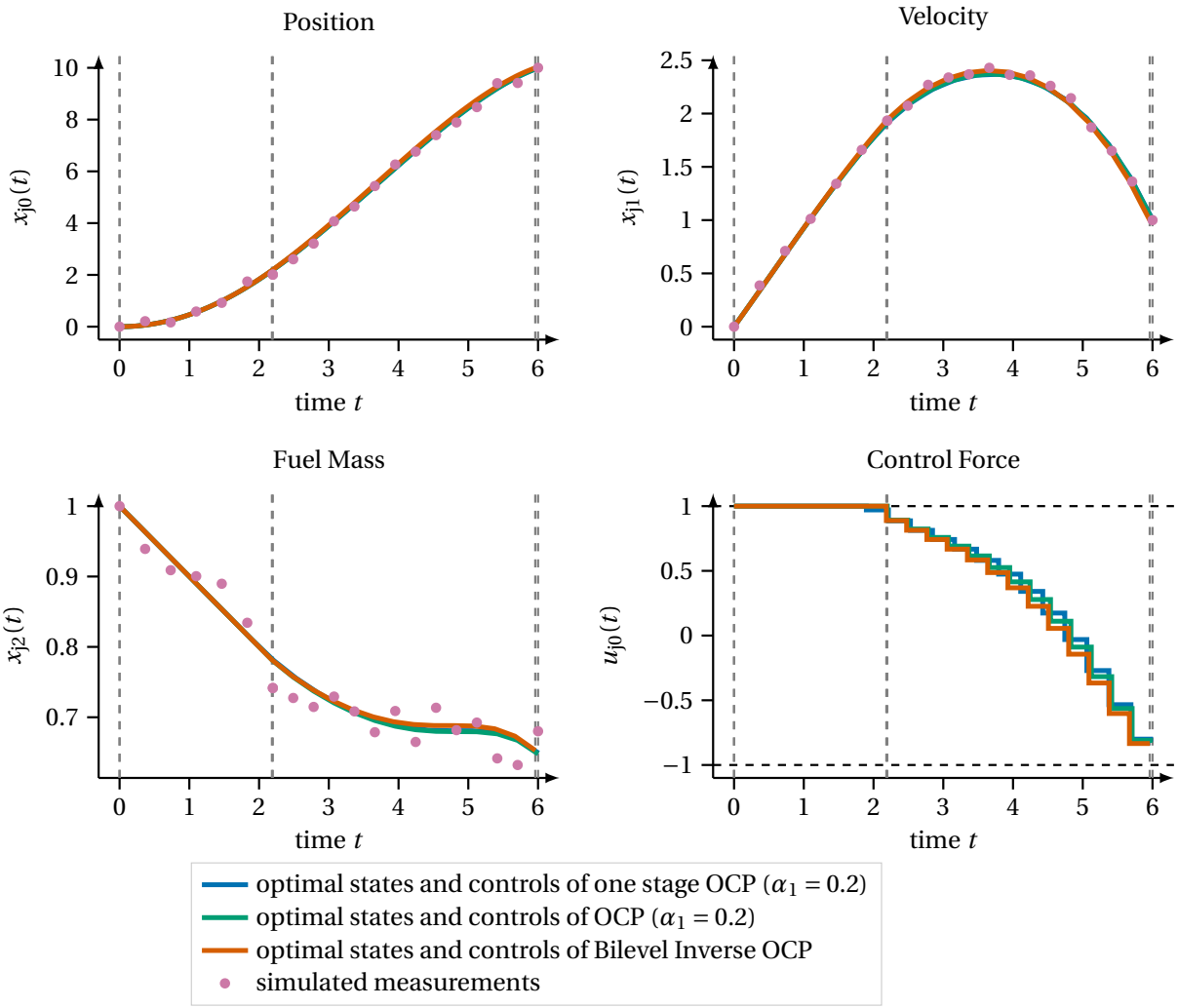


**Figure 8.4:** This figure depicts the optimal differential states and controls in the solution of Bilevel Inverse OCP (8.4) for case 2 computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes in the first model stage and 14 in the second. This result is compared to the corresponding OCP (8.3) solution and the single-stage OCP (8.1) solution with PARDYNOPT by applying the Direct Multiple Shooting Method. Because the solutions are very similar the lines mostly overlap in the illustration. With the OCP solution pseudo-measurements are generated as described at the beginning of subsection 8.1.2. They are illustrated in the plots by dots.

**Table 8.6:** Computational result of Bilevel Inverse OCP (8.4) for case 2 with our DISIMFAS in PARDYNOPT with settings summarized in Table 8.4 and comparison of the estimates $\alpha_1^*$ and $p_0^*$ to the true values and the phase durations $d_1^*$ and $d_2^*$ to the corresponding OCP (8.3) solution.

| case 2 | Bilevel Inverse OCP (8.4) | |
|---|---|---|
| objective $\Psi^*$ | 8.6764 | |
| constraint violation | $3.56 \cdot 10^{-11}$ | |
| dual infeasibility | $2.32 \cdot 10^{-7}$ | |
| # iterations | 19 | |
| | | |
| parameters | estimate (rel. Err%) | true value |
| $\alpha_1^*$ | $1.9859 \cdot 10^{-1}$ (0.71%) | 0.2 |
| $p_0^*$ | $9.1744 \cdot 10^{-2}$ (8.26%) | 0.1 |
| | | |
| stage durations | | OCP (8.3) |
| $d_1^*$ | 2.1839 | 2.1972 |
| $d_2^*$ | 3.7759 | 3.8032 |

#### Three-Stage Bilevel Inverse OCP for Case 3

In case 3 the upper and lower bounds of the control function are active in the solution of (8.1). First, the upper bound is active and after a phase where the control function lies between the bounds −1 and 1, the lower bound is active for the last phase, see Figure 8.2. With this information the single-stage OCP (8.1) can be reformulated into a three-stage OCP of the form

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \boldsymbol{\Phi}(\cdot) = \alpha_1 \cdot (d_1 + d_2 + d_3) - x_{32}(1) \tag{8.5a}$$

$$\text{s.t.} \quad \dot{x}_{\mathrm{j}0}(t) = d_{\mathrm{j}} x_{\mathrm{j}1}(t), \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2,3, \tag{8.5b}$$

$$\dot{x}_{\mathrm{j}1}(t) = d_{\mathrm{j}} \left(u_{\mathrm{j}0}(t) - p_0 {x_{\mathrm{j}1}}^2(t)\right) / \left(x_{\mathrm{j}2}(t) + \gamma\right), \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2,3, \tag{8.5c}$$

$$\dot{x}_{\mathrm{j}2}(t) = -d_{\mathrm{j}} \varrho {u_{\mathrm{j}0}}^2(t), \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2,3, \tag{8.5d}$$

$$\boldsymbol{x}_{\mathrm{j}}(1) = \boldsymbol{x}_{\mathrm{j}+1}(0), \qquad \mathrm{j} = 1,2 \tag{8.5e}$$

$$x_{10}(0) = 0, x_{11}(0) = 0, x_{12}(0) = 1, \tag{8.5f}$$

$$x_{30}(1) = 10, x_{31}(1) = 1, \tag{8.5g}$$

$$-10 \le x_{\mathrm{j}0}(t) \le 10, \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2,3, \tag{8.5h}$$

$$-10 \le x_{\mathrm{j}1}(t) \le 10, \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2,3, \tag{8.5i}$$

$$0 \le x_{\mathrm{j}2}(t) \le 10, \qquad t \in \mathcal{T}^{\mathrm{j}}, \mathrm{j} = 1,2,3, \tag{8.5j}$$

$$u_{10}(t) = 1, \qquad t \in \mathcal{T}^1, \tag{8.5k}$$

$$-1 \le u_{20}(t) \le 1, \qquad t \in \mathcal{T}^2, \tag{8.5l}$$

$$u_{30}(t) = -1, \qquad t \in \mathcal{T}^3, \tag{8.5m}$$

$$0 \le \boldsymbol{d}, \tag{8.5n}$$

with the quantities defined in Table 8.1 extended by a stage subscript, $\mathrm{j} = 1,2,3$, for the differential states and controls. Similar to OCPs (8.1) and (8.3) we perform a time transformation on all three stages onto fixed time horizons $\mathcal{T}^{\mathrm{j}} = [0,1], \mathrm{j} = 1,2,3$, such that stage duration parameters $\boldsymbol{d} = \begin{pmatrix} d_1 & d_2 & d_3 \end{pmatrix}^T$ are added to the optimization variables in OCP (8.5).

Analogous to the two cases before, with simulated measurements from the solution of (8.5) with the chosen objective weight $\alpha_1$ and model parameter $p_0$, we set up a three-stage Bilevel Inverse OCP, where the upper level

PE is constrained by the lower level OCP (8.5) for case 3 as follows

$$\min_{\substack{\alpha_1, p_0, \\ \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}}} \quad \boldsymbol{\Psi}(\cdot) = \frac{1}{2} \sum_{\mathrm{j}=1}^{3} \sum_{\mathrm{n}=0}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1} \sum_{\mathrm{k}=0}^{2} \frac{\left(x_{\mathrm{jk}}(t_{\mathrm{jn}}^{\mathrm{m}}) - \eta_{\mathrm{jnk}}\right)^2}{\sigma_{\mathrm{k}}^2} \tag{8.6a}$$

$$\text{s.t.} \quad (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \text{ solve OCP (8.5)}, \tag{8.6b}$$

$$0 \leq \alpha_1 \leq 10, \tag{8.6c}$$

$$-20 \leq p_0 \leq 20. \tag{8.6d}$$

Similar to the previous cases, in the resulting NLP (4.13) of the DISIMFAS all bounds on the NLP variables related to the bound constraints (8.5h)-(8.5j) and (8.5l) are neglected. However, in the solution of Bilevel Inverse OCP (8.6) these bound constraints are still satisfied. To incorporate the active upper and lower bounds of the control function in the solution of OCP (8.1) for case 3, the control function is fixed to its upper bound 1 in (8.5k) in the first model stage and fixed to its lower bound −1 in (8.5m) in the last model stage.

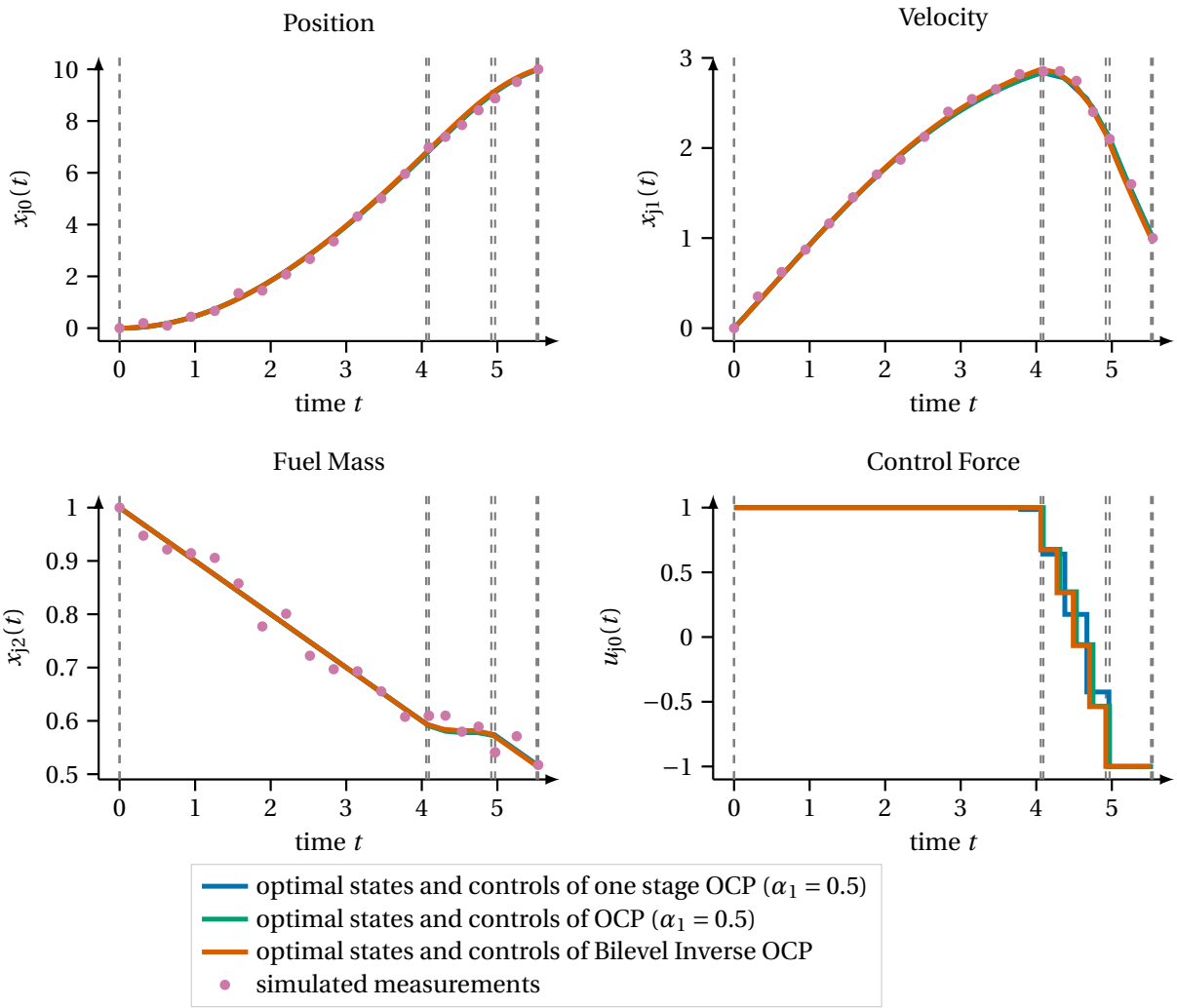


**Figure 8.5:** This figure depicts the optimal differential states and controls in the solution of Bilevel Inverse OCP (8.6) for case 3 computed with the DISIMFAS in PARDYNOPT with 14 multiple shooting nodes in the first model stage, 5 in the second and 3 in the last. This result is compared to the corresponding OCP (8.5) and the single-stage OCP (8.1) solutions with PARDYNOPT by applying the Direct Multiple Shooting Method. Because the solutions are very similar the lines mostly overlap in the illustration. With the OCP solution pseudo-measurements are generated as described at the beginning of subsection 8.1.2. They are illustrated in the plots by dots.

The DISIMFAS converges after 11 iterations with an objective value of 8.6234 and stage durations $d_1^* = 4.0588$, $d_2^* = 8.6122 \cdot 10^{-1}$ and $d_3^* = 5.9930 \cdot 10^{-1}$. With a relative error of 1.48% for objective weight $\alpha_1^* = 5.0738 \cdot 10^{-1}$ and 3.87% for model parameter $p_0^* = 9.6128 \cdot 10^{-2}$, both quantities are reproduced with a relative error of less than 5%. In Figure 8.5 the optimal differential states and control parameters in the solution of (8.6) calculated with PARDYNOPT are illustrated and compared to the three-stage OCP (8.5) solution and the single-stage OCP (8.1) for the third case. In Table 8.7 the results achieved with small constraint violation and infeasibility measure, see IPOPT [169], are summarized.

**Table 8.7:** Computational result of Bilevel Inverse OCP (8.6) for case 3 with our DISIMFAS in PARDYNOPT with settings summarized in Table 8.4 and comparison of the estimates $\alpha_1^*$ and $p_0^*$ to the true values and the phase durations $d_1^*$,$d_2^*$ and $d_3^*$ to the corresponding OCP (8.5) solution.

| case 3 | Bilevel Inverse OCP (8.6) | |
|---|---|---|
| objective $\Psi^*$ | 8.6234 | |
| constraint violation | $3.36 \cdot 10^{-12}$ | |
| dual infeasibility | $1.28 \cdot 10^{-9}$ | |
| # iterations | 11 | |
| | | |
| parameters | estimate (rel. Err%) | true value |
| $\alpha_1^*$ | $5.0738 \cdot 10^{-1}$ (1.48%) | 0.5 |
| $p_0^*$ | $9.6128 \cdot 10^{-2}$ (3.87%) | 0.1 |
| | | |
| stage durations | | OCP (8.5) |
| $d_1^*$ | 4.0588 | 4.0935 |
| $d_2^*$ | $8.6122 \cdot 10^{-1}$ | $8.7662 \cdot 10^{-1}$ |
| $d_3^*$ | $5.9930 \cdot 10^{-1}$ | $5.7127 \cdot 10^{-1}$ |

### 8.1.3 Summary

In this section we considered a case study on the rocket car example for assessing the applicability and performance of our DISIMFAS. For this particular example, we exploited the structure of the control function and integrated this information in different Bilevel Inverse OCP formulations. Hence, the DISIMFAS could be applied successfully without further estimation of the active set in the solution of the resulting NLP. For cases, where the optimal active set is not known, it can be identified with our proposed squential solution approach in section 4.4. In all three cases objective weight $\alpha_1$ and model parameter $p_0$ are estimated with a relative error of less than 10%, for the last case with even less than 5%. Summarized, all estimates, as well as all other unknowns including the state trajectories, are very close to the true values. With our DISIMFAS the results for all Bilevel Inverse OCPs could be achieved within less than 20 iterations with a convergence tolerance set to $10^{-6}$ in IPOPT.

## 8.2 Case Study: Polar Robot and a Comparison of ParDynOpt with ParaOCP

We consider a polar robot example modeled as a rigid multibody system with three DOFs. It consists of an extendable gripper arm linked to a fixed tripod by a sleeve in such a way that the arm can rotate along vertical and horizontal axes as illustrated in Figure 8.6. The gripper with its load at the tip of the robot arm is considered as point mass. In total, the multibody system comprises three rigid bodies and one point mass.

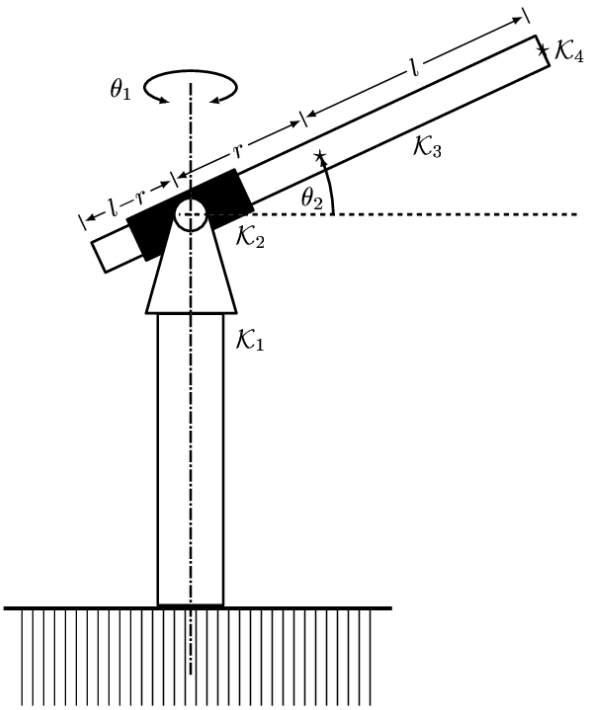


**Figure 8.6:** Polar robot example with three DOFs, ©M. Steinbach, [164].

The dynamics of the polar robot are deduced in the diploma thesis of Steinbach in [164]. With the equations (8.7) and the quantities defined in Table 8.8

$$a_0(r) := I_1^{23} - I_2^{23} + m_b r^2 + m_l(\ell + r)^2 \tag{8.7a}$$

$$s(r) := 2\left((m_b + m_l)r + m_l\ell\right) \tag{8.7b}$$

they can be formulated as follows:

$$\ddot{\theta}_1 = \frac{\tau_1 - s(r)\dot{r}\dot{\theta}_1\cos^2\theta_2 + 2a_0(r)\dot{\theta}_1\dot{\theta}_2\sin\theta_2\cos\theta_2}{I_3^1 + I_2^{23} + a_0(r)\cos^2\theta_2} \tag{8.8a}$$

$$\ddot{\theta}_2 = \frac{\tau_2 - s(r)\dot{r}\dot{\theta}_2 - a_0(r)\dot{\theta}_1^2\sin\theta_2\cos\theta_2 - 0.5s(r)g\cos\theta_2}{a_0(r)} \tag{8.8b}$$

$$\ddot{r} = \frac{\tau_3 + 0.5s(r)(\dot{\theta}_1^2\cos^2\theta_2 + \dot{\theta}_2^2) - (m_b + m_l)g\sin\theta_2}{m_b + m_l}. \tag{8.8c}$$

For a compact representation, the time dependence of the quantities is omitted in (8.8).

**Table 8.8:** Definition of quantities which appear in the equations of motion (8.12) for the polar robot example.

| Symbol | Description | Value | |
|---|---|---|---|
| $\theta_1$ | angle around horizontal axis | $\theta_1(0) = 0,$ | $\theta_1(1) = 0$ |
| $\theta_2$ | angle around vertical axis | $\theta_2(0) = 0,$ | $\theta_2(1) = 0$ |
| $r$ | distance robot arm center - rotation center | $r(0) = 0.7,$ | $r(1) = -0.1$ |
| $\dot{\theta}_1$ | horizontal angular velocity | $\dot{\theta}_1(0) = 0,$ | $\dot{\theta}_1(1) = 0$ |
| $\dot{\theta}_2$ | vertical angular velocity | $\dot{\theta}_2(0) = 0,$ | $\dot{\theta}_2(1) = 0$ |
| $\dot{r}$ | rotational velocity | $\dot{r}(0) = 0,$ | $\dot{r}(1) = 0$ |
| $\tau_1$ | torque around horizontal axis | | |
| $\tau_2$ | torque around vertical axis | | |
| $\tau_3$ | force along robot arm | | |
| $d_1$ | duration of process | | |
| $\ell$ | half length of robot arm | 0.75 | |
| $m_b$ | mass of robot arm | 40 | |
| $m_l$ | mass of load | 10 | |
| $I_1^{23}$ | inertia of robot arm and sleeve around axis 1 | 18.5 | |
| $I_2^{23}$ | inertia of robot arm and sleeve around axis 2 | 0.12 | |
| $I_3^{23}$ | inertia of robot arm and sleeve around axis 3 | 18.5 | |
| $I_3^1$ | inertia of tripod around axis 3 | 10 | |
| $g$ | gravitational acceleration | 9.81 | |

### 8.2.1 An OCP for the Polar Robot Example

We now consider an optimal movement of the robot arm from one position to another with initial and final positions as listed in Table 8.8. We define 6 differential states and 3 controls as

$$\boldsymbol{x}(t) := \begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{pmatrix} = \begin{pmatrix} \theta_1(t) \\ \theta_2(t) \\ r(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \dot{r}(t) \end{pmatrix} \quad \text{and} \quad \boldsymbol{u}(t) := \begin{pmatrix} u_0(t) \\ u_1(t) \\ u_2(t) \end{pmatrix} = \begin{pmatrix} \tau_1(t) \\ \tau_2(t) \\ \tau_3(t) \end{pmatrix}. \tag{8.9}$$

and choose the model parameter to be the half length of the robot arm, $p_0 = \ell$. With the equations of motion given by (8.8) and duration parameter, $d_1$, which originates from a time transformation of the process onto a fixed time horizon, $\mathcal{T} = [0,1]$, the differential equation can be summarized in

$$\dot{\boldsymbol{x}}(t) = d_1 \cdot \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), p_0).$$

For finding trajectories which minimize a weighted sum of two selected criteria, the final time of the process, $d_1$, and the retraction energy, $\int_0^1 u_2(t)^2 \mathrm{d}t$, the following OCP for the polar robot example can be formulated

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \boldsymbol{\Phi}(\cdot) = \alpha_1 \cdot d_1 + \alpha_2 \cdot \int_0^1 u_2(t)^2 \mathrm{d}t \tag{8.10a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = d_1 \cdot \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), p_0), \qquad t \in \mathcal{T}, \tag{8.10b}$$

$$\boldsymbol{x}(0) = \begin{pmatrix} 0 \\ 0 \\ 0.7 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \ \boldsymbol{x}(1) = \begin{pmatrix} 0 \\ 0 \\ -0.1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{8.10c}$$

$$\begin{pmatrix} -1 \\ -1 \\ -0.5 \\ -5 \\ -5 \\ -6 \end{pmatrix} \leq \boldsymbol{x}(t) \leq \begin{pmatrix} 1 \\ 1 \\ 0.7 \\ 5 \\ 5 \\ 6 \end{pmatrix}, \qquad t \in \mathcal{T}, \tag{8.10d}$$

$$\begin{pmatrix} -1000 \\ -1500 \\ -1000 \end{pmatrix} \leq \boldsymbol{u}(t) \leq \begin{pmatrix} 1000 \\ 1500 \\ 1000 \end{pmatrix}, \qquad t \in \mathcal{T}, \tag{8.10e}$$

$$0 \leq d_1, \tag{8.10f}$$

under consideration of initial and final conditions, (8.10c), and bounds for states, controls and duration parameter, (8.10d)-(8.10f). This OCP formulation is similar to the one stated in the work of Hatz [80] but with slightly different dynamics for the polar robot. However, the derivation of the equations of motion used in [80] are not revealed such that we decided to use the dynamics by Steinbach [164] in a first case study in subsection 8.2.3. Here, our goal is to investigate how structurally different initial guesses impact the performance of our DISIMFAS implemented in PARDYNOPT.

For a direct comparison of our DISIMFAS to the Direct All-at-Once Approach implemented in PARAOCP, we choose the identical setting as in section 14.5 in the thesis of Hatz [80] and treat the generation of measurements and initial guesses in exactly the same way as described there. Hence, in our second case study in subsection 8.2.3, the dynamics of the polar robot are used as introduced in the work of Hatz.

### 8.2.2 Numerical Set-Up for Case Studies A and B

For both case studies the arising OCPs (8.10) are solved with the software package PARDYNOPT and its implementation of the Direct Multiple Shooting Method [26] as described in subsection 2.2.1 with an equidistant multiple shooting grid with 7 nodes and constant control approximations on each multiple shooting interval. Furthermore, for the arising initial value problems SOLVIND is used with an adaptive BDF method DAESOL-II [5] and its sensitivity and derivative generation with an interface to ADOL-C[171]. IPOPT [169] solves the resulting NLPs with an exact Hessian approximation and a convergence tolerance set to $10^{-6}$ (defined in [169] as error tolerance). Unless otherwise specified, the initial guesses for the state values are set to $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ at each multiple shooting node, the initial guesses for the values of all control parameters are set to 10 and the duration parameter $d_1$ to 1.

The Bilevel Inverse OCPs in both case studies are solved with our DISIMFAS implemented in PARDYNOPT. Here, the same settings as above are chosen for solving the arising initial value problems, sensitivity and derivative generations on the multiple shooting grid. The resulting NLPs of the form (4.13) are then solved with IPOPT [169] with a convergence tolerance set to $10^{-6}$. For the constraint Jacobian we choose the structure exploiting implementation in PARDYNOPT and for the Hessian of the Lagrangian a limited-memory quasi-NEWTON method (L-BFGS update formula) in IPOPT. The PARDYNOPT setting for solving OCPs and Bilevel Inverse OCPs is summarized in Table 8.9.

Pseudo-measurements are generated from solutions $\boldsymbol{x}^{*\mathrm{OCP}}$ of the corresponding lower level OCPs (8.10) by adding normally distributed noise $\boldsymbol{\varepsilon}$ with zero mean and a standard deviation $\boldsymbol{\sigma}$ with values $\sigma_{\mathrm{k}} = 0.05 \cdot 0.2$, $\mathrm{k} = 0, \ldots, 2$, for the angular and radial states and $\sigma_k = 0.05 \cdot 1, \mathrm{k} = 3, \ldots, 5,$ for the corresponding velocities:

$$\eta_{\mathrm{nk}} = x_{\mathrm{k}}^{*\mathrm{OCP}}(t_{\mathrm{n}}^{\mathrm{m}}) + \varepsilon_{\mathrm{nk}}, \quad \mathrm{n} = 0, \ldots, \mathrm{n_m} - 1,\ \mathrm{k} = 0, \ldots, 5.$$

The measurement times $t_{\mathrm{n}}^{\mathrm{m}}$ are chosen to be the same as the multiple shooting gid.

For solving the Bilevel Inverse OCPs in the two case studies the initial guesses are also solutions of OCPs achieved with PARDYNOPT and the setting summarized in Table 8.9, but with varying objective weight $\alpha_2$ and model parameter $p_0$.

**Table 8.9:** Setting used in PARDYNOPT for the two case studies in subsection 8.2.3 and subsection 8.2.4 to solve OCPs of the form (8.10) and Bilevel Inverse OCPs of the form (8.11).

| | OCP | Bilevel Inverse OCP |
|---|---|---|
| solution method | Direct Multiple Shooting Method | Direct Simultaneous Approach with Fixed Active Set |
| # multiple shooting nodes | (7) | (7) |
| NLP solver | IPOPT | IPOPT |
| convergence tolerance | $10^{-6}$ | $10^{-6}$ |
| constraint Jacobian | sparse | sparse |
| Hessian | exact | BFGS update |
| integrator and derivatives | SOLVIND | SOLVIND |

### 8.2.3 Case Study A: Performance of DiSimFAS with Structurally Different Initial Guesses

Although, in our first case study we use slightly different dynamics for the polar robot as in [80], the weights and model parameters in OCP (8.10) are chosen to be the same as in section 13.4 in the thesis of Hatz for the generation of measurements and initial guesses for solving a Bilevel Inverse OCP. This setting leads to significant changes in the structure. In the work of Hatz the Lifting Approach has to be invoked because the Direct All-at-Once Approach without an appropriate treatment of the complementarity constraints did not converge.

Now we investigate the performance of our DISIMFAS described in chapter 4 with regard to structurally different initial guesses for solving Bilevel Inverse OCPs. We start by considering OCP (8.10) and set the objective weights $\alpha_1$ and $\alpha_2$ to the fixed values $0.5 \cdot 10^4$ and 0.5, respectively, and the model parameter $p_0$ to 0.75, which describes the half length of the robot arm $\ell$. This OCP is now solved with the Direct Multiple Shooting Method as described in the previous subsection and the PARDYNOPT settings summarized in Table 8.9. Its result is labeled with OCP 1. By choosing different initial guesses for state and control parameter values, while considering the solution OCP 1, we find two more local solutions of the same OCP (8.10), labeled as OCP 2 and OCP 3, respectively. The results of the three solutions achieved with small constraint violation and infeasibility measure, see IPOPT [169], are summarized in Table 8.10 and illustrated in Figure 8.7 and Figure 8.8.

**Table 8.10:** Computational results of OCP (8.10) for the polar robot example with dynamics by Steinbach [164] achieved with the Direct Multiple Shooting Method implemented in PARDYNOPT.

| | OCP 1 | OCP 2 | OCP 3 |
|---|---|---|---|
| objective $\mathbf{\Phi}^*$ | $6.3918 \cdot 10^{-1}$ | $6.4228 \cdot 10^{-1}$ | $6.3918 \cdot 10^{-1}$ |
| constraint violation | $9.37 \cdot 10^{-9}$ | $1.17 \cdot 10^{-6}$ | $7.62 \cdot 10^{-12}$ |
| dual infeasibility | $1.07 \cdot 10^{-12}$ | $1.14 \cdot 10^{-9}$ | $1.57 \cdot 10^{-13}$ |
| # iterations | 312 | 147 | 141 |
| $d_1^*$ | 1.2234 | 1.2283 | 1.2234 |

The solution of OCP 3 is the symmetric counterpart of OCP 1 with the same objective value $\mathbf{\Phi}^* = 6.3918 \cdot 10^{-1}$ and duration parameter $d_1^* = 1.2234$. The difference originates from the varying initial deflection of the robot arm around the vertical axis, $\theta_1$. In contrast to this, in the solution of OCP 2, with slightly higher objective

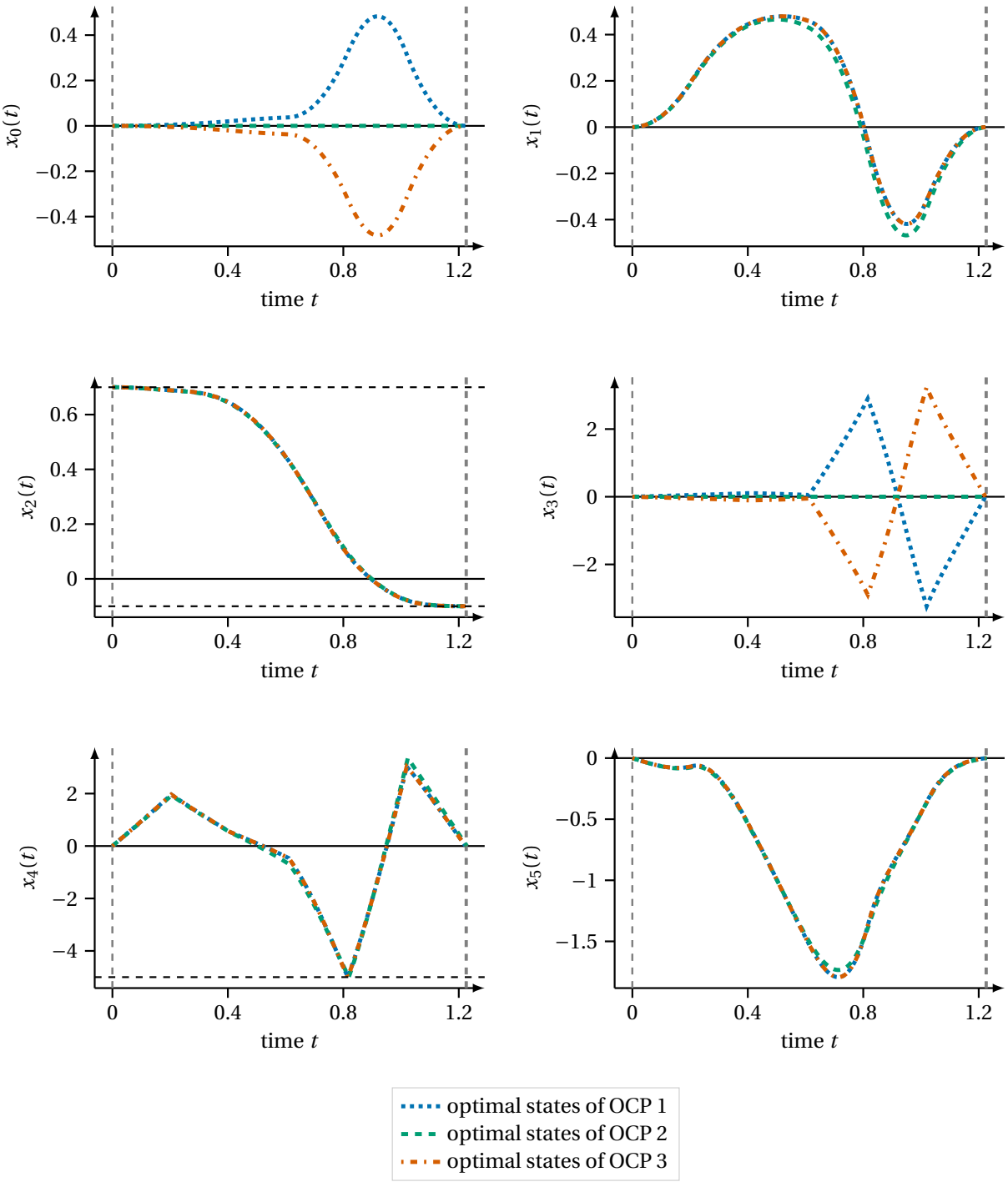


**Figure 8.7:** Optimal differential states in three local solutions of OCP (8.10) with dynamics by Steinbach [164]. All results are computed with ParDynOpt by applying the Direct Multiple Shooting Method with 7 shooting nodes.

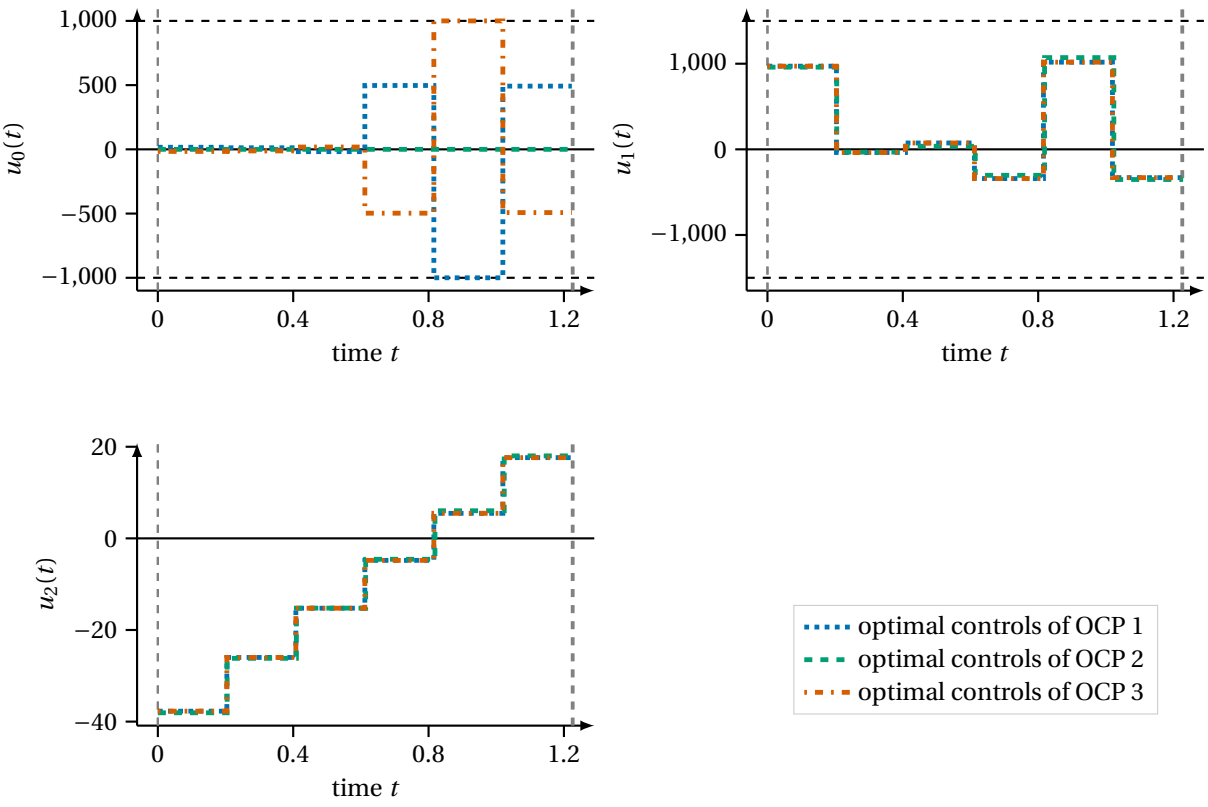


**Figure 8.8:** Optimal controls in three local solutions of OCP (8.10) with dynamics by Steinbach [164]. All results are computed with ParDynOpt by applying the Direct Multiple Shooting Method with 7 shooting nodes.

$\mathbf{\Phi}^* = 6.4228 \cdot 10^{-1}$ and duration parameter $d_1^* = 1.2283$ there is no such deflection at all. Hence, the final position of the robot arm is achieved solely by moving the robot arm around the horizontal axis. As illustrated in Figure 8.7 in all three solutions the lower bound of the NLP variable which corresponds to angular velocity $\dot{\theta}_1$ at shooting node 5 is active. In addition to this, for OCP 1 the lower bound and for OCP 3 the upper bound of the control parameter at shooting node 5 corresponding to torque $\tau_0$ is active, see Figure 8.8.

Because of symmetric solutions, in the following we only consider OCP 1 and OCP 2 and ignore OCP 3. Hence, with the results OCP 1 and OCP 2, measurements are generated as described in subsection 8.2.2 and with the Bilevel Inverse OCP stated as

$$\min_{\substack{\alpha_2, p_0, \\ \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}}} \quad \mathbf{\Psi}(\cdot) = \frac{1}{2} \sum_{n=0}^{n_m - 1} \sum_{k=0}^{2} \frac{\left(x_k(t_n^m) - \eta_{nk}\right)^2}{\sigma_k^2} \tag{8.11a}$$

$$\text{s.t.} \quad (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \text{ solve OCP (8.10)}, \tag{8.11b}$$

$$\alpha_1 = \alpha_1' \cdot 10^4, \tag{8.11c}$$

$$0 \leq \alpha_2, \tag{8.11d}$$

$$0 \leq p_0, \tag{8.11e}$$

for both measurement sets, model parameter $p_0$ and objective weight $\alpha_2$ are determined while $\alpha_1 = \alpha_1' \cdot 10^4$ is fixed to its value with $\alpha_1' = 0.5$.

To apply our DiSimFAS described in chapter 4, only equality constraints of the discretized lower level OCP and bounds on the decision variables related to the fixed active set are considered in the resulting NLP (4.13). This implicates the necessity to identify the bounds of the NLP variables related to the constraints (8.10d)-(8.10f) that are active in the OCP solution. So far, in the software package ParDynOpt the corresponding working set $\mathcal{W}$ as a guess of the active set $\mathcal{A}^*$ has to be specified. In section 4.4 an outlook is given how the optimal

active set $\mathcal{A}^*$ can be identified in the future. With a given active set all calculations with PARDYNOPT are performed with the set-up as described in subsection 8.2.2 and summarized in Table 8.9. As initial guesses we choose two different but again symmetrical solutions of an OCP with objective weights $\alpha_1 = 0.5 \cdot 10^4, \alpha_2 = 0.2$ and model parameter $p_0 = 0.6$. Both solutions have a significantly changed structure compared to the OCP solutions in Figure 8.7 and Figure 8.8 and are labeled with $\text{OCP}^{\text{guess}}1$ and $\text{OCP}^{\text{guess}}2$. To investigate the performance of our DISIMFAS with regard to structural different initial guesses, we analyze the solutions of Bilevel Inverse OCP (8.11) for the two measurement sets related to the OCPs 1 and OCPs 2 together with the two OCPs solutions for the initial guesses. In total, we end up with 4 cases as summarized in Table 8.11. In Figure 8.9 - 8.12 the optimal trajectories and controls of the solutions of the Bilevel Inverse OCPs (8.11) for each case are illustrated. The results achieved with small constraint violation and infeasibility measure, see IPOPT [169], are summarized in Table 8.12 and Table 8.13.

**Table 8.11:** Selected cases to solve Bilevel Inverse OCPs for the polar robot example with dynamics by Steinbach [164].

| | Measurements | Initial Guess | Bilevel Inverse OCP |
|---|---|---|---|
| case 1 | OCP 1 | $\text{OCP}^{\text{guess}}1$ | Bilevel Inverse OCP 1 |
| case 2 | OCP 1 | $\text{OCP}^{\text{guess}}2$ | Bilevel Inverse OCP 2 |
| case 3 | OCP 2 | $\text{OCP}^{\text{guess}}1$ | Bilevel Inverse OCP 3 |
| case 4 | OCP 2 | $\text{OCP}^{\text{guess}}2$ | Bilevel Inverse OCP 4 |

For case 1 the DISIMFAS converges after 15 iterations with an objective value of 3.4216 and a stage duration of 1.2259. For case 2 it converges after 18 iterations to the same solution. With a relative error of 2.36% for objective weight $\alpha_2^* = 5.1182 \cdot 10^{-1}$ and 2.59% for model parameter $p_0^* = 7.3058 \cdot 10^{-1}$, both quantities are reproduced with a relative error of less than 3%.

**Table 8.12:** Computational results of Bilevel Inverse OCPs (8.11) for case 1 and case 2 with dynamics by Steinbach [164] with our DISIMFAS in PARDYNOPT with settings summarized in Table 8.9 and comparison of the estimates $\alpha_2^*$ and $p_0^*$ to the true values and the phase duration $d_1^*$ to the corresponding OCP 1 solution.

| | case 1 | case 2 | | |
|---|---|---|---|---|
| objective $\Psi^*$ | 3.4216 | 3.4216 | | |
| constraint violation | $1.27 \cdot 10^{-8}$ | $1.42 \cdot 10^{-7}$ | | |
| dual infeasibility | $1.15 \cdot 10^{-7}$ | $3.32 \cdot 10^{-8}$ | | |
| # iterations | 15 | 18 | | |
| parameters | estimate (rel. Err%) | estimate (rel. Err%) | true value | initial guess |
| $\alpha_2^*$ | $5.1182 \cdot 10^{-1}$ (2.36%) | $5.1182 \cdot 10^{-1}$ (2.36%) | 0.5 | 0.2 |
| $p_0^*$ | $7.3058 \cdot 10^{-1}$ (2.59%) | $7.3058 \cdot 10^{-1}$ (2.59%) | 0.75 | 0.6 |
| stage duration | | | OCP 1 | |
| $d_1^*$ | 1.2259 | 1.2259 | 1.2234 | |

For case 3 the DISIMFAS converges after 17 iterations with an objective value of 3.9488 and a stage duration of 1.2259. For case 4 it converges after 17 iterations to the same solution. With a relative error of 1.23% for objective weight $\alpha_2^* = 5.0614 \cdot 10^{-1}$ and 2.93% for model parameter $p_0^* = 7.2801 \cdot 10^{-1}$, both quantities are reproduced with a relative error of less than 3%.

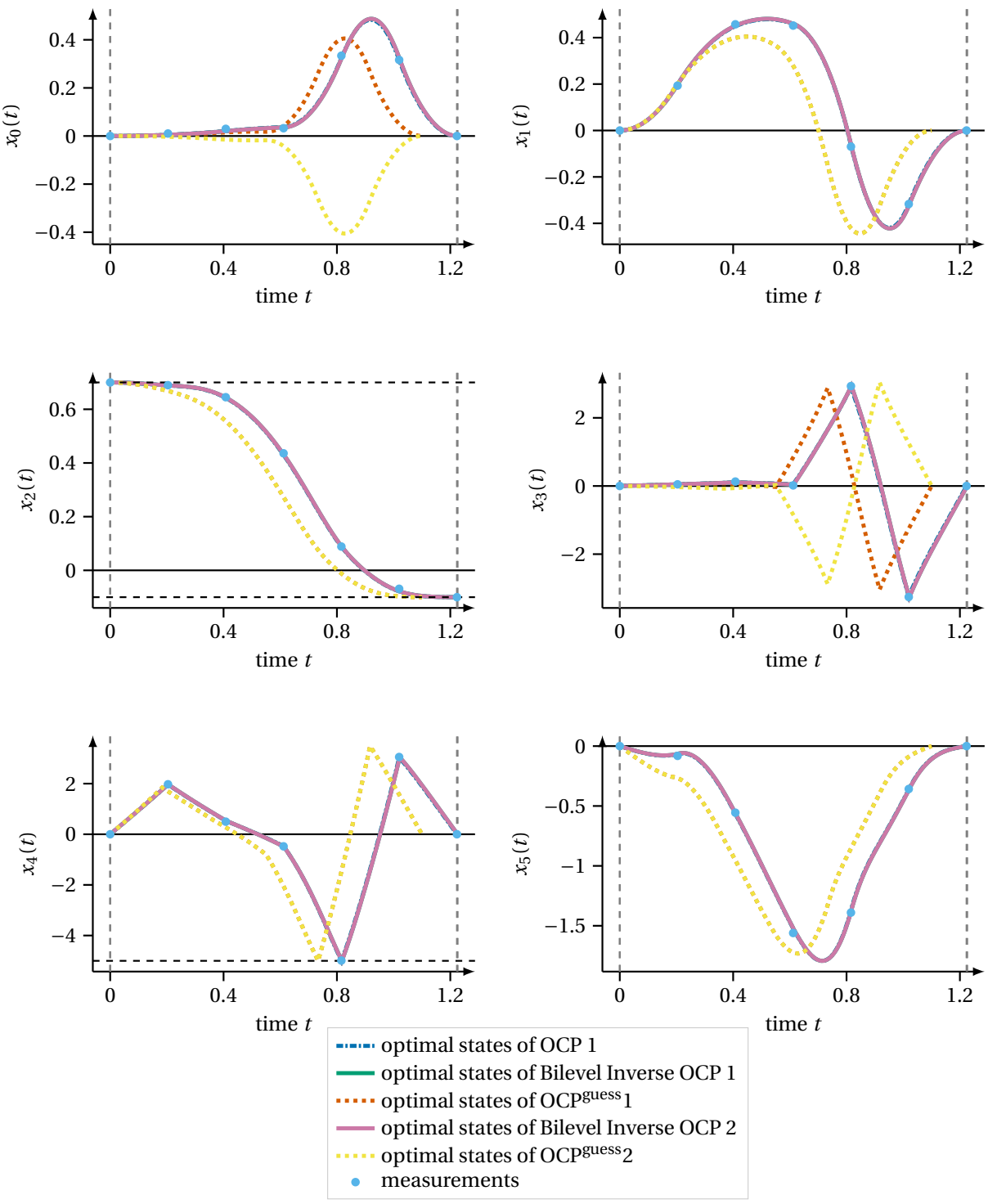


**Figure 8.9:** This figure depicts the optimal differential states in the solutions of Bilevel Inverse OCP (8.11) for case 1 and case 2 computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes. The results are compared to the corresponding OCP 1 (8.10) solution. Generated pseudo-measurements and initial guesses OCP$^{\text{guess}}$1 and OCP$^{\text{guess}}$2 are also illustrated. The dynamics by Steinbach [164] are used.

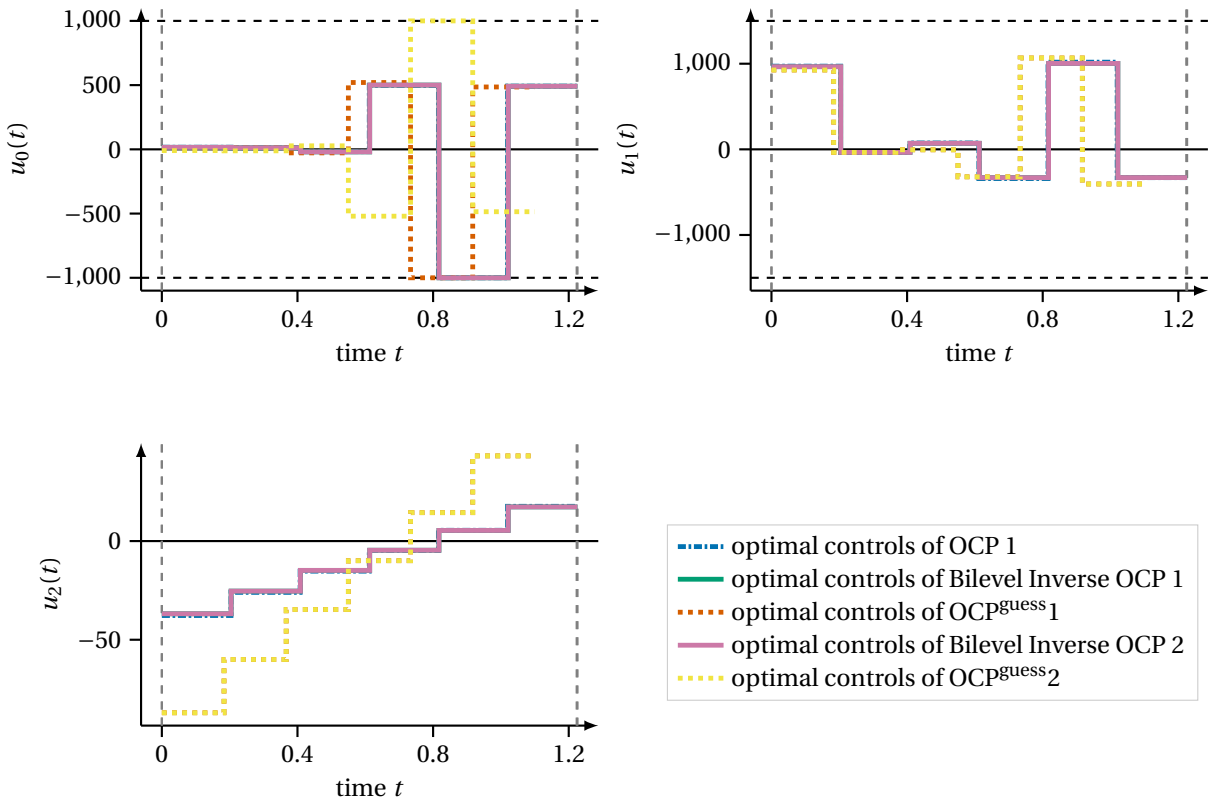


**Figure 8.10:** This figure depicts the optimal controls in the solutions of Bilevel Inverse OCP (8.11) for case 1 and case 2 computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes. The results are compared to the corresponding OCP 1 (8.10) solution. Initial guesses OCP$^{\text{guess}}$1 and OCP$^{\text{guess}}$2 are also illustrated. The dynamics by Steinbach [164] are used.

**Table 8.13:** Computational results of Bilevel Inverse OCPs (8.11) for case 3 and case 4 with dynamics by Steinbach [164] with our DISIMFAS in PARDYNOPT with settings summarized in Table 8.9 and comparison of the estimates $\alpha_2^*$ and $p_0^*$ to the true values and the phase duration $d_1^*$ to the corresponding OCP 2 solution.

| | case 3 | case 4 | | |
|---|---|---|---|---|
| objective $\Psi^*$ | 3.9488 | 3.9488 | | |
| constraint violation | $7.86 \cdot 10^{-7}$ | $9.64 \cdot 10^{-9}$ | | |
| dual infeasibility | $6.90 \cdot 10^{-5}$ | $6.47 \cdot 10^{-8}$ | | |
| # iterations | 17 | 17 | | |
| parameters | estimate (rel. Err%) | estimate (rel. Err%) | true value | initial guess |
| $\alpha_2^*$ | $5.0614 \cdot 10^{-1}$ (1.23%) | $5.0614 \cdot 10^{-1}$ (1.23%) | 0.5 | 0.2 |
| $p_0^*$ | $7.2801 \cdot 10^{-1}$ (2.93%) | $7.2801 \cdot 10^{-1}$ (2.93%) | 0.75 | 0.6 |
| stage duration | | | OCP 2 | |
| $d_1^*$ | 1.2295 | 1.2295 | 1.2283 | |

In sum, with our DISIMFAS, for all cases from Table 8.11, objective weight $\alpha_2$ and model parameter $p_0$ could be determined within less than 20 iterations and a relative error of less than 3% regardless of the distict structure of the initial guesses. Additionally, all solutions of the Bilevel Inverse OCPs achieved by applying our DISIMFAS satisfy the bound constraints, although not considered during the solution procedure.

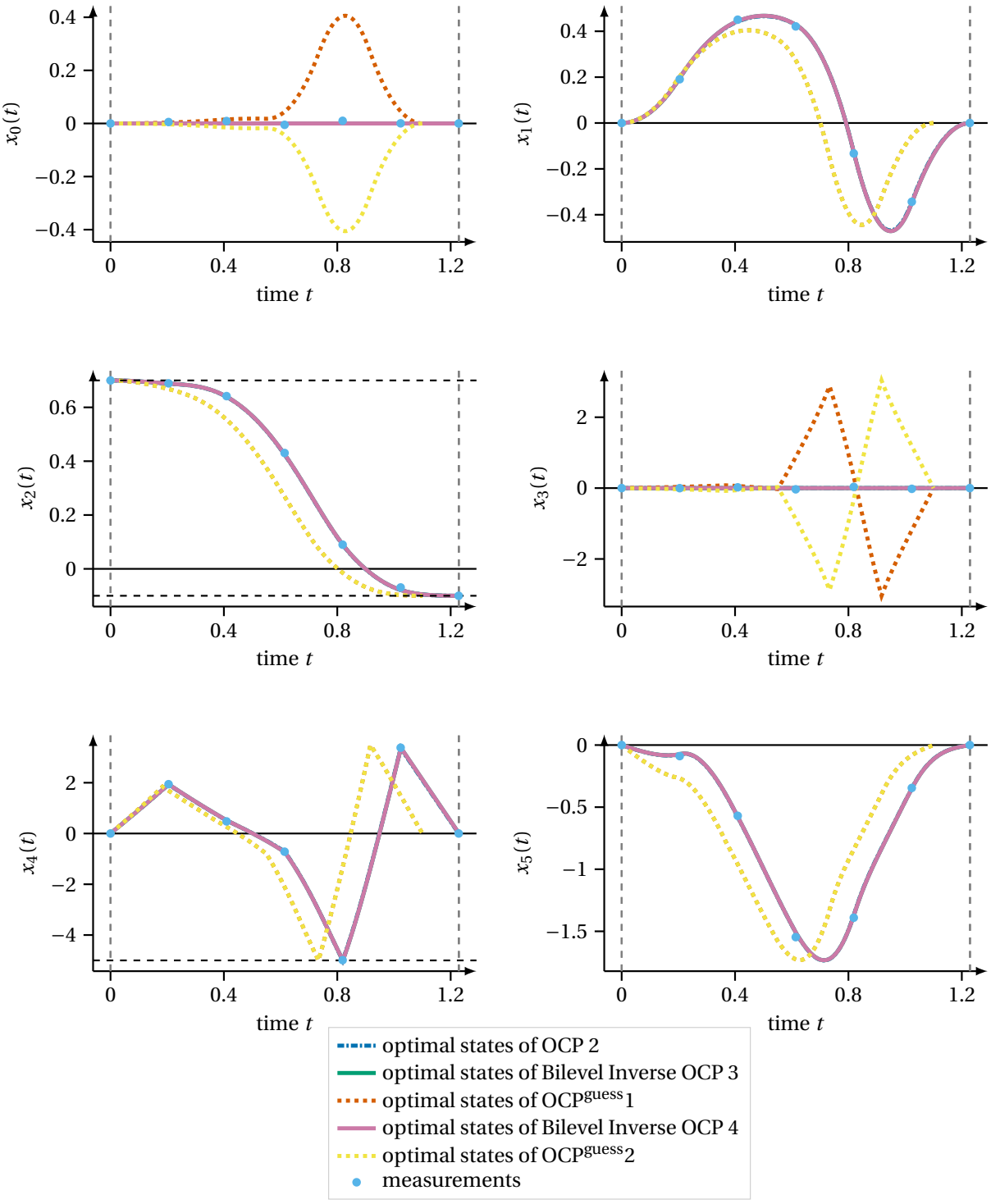


**Figure 8.11:** This figure depicts the optimal differential states in the solutions of Bilevel Inverse OCP (8.11) for case 3 and case 4 computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes. The results are compared to the corresponding OCP 2 (8.10) solution. Generated pseudo-measurements and initial guesses OCP$^{\text{guess}}$1 and OCP$^{\text{guess}}$2 are also illustrated. The dynamics by Steinbach [164] are used.

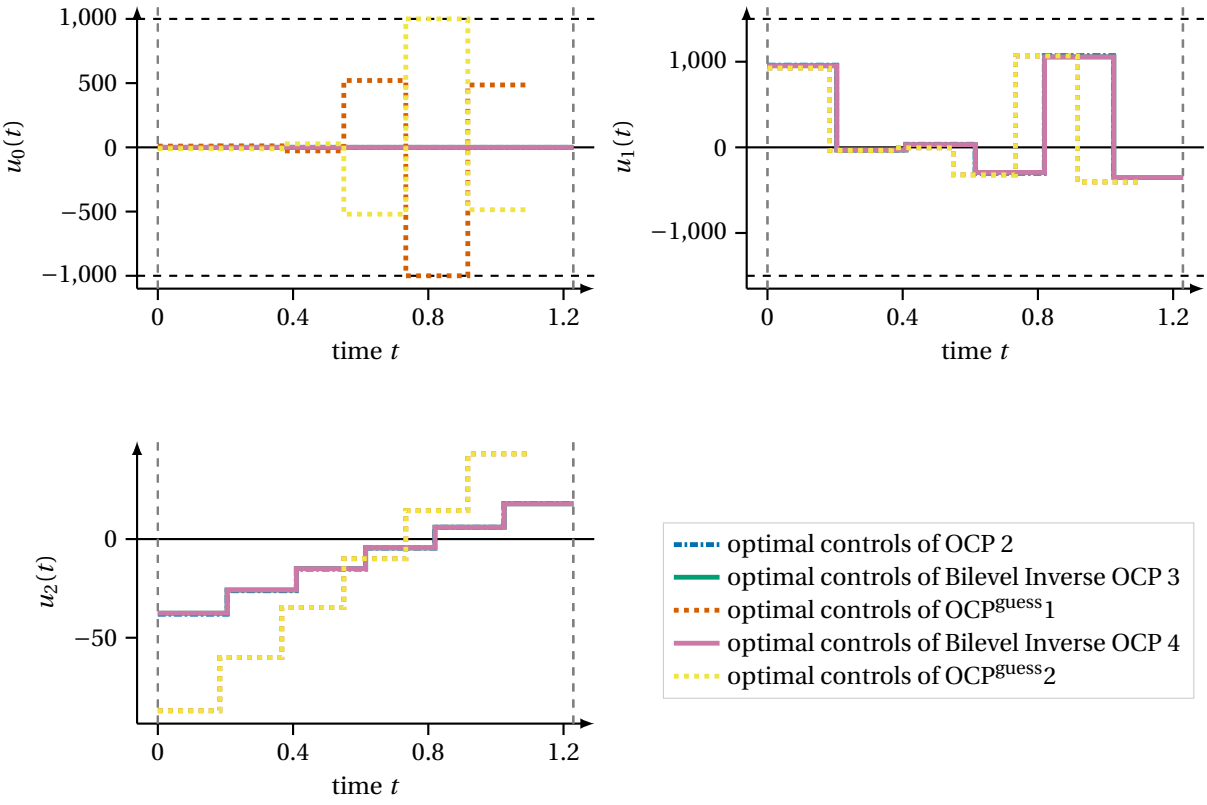


**Figure 8.12:** This figure depicts the optimal controls in the solutions of Bilevel Inverse OCP (8.11) for case 3 and case 4 computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes. The results are compared to the corresponding OCP 2 (8.10) solution. Initial guesses OCP$^{\text{guess}}$1 and OCP$^{\text{guess}}$2 are also illustrated. The dynamics by Steinbach [164] are used.

### 8.2.4 Case Study B: Comparison of DiSimFAS in PARDYNOPT with Direct All-at-Once Approach in PARAOCP

For a comparison of our DISIMFAS with the Direct All-at-Once Approach described in the work of Hatz [80] we start with a reformulation of the dynamics (8.8) deduced in the work of Steinbach [164] and get

$$\ddot{\theta}_1 = \frac{\tau_1 + \dot{\theta}_1 \cdot \left(\sin(2\theta_2)\dot{\theta}_2 \left(I_1^{23} - I_2^{23} + m_b r^2 + m_l(\ell + r)^2\right) - 2\left((m_b r + m_l(\ell + r))\dot{r}\cos^2\theta_2\right)\right)}{I_3^1 + I_2^{23}\sin^2\theta_2 + \cos^2\theta_2\left(I_1^{23} + m_b r^2 + m_l(\ell + r)^2\right)}, \tag{8.12a}$$

$$\ddot{\theta}_2 = \frac{1}{I_1^{23} - I_2^{23} + D_{\ddot{\theta}_2}} \cdot \left(\tau_2 + 0.5\sin(2\theta_2)\dot{\theta}_1^2\left(I_2^{23} - I_1^{23} - m_b r^2 - m_l(\ell + r)^2\right)\right. \tag{8.12b}$$

$$\left. - (m_b r + m_l(\ell + r))\left(g\cos\theta_2 + 2\dot{r}\dot{\theta}_2\right)\right),$$

$$\ddot{r} = \frac{\tau_3 + (m_b r + m_l(\ell + r))\left(\cos^2\theta_2\dot{\theta}_1^2 + \dot{\theta}_2^2\right) - (m_b + m_l)g\sin\theta_2}{m_b + m_l}, \tag{8.12c}$$

by defining

$$D_{\ddot{\theta}_2} := m_b r^2 + m_l(\ell + r)^2.$$

In contrast to this, the dynamics of the polar robot as introduced in the work of Hatz [80] are formulated as follows:

$$\ddot{\theta}_1 = \frac{\tau_1 + \dot{\theta}_1 \cdot \left(\sin(2\theta_2)\dot{\theta}_2 \left(I_1^{23} - I_2^{23} + m_b r^2 + m_l(\ell + r)^2\right) - 2\left(m_b r + m_l(\ell + r)\dot{r}\cos^2\theta_2\right)\right)}{I_3^1 + I_2^{23}\sin^2\theta_2 + \cos^2\theta_2\left(I_1^{23} + m_b r^2 m_l(\ell + r)^2\right)}, \tag{8.13a}$$

$$\ddot{\theta}_2 = \frac{1}{I_3^{23} + D_{\ddot{\theta}_2}} \cdot \left(\tau_2 + 0.5\sin(2\theta_2)\dot{\theta}_1^2\left(I_2^{23} - I_1^{23} - m_b r^2 - m_l(\ell + r)^2\right)\right. \tag{8.13b}$$

$$
\begin{aligned}
&\quad -(m_b r + m_l(\ell + r))\left(g\cos\theta_2 + 2\dot{r}\dot{\theta}_2\right)\Big), \\
\ddot{r} \;&= \frac{\tau_3 + (m_b r + m_l(\ell + r))\left(\cos^2\theta_2\dot{\theta}_1^2 + \dot{\theta}_2^2\right) - (m_b + m_l)g\sin\theta_2}{m_b + m_l},
\end{aligned} \tag{8.13c}
$$

and differ in (8.13a) in last term of the numerator and in the last term of the denominator, and in the denominator of (8.13b) compared to the equations (8.12a) and (8.12b) by Steinbach. However, in our second case study for a direct comparison of our DISIMFAS to results achieved in [80, section 14.4] for a Bilevel Inverse OCP of the form (8.11), we choose the dynamics from (8.13) and use the same settings as described in the work of Hatz. We produce pseudo-measurements by solving OCP (8.10) with objective weights $\alpha_1 = 1.0 \cdot 10^4$, $\alpha_2 = 2.0$ and model parameter $p_0 = 0.75$ as described in subsection 8.2.2. As initial guesses we use the solution of OCP (8.10) achieved in the same way but with varying objective weight $\alpha_2 = 2.3$ and model parameter $p_0 = 1.5$. Lagrange multipliers are also a part of the decision variables of the resulting NLP when solving a Bilevel Inverse OCP with the DISIMFAS and the Direct All-at-Once Approach. In [80] they are initialized with the solution of the OCP. However, in calculations with our method it was sufficient to initialize them with 0.

To apply our DISIMFAS described in chapter 4 only equality constraints of the discretized lower level OCP and bounds on the decision variables related to the fixed active set are considered in the resulting NLP (4.13). This implicates the necessity to identify the bounds of the NLP variables related to the constraints (8.10d)-(8.10f) that are active in the OCP solution. As illustrated in Figure 8.13 the lower bound of the NLP variable, which corresponds to angular velocity $\dot{\theta}_1$ at shooting node 5 is active. In addition to this, the lower bound of the control parameter at shooting node 5, which corresponds to torque $\tau_0$ is active, see Figure 8.14. So far in PARDYNOPT the working set $\mathcal{W}$ has to be specified by the user. In the solution of the Bilevel Inverse OCP all bound constraints are satisfied.

With the simulated measurements and the initial guesses, the Bilevel Inverse OCP (8.11) is solved with our DISIMFAS and the PARDYNOPT settings as summarized in Table 8.9. Model parameter $p_0$ and objective weight $\alpha_2$ are determined while $\alpha_1 = \alpha_1' \cdot 10^4$ is fixed to its value with $\alpha_1' = 1.0$. The optimal differential states and controls achieved by our DISIMFAS are illustrated in Figure 8.13 and Figure 8.14, respectively. The computational results are summarized in Table 8.14 and compared to results with PARAOCP taken from [80].

**Table 8.14:** Computational results of Bilevel Inverse OCP (8.11) with dynamics by Hatz [80] with our DISIMFAS in PARDYNOPT with settings summarized in Table 8.9 and its comparison to results achieved in [80] with the Direct All-at-Once Approach in PARAOCP are summarized.

| Bilevel Inverse OCP (8.11) | PARDYNOPT | PARAOCP | | |
|---|---|---|---|---|
| objective $\Psi^*$ | 3.4535 | $1.0522 \cdot 10^1$ | | |
| constraint violation | $6.67 \cdot 10^{-8}$ | (not specified in [80]) | | |
| dual infeasibility | $2.77 \cdot 10^{-9}$ | (not specified in [80]) | | |
| # iterations | 29 | 649 | | |
| | | | | |
| parameters | estimate (rel. Err%) | estimate (rel. Err%) | true value | initial guess |
| $\alpha_2^*$ | 2.1919 (9.60%) | 2.2815 (14.1%) | 2 | 2.3 |
| $p_0^*$ | $7.3055 \cdot 10^{-1}$ (2.59%) | $7.3888 \cdot 10^{-1}$ (1.48%) | 0.75 | 0.6 |
| | | | | |
| stage duration | | | | |
| $d_1^*$ | 1.2797 | 1.2698 | | |

The DISIMFAS converges after 29 iterations with an objective value of 3.4535 and a stage duration of 1.2797, whereas the Direct All-at-Once Approach takes 649 iterations to achieve a higher least-squares objective of $1.0522 \cdot 10^1$ with an estimated duration parameter of 1.2698. Hence, with the possibility to incorporate knowl-

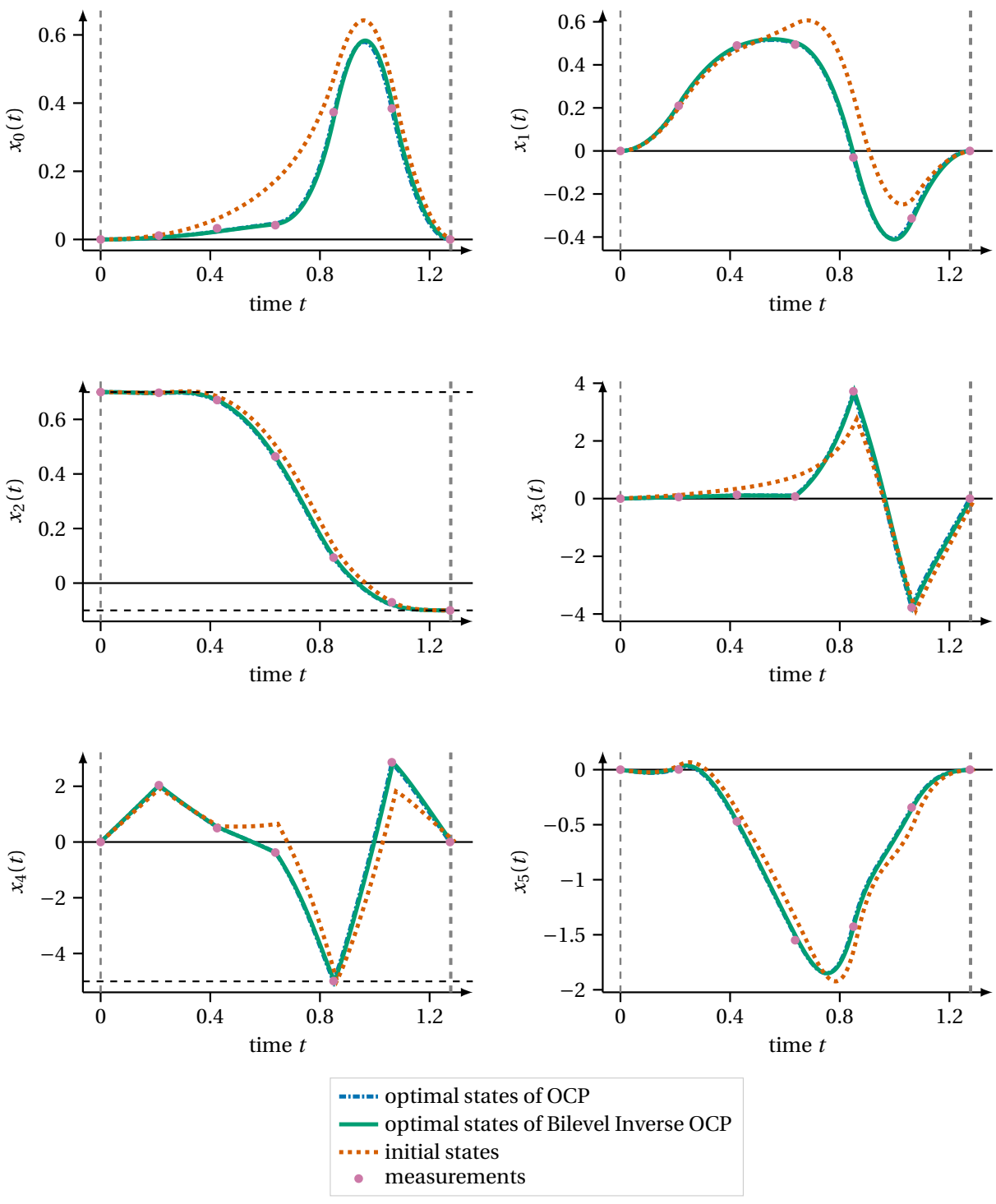


**Figure 8.13:** This figure depicts the optimal differential states in the solution of Bilevel Inverse OCP (8.11) computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes. This result is compared to the corresponding OCP (8.10) solution achieved with the Direct Multiple Shooting Method in PARDYNOPT. Generated pseudo-measurements and initial guesses are also illustrated. The dynamics by Hatz [80] are used as described in subsection 8.2.4.

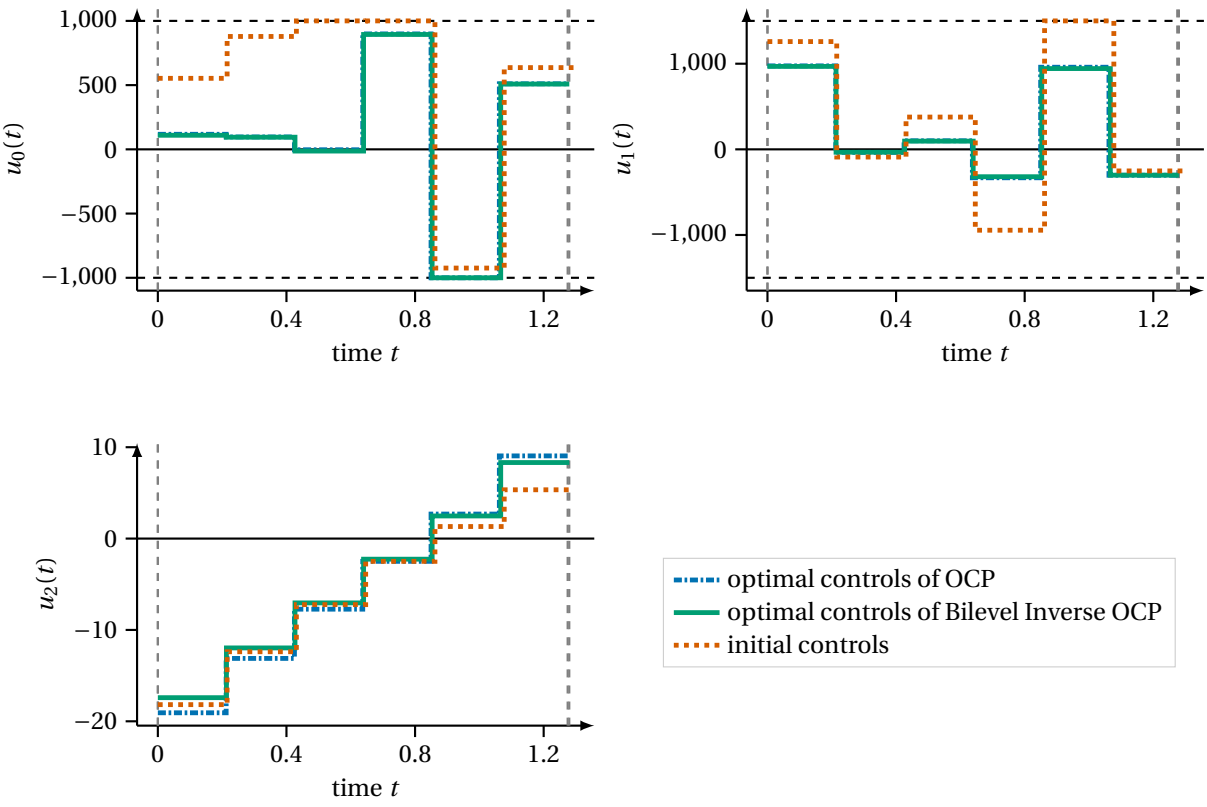


**Figure 8.14:** This figure depicts the optimal differential states and controls in the solution of Bilevel Inverse OCP (8.11) computed with the DISIMFAS in PARDYNOPT with 7 multiple shooting nodes. This result is compared to the corresponding OCP (8.10) solution achieved with the Direct Multiple Shooting Method in PARDYNOPT. Initial guesses are also illustrated. The dynamics by Hatz [80] are used as described in subsection 8.2.4.

edge of the active set in the solution of the corresponding OCP (8.10) the computation of the solution of Bilevel Inverse OCP (8.11) can be accelerated significantly with our DISIMFAS.

With a relative error of 9.60% for objective weight $\alpha_2^* = 2.1919$ and 2.59% for model parameter $p_0^* = 7.3055 \cdot 10^{-1}$, both quantities are reproduced with a relative error of less than 10% with PARDYNOPT. With PARAOCP the objective weight $\alpha_2^* = 2.2815$ can be approximated with a relative error of 14.1% and model parameter $p_0^* = 7.3888 \cdot 10^{-1}$ with 1.48%. With both methods the objective weight and model parameter of the corresponding OCP are estimated with an acceptable accuracy, but with much less computational effort with our DISIMFAS.

### 8.2.5 Summary

So far, to apply our DISIMFAS as described in chapter 4 the knowledge of the optimal active set $\mathcal{A}^*$ is essential. In some applications this is a huge drawback. However, if measurements are considered as in Bilevel Inverse OCPs, where a PE Problem is constrained by an OCP, this structural information is often available in advance and, hence, their numerical treatment can be accelerated significantly. This could be illustrated in the second case study by comparison of results performed with our DISIMFAS and results achieved in the work of Hatz [80] for a similar set-up. Furthermore, structurally different initial guesses do not have a considerable impact in the calculations of the first case study. A comparable analysis in [80] required the use of the lifting approach by Hatz to cope with the resulting MPCC, because the Direct All-at-Once Approach did not converge.

# Chapter 9

# Bilevel Inverse Optimal Control for a Basic Walker Gait Model

In this chapter we present numerical results for the basic walker gait model introduced in chapter 5 for assessing the applicability and performance of our DISIMFAS implemented in the software package PARDYNOPT. It serves as a basic rigid multibody system gait model for a human like locomotion.

## 9.1 Case Study: Basic Walker as Basic Model for Human Locomotion

In this case study we consider an optimal gait of the basic walker rigid multibody system and solve a Bilevel Inverse OCP of the form (5.28) to identify individual objective weights and model parameters. The basic walker model is chosen as a basic model to study our DISIMFAS and its application in the context of human locomotion, where we use simulated measurements for a better investigation of the performance of the developed method. The multi-stage OCP (5.16), which describes two optimal steps, is described in detail in section 5.2. In the solution of (5.16) the phasewise defined differential states and controls,

$$\boldsymbol{x}_{\mathrm{j}}(t) = \begin{pmatrix} \boldsymbol{q}_{\mathrm{j}}(t) \\ \dot{\boldsymbol{q}}_{\mathrm{j}}(t) \end{pmatrix} = \begin{pmatrix} x_h^{\mathrm{j}}(t) \\ y_h^{\mathrm{j}}(t) \\ \varphi_l^{\mathrm{j}}(t) \\ \varphi_r^{\mathrm{j}}(t) \\ \dot{x}_h^{\mathrm{j}}(t) \\ \dot{y}_h^{\mathrm{j}}(t) \\ \dot{\varphi}_l^{\mathrm{j}}(t) \\ \dot{\varphi}_r^{\mathrm{j}}(t) \end{pmatrix} \quad \text{and} \quad \boldsymbol{u}_{\mathrm{j}}(t) = \begin{pmatrix} \tau_{\mathrm{j0}}^{\mathrm{act}}(t) \\ \tau_{\mathrm{j1}}^{\mathrm{act}}(t) \end{pmatrix} = \begin{pmatrix} \tau_l^{\mathrm{j}}(t) \\ \tau_r^{\mathrm{j}}(t) \end{pmatrix},$$

respectively, are optimal with respect to a weighted sum of two selected optimization criteria - the final time of the process and the effort to produce the steps. The OCP is defined for some chosen weights $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \alpha_2 \end{pmatrix}^T$ on fixed and *normalized* time horizons $\mathcal{T}^{\mathrm{j}} := [t_{\mathrm{s}}^{\mathrm{j}}, t_{\mathrm{f}}^{\mathrm{j}}] = [0,1]$ with stage duration parameters $d_{\mathrm{j}}$ for each model stage $\mathrm{j} = 1,2$. Hence, the duration parameters are decision variables in (5.16) defining the final time. In Table 9.1 these quantities are summarized together with model parameters $\boldsymbol{p} = \begin{pmatrix} \ell & M & m \end{pmatrix}^T$, which describe the basic walker model. All other arising quantities in OCP (5.16) are defined and explained in detail in section 5.2 including objective function, dynamics, jump conditions and other constraints. To simulate measurements and generate initial guesses for setting up a Bilevel Inverse OCP (9.1) as defined in subsection 9.1.1, the corresponding OCPs of the form (5.16) are solved with the software package PARDYNOPT and its implementation of the Direct Multiple Shooting Method [26] as described in subsection 2.2.1 with an equidistant multiple shooting grid with 11 nodes per model stage $\mathrm{j} = 1,2$ and constant controls on each multiple shooting interval. Furthermore, for the arising initial value problems SOLVIND is used with an adaptive BDF method implemented in DAESOL-II [5] and its sensitivity and derivative generation with its interface to ADOL-C[171]. IPOPT [169] solves the resulting NLP with an exact Hessian approximation and a relative convergence tolerance of $10^{-6}$. The setting in PARDYNOPT is summarized in Table 9.3. The initial guesses for the state values are set to $\begin{pmatrix} 0 & \ell & 0.2 & -0.2 & \ell & 0.2\cdot\ell & \ell & \ell \end{pmatrix}^T$ at the first multiple shooting node, to $\begin{pmatrix} 0.45\cdot\ell & \ell & -0.2 & 0.2 & \ell & 0.2\cdot\ell & \ell & \ell \end{pmatrix}^T$ at the transition between model stage 1 and 2, and to $\begin{pmatrix} 0.9\cdot\ell & \ell & 0.2 & -0.2 & \ell & 0.2\cdot\ell & \ell & \ell \end{pmatrix}^T$ at the final multiple shooting node. All other initial guesses for the

**Table 9.1:** Definition of quantities which appear in the multi-stage OCP (5.16) for the basic walker example.

| Symbol | Description | Value | |
|---|---|---|---|
| $x_h^{\mathrm{j}}$ | horizontal position of head | $x_h^1(t_{\mathrm{s}}^1) = 0,$ | $x_h^2(t_{\mathrm{f}}^2) = 0.9\ell$ |
| $y_h^{\mathrm{j}}$ | vertical position of head | $y_h^1(t_{\mathrm{s}}^1) = y_h^2(t_{\mathrm{f}}^2)$ | |
| $\varphi_l^{\mathrm{j}}$ | rotation of left leg around head | | |
| $\varphi_r^{\mathrm{j}}$ | rotation of right leg around head | | |
| $\dot{x}_h^{\mathrm{j}}$ | horizontal velocity of head | $\dot{x}_h^1(t_{\mathrm{s}}^1) = \dot{x}_h^2(t_{\mathrm{f}}^2)$ | |
| $\dot{y}_h^{\mathrm{j}}$ | vertical velocity of head | $\dot{y}_h^1(t_{\mathrm{s}}^1) = \dot{y}_h^2(t_{\mathrm{f}}^2)$ | |
| $\dot{\varphi}_l^{\mathrm{j}}$ | angular velocity of left leg | $\dot{\varphi}_l^1(t_{\mathrm{s}}^1) = \dot{\varphi}_l^2(t_{\mathrm{f}}^2)$ | |
| $\dot{\varphi}_r^{\mathrm{j}}$ | angular velocity of right leg | $\dot{\varphi}_r^1(t_{\mathrm{s}}^1) = \dot{\varphi}_r^2(t_{\mathrm{f}}^2)$ | |
| $\tau_l^{\mathrm{j}}$ | left torque around head | | |
| $\tau_r^{\mathrm{j}}$ | right torque around head | | |
| $d_{\mathrm{j}}$ | duration of single support phase | | |
| $\ell$ | length of legs | | |
| $M$ | point mass of base segment | | |
| $m$ | point mass of foot segment | | |
| $g$ | gravitational acceleration | 9.81 | |

state values are set to 0 at each multiple shooting node, the initial guesses for the values of all control parameters are set to 1, and the duration parameter $d_1$ and $d_2$ to 1. Furthermore, the bounds for all variables are chosen as defined in subsection 5.2.3 such that they are not active in the solutions.

### 9.1.1 Bilevel Inverse OCP for a Basic Walker Example

For setting up a Bilevel Inverse OCP of the form (5.28) we consider objective weights $\boldsymbol{\alpha} = \begin{pmatrix}\alpha_1 & \alpha_2\end{pmatrix}^T$ and model parameters $\boldsymbol{p} = \begin{pmatrix}\ell & M & m\end{pmatrix}^T$ which enter the corresponding lower level OCP. These can be identified by solving the following Bilevel Inverse OCP

$$\min_{\substack{\alpha_1, p_0,\\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \frac{1}{2}\sum_{\mathrm{j}=1}^{2}\sum_{\mathrm{n}=0}^{\mathrm{n}_{\mathrm{m}}^{\mathrm{j}}-1}\sum_{\mathrm{k}=0}^{\mathrm{n}_{\mathrm{h}}^{\mathrm{j}}-1}\frac{\left(q_{\mathrm{jk}}(t_{\mathrm{jn}}^{\mathrm{m}}) - \eta_{\mathrm{jnk}}\right)^2}{\sigma_{\mathrm{jnk}}^2} \tag{9.1a}$$

$$\text{s.t.} \quad (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \text{ solve OCP (5.16)}, \tag{9.1b}$$

$$\alpha_1 \geq 0, \alpha_2 = 1, \tag{9.1c}$$

$$0.5 \leq p_0 \leq 2, \tag{9.1d}$$

$$p_1 = 2, p_2 = 1. \tag{9.1e}$$

In this case study we fix weight $\alpha_2 = 1$ and determine $\alpha_1$ by solving (9.1). This can be done here because we choose $\alpha_2 \neq 0$ for the generation of measurements. Furthermore, we set $p_1 = M = 2$ and $p_2 = m = 1$ and only use model parameter $p_0 = \ell$ as decision variable. We find that $\alpha_1$ is correlated to $M$ and $m$ in the way that for varying pairs of $(\alpha_1, (M, m))$ we achieve the same optimal generalized coordinates $x_{\mathrm{jk}}^{*\mathrm{OCP}}(t) = q_{\mathrm{jk}}^{*\mathrm{OCP}}(t)$, $\mathrm{j} = 1,2,\ \mathrm{k} = 0,\ldots,3$ as solutions of the OCPs.

Similar to the previous chapter 8, we simulate measurements from a solution of (5.16) with objective weights $\boldsymbol{\alpha} = \begin{pmatrix}1 & 1\end{pmatrix}^T$ and model parameters $\boldsymbol{p} = \begin{pmatrix}1 & 2 & 1\end{pmatrix}^T$ for a better investigation of the proposed method. We add normally distributed noise $\boldsymbol{\varepsilon}$ with zero mean and standard deviation $\boldsymbol{\sigma}$ with values $\sigma_{\mathrm{k}} = 0.05 \cdot 1$, for $\mathrm{l} = 0,2,3$ and $\sigma_1 = 0.05 \cdot 0.04$ to the generalized coordinates $\boldsymbol{q}_{\mathrm{j}}^{*\mathrm{OCP}}$ of solution $\boldsymbol{x}^{*\mathrm{OCP}}$ of the corresponding lower level OCP.

$$\eta_{\mathrm{jnk}} = q_{\mathrm{jk}}^{*\mathrm{OCP}}(t_{\mathrm{jn}}^{\mathrm{m}}) + \varepsilon_{\mathrm{jnk}}, \quad \mathrm{j} = 1,2,\ \mathrm{n} = 0,\ldots,\mathrm{n}_{\mathrm{m}}^{\mathrm{j}} - 1,\ \mathrm{k} = 0,\ldots,3.$$

The measurement times coincide with the multiple shooting time grid chosen in the calculations. The initialization for solving the Bilevel Inverse OCP is again performed by solving an OCP but for estimated weights and model parameters which are set to $\boldsymbol{\alpha} = \begin{pmatrix} 10 & 1 \end{pmatrix}^T$ and $\boldsymbol{p} = \begin{pmatrix} 1.2 & 2 & 1 \end{pmatrix}^T$. The settings for the generations of measurements and initial guesses are summarized in Table 9.2.

**Table 9.2:** Settings of objective weights $\boldsymbol{\alpha}$ and model parameters $\boldsymbol{p}$ in OCPs of the basic walker example for generations of measurements and initial guesses. Quantities in parentheses $(\cdot)$ are not determined in the Bilevel Inverse OCP.

| OCP for | measurements | initial guesses |
|---|---|---|
| $\alpha_1$ | 1 | 10 |
| $\alpha_2$ | (1) | (1) |
| $p_0$ | 1 | 1.2 |
| $p_1$ | (2) | (2) |
| $p_2$ | (1) | (1) |

The Bilevel Inverse OCP (9.1) is now solved with our DISIMFAS implemented in PARDYNOPT. The arising initial value problems and the corresponding sensitivities as well as the derivative generations of all model functions are calculated on the multiple shooting grid in the same way as described previously for the solution of OCP (5.16) using SOLVIND [5]. Furthermore, the bounds on the variables corresponding to the lower level OCP are broadly formulated, and the arising decoupled inequality constraints are supposed to be always $> 0$ during the single support phases. Hence, no active set has to be specified in PARDYNOPT. Usually, in PE Problems constrained by OCPs which describe human gaits, very loose bounds on the differential states $\boldsymbol{x}_j(t)$, control functions $\boldsymbol{u}_j(t)$, and duration parameters $d_j$ defined on each model stage $j = 1,2$ can be stated because measurements are tracked and, hence, the corresponding decision variables typically do not exceed natural bounds. In the basic gait model (9.1) this is also the case, such that in the solution of the Bilevel Inverse OCP all inequality constraints of the lower level OCP are satisfied. In the DISIMFAS the resulting NLP of the form (4.13) is then solved with IPOPT [169] with a relative convergence tolerance of $10^{-5}$ (defined in [169] as error tolerance). For the arising gradient of the Lagrangian of the corresponding lower level OCP as constraint, an efficient structure exploiting implementation in PARDYNOPT is used. For the constraint Jacobian we choose the dense finite differences implementation in IPOPT and for the Hessian of the Lagrangian a limited-memory quasi-NEWTON method (L-BFGS update formula) in IPOPT. The PARDYNOPT setting for solving OCP (5.16) and Bilevel Inverse OCP (9.1) is summarized in Table 9.3.

**Table 9.3:** Setting used in PARDYNOPT to solve the multi-stage OCP (5.16) and the corresponding Bilevel Inverse OCP (9.1).

| | OCP (5.16) | Bilevel Inverse OCP (9.1) |
|---|---|---|
| solution method | Direct Multiple Shooting Method | Direct Simultaneous Approach with Fixed Active Set |
| # multiple shooting nodes | (11,11) | (11,11) |
| NLP solver | IPOPT | IPOPT |
| convergence tolerance | $10^{-5}$ | $10^{-5}$ |
| constraint Jacobian | sparse | dense |
| Hessian | exact | BFGS update |
| integrator and derivatives | SOLVIND | SOLVIND |
| initial guess $\alpha_1$ | - | 10 |
| initial guess $p_0$ | - | 1.2 |

**Table 9.4:** Computational result of Bilevel Inverse OCP (9.1) with our DISIMFAS in PARDYNOPT with settings summarized in Table 9.3 and comparison of weight $\alpha_1^*$ to the true values and the phase durations $d_1^*$ and $d_2^*$ to the corresponding OCP (5.16) solution.

| | Bilevel Inverse OCP (9.1) | |
|---|---|---|
| objective $\Psi^*$ | $1.64212 \cdot 10^1$ | |
| constraint violation | $6.42 \cdot 10^{-11}$ | |
| dual infeasibility | $3.60 \cdot 10^{-5}$ | |
| # iterations | 33 | |
| weight | estimate (rel. Err%) | true value |
| $\alpha_1^*$ | 1.2589 (25.89%) | 1 |
| $p_0^*$ | 1.00005 (0.005%) | 1 |
| stage durations | | OCP (5.16) |
| $d_1^*$ | 1.2325 | 1.2434 |
| $d_2^*$ | 1.2325 | 1.2434 |

The DISIMFAS converges after 33 iterations with an objective value of $1.64212 \cdot 10^1$ and a stage duration of 1.2325 for both single support phases, which is similar to the duration 1.2434 of the corresponding OCP solution used for measurement generation. With a relative error of 0.005% for model parameter $p_0^* = 1.00005$ this quantity is reproduced within a very high accuracy by solving the Bilevel Inverse OCP (5.28) with our DISIMFAS. The relative error for objective weight $\alpha_1^* = 1.2589$ is with 25.89% much higher. However, the generated measurements are reproduced within an acceptable quality as illustrated in Figure 9.1 and, hence, the estimated objective weight lies within an acceptable accuracy. In Table 9.4 the result achieved with small constraint violation and infeasibility measure, see IPOPT [169], is summarized, and in Figure 9.1, Figure 9.2, and Figure 9.3 the optimal trajectories and approximations of the control functions are illustrated. In Figure 9.4 the visualization of the basic walker gait as solution of the Bilevel Inverse OCP (5.28) is shown and compared to the solution of the corresponding OCP (5.16) with true weights and parameters.

### 9.1.2 Summary

With the described case study for the basic walker gait model we could illustrate the successful applicability of our DISIMFAS in the context of human locomotion. It serves as a first study of the proposed method with promising results for future identification of unknowns in an OCP describing the gait of a CP patient and its characterization by solving an appropriate Bilevel Inverse OCP as derived in chapter 6. Similar to the basic walker example because of given measurements, the arising inequality constraints and bounds on the variables can usually be chosen very loosely on the lower level. A first investigation is described in the following chapter 10.

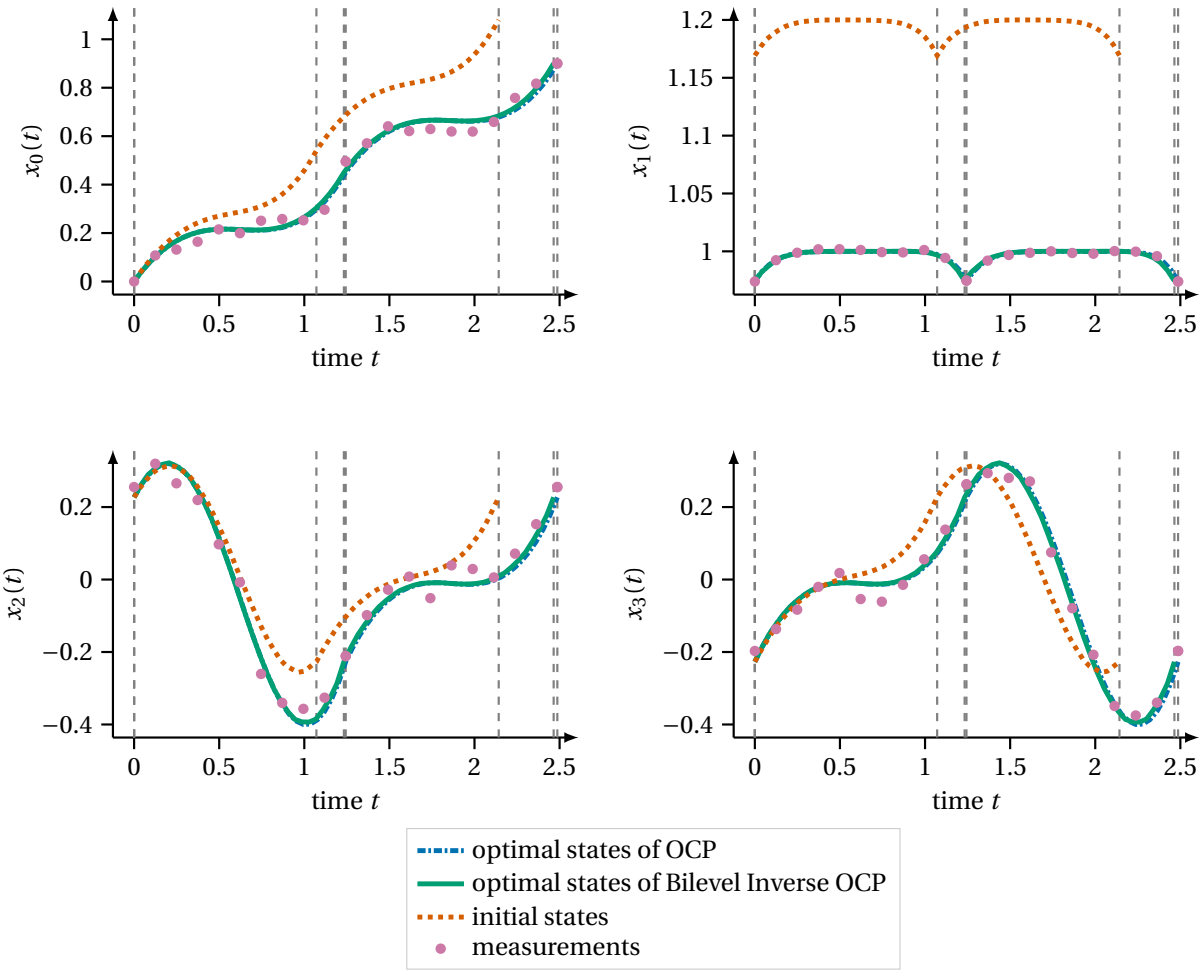


**Figure 9.1:** This figure depicts the optimal differential states corresponding to the generalized coordinates in the solution of Bilevel Inverse OCP (5.28) computed with the DISIMFAS in PARDYNOPT with 22 multiple shooting nodes. This result is compared to the corresponding OCP (5.16) solution. Generated pseudo-measurements and initial guesses are also illustrated.

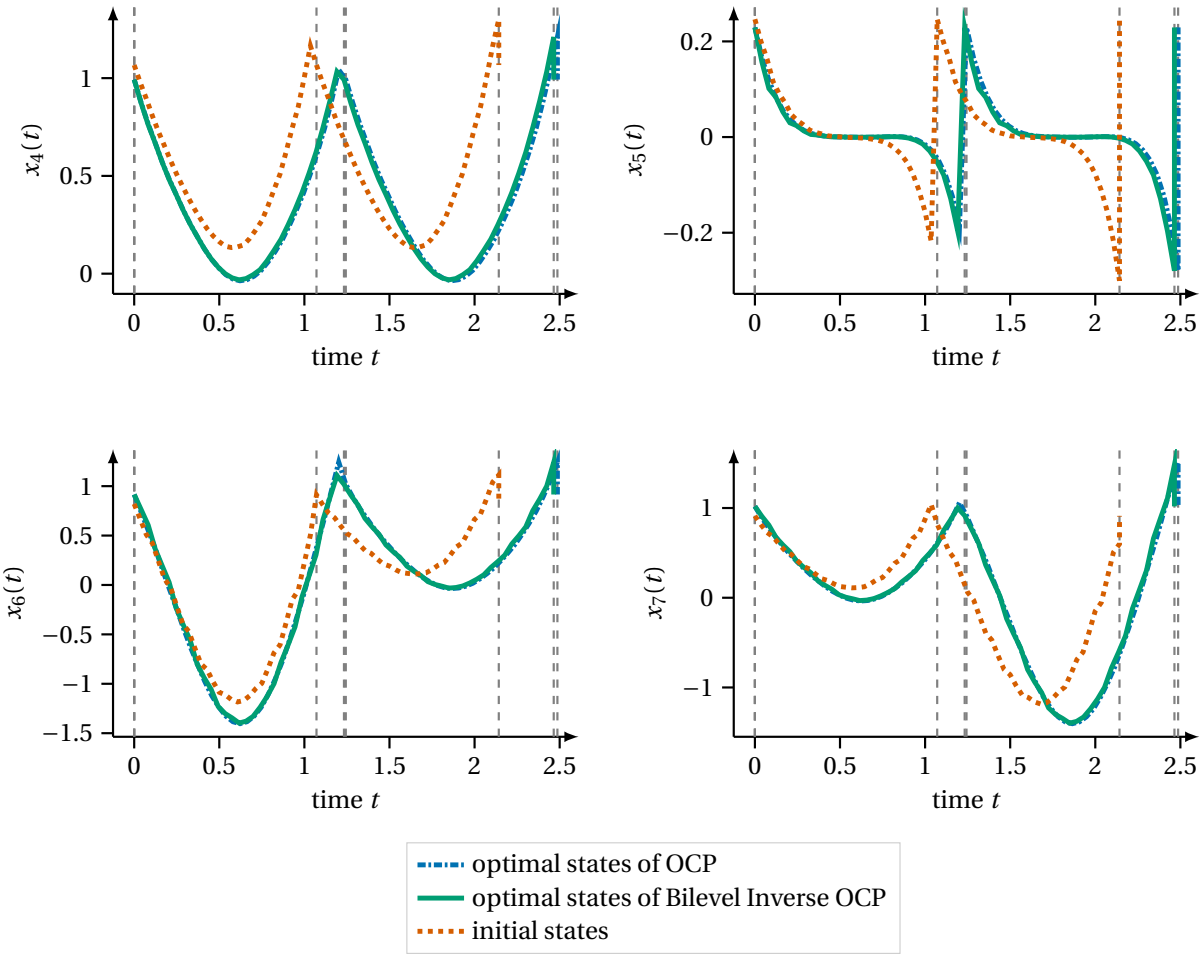


**Figure 9.2:** This figure depicts the optimal differential states corresponding to the generalized velocities in the solution of Bilevel Inverse OCP (5.28) computed with the DISIMFAS in PARDYNOPT with 22 multiple shooting nodes. This result is compared to the corresponding OCP (5.16) solution. Initial guesses are also illustrated.

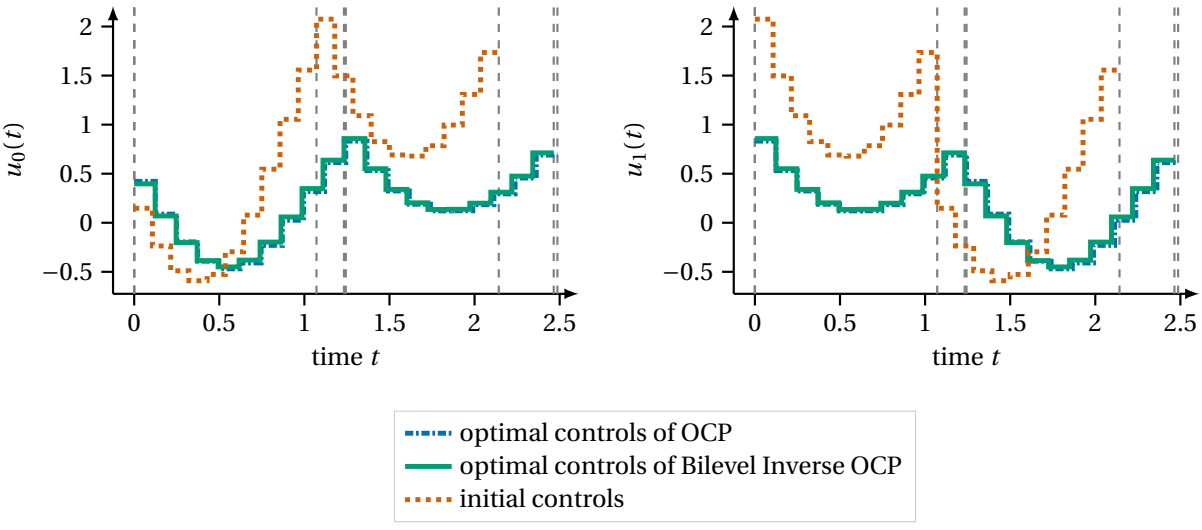


**Figure 9.3:** This figure depicts the optimal controls in the solution of Bilevel Inverse OCP (5.28) computed with the DISIMFAS in PARDYNOPT with 22 multiple shooting nodes. This result is compared to the corresponding OCP (5.16) solution. Initial guesses are also illustrated.

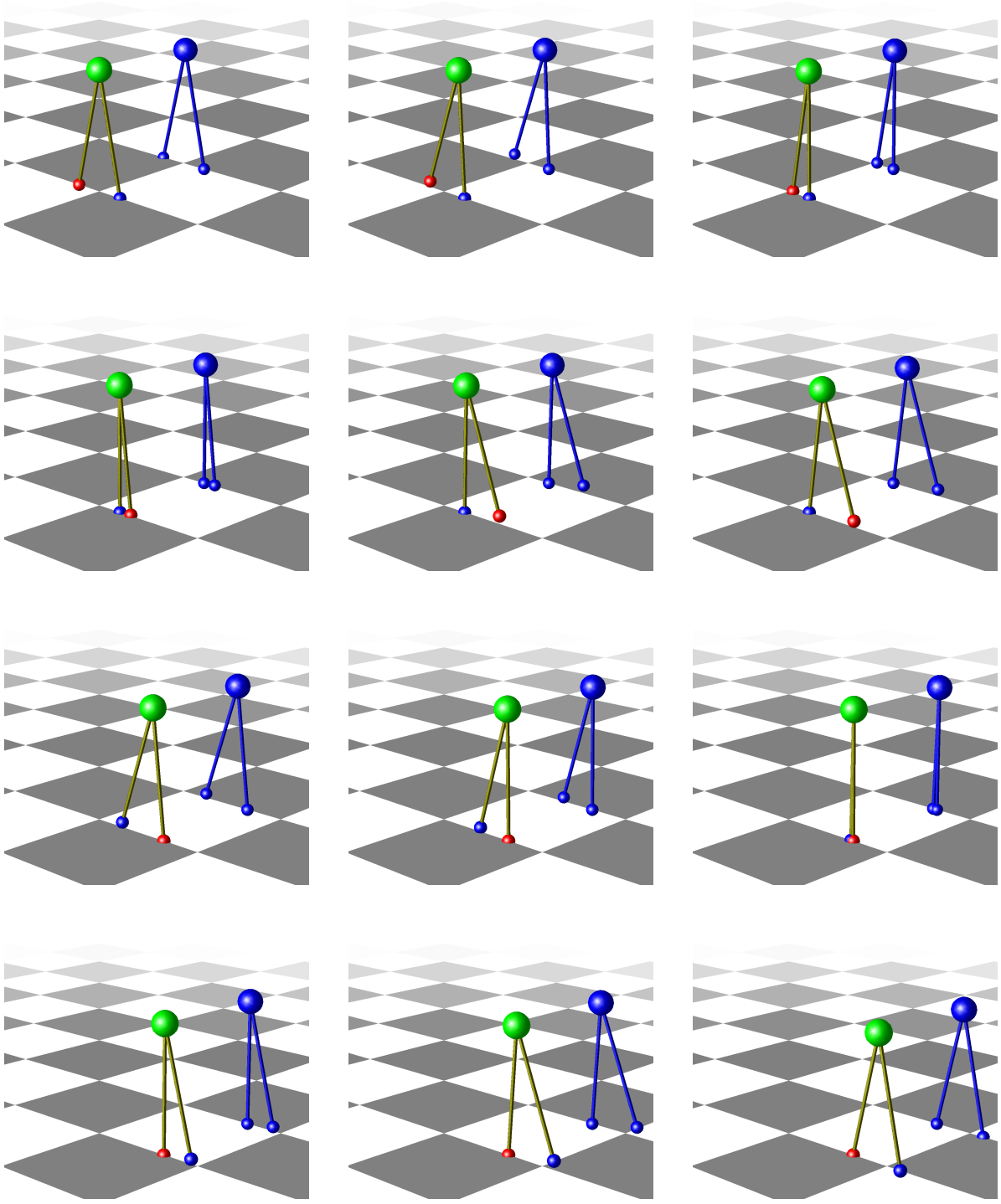

**Figure 9.4:** Visualization of basic walker gait as solution of Bilevel Inverse OCP (5.28) (colored multibody system model) at various time points and comparison to the corresponding OCP (5.16) solution (blue multibody system model). Snapshots are arranged in reading direction, with initial position at upper left corner, and final position at lower right corner.

# Chapter 10

# Numerical Results for Cerebral Palsy Gait Model

In this chapter we present numerical results for the patient-specific CP walker gait model derived in chapter 6 for reconstruction of the dynamics to given motion capture data provided by the HEIDELBERG MOTION-LAB [173]. Furthermore, we analyze the proposed multi-stage OCP formulation to synthesize varying gaits of a patient with CP for the chosen combination of optimization criteria with differently set weights. This OCP formulation serves as a gait model of a CP patient at the lower level of the Bilevel Inverse OCP for the identification of objective weights and model parameters under consideration of given measurements. In the last section 10.4 we give a case study for investigation of the derived supervised learning approach from section 6.7 for the identification of optimal weights via DNNs.

## 10.1 Solution Approaches, Initialization and Implementation Notes

Before we discuss the numerical results we give the solution approaches, initialization procedures and other implementation notes needed in the next sections to solve the least-squares multi-stage OCP (6.18) and the multi-stage OCPs (6.35) describing varying gaits of a CP patient. For the solution of both problems we follow the Direct Multiple Shooting Method [26] introduced in subsection 2.2.1. For this purpose, we use the efficient implementation of the proposed method together with a structure exploiting SQP method in the software package MUSCOD-II [108] and apply the setting summarized in Table 10.1.

**Table 10.1:** Setting used in MUSCOD-II to solve the multi-stage OCP (6.18) for reconstruction of the dynamics and differently weighted OCPs (6.35) for gait synthesis for a patient with CP.

| Setting in MUSCOD-II | for sol. of OCP (6.18) and OCP(6.35) |
|---|---|
| solution method | Direct Multiple Shooting Method |
| # multiple shooting nodes | (16, 17) |
| NLP algorithm | structure exploiting SQP |
| convergence tolerance | $10^{-5}$ |
| Hessian | BFGS update |
| integrator | DAESOL-II |
| derivatives | VDE |

Please note, that the transitions in MUSCOD-II can be modeled as extra model stages with 2 multiple shooting nodes for each stage and time duration parameters fixed to 0. This is not listed in Table 10.1, to be aligned with the OCP formulations, which consider only two model stages. Furthermore, the controls are approximated by piecewise linear and continuous control functions at each model stage. As already discussed earlier, we use the software tool RBDL [53] for all computations regarding the rigid multibody system model of a patient with CP from section 6.1 and section 6.2. For initialization of the differential states we use processed measurements and set the discretized control parameter values to 1 in the least-squares OCP. In the OCP describing CP gaits we use the solution of this dynamics reconstruction and fix the model parameters to the estimated values. The lower and upper bounds on the decision variables are set appropriately, such that they are not reached in the solutions, see subsection 6.3.3. Appropriate scaling of the optimization criteria, the decision variables, as well as of the constraints are performed to account for different orders of magnitudes. The diagonal entries in the

matrices $\boldsymbol{W^{\dot{\tau}}}$ and $\boldsymbol{W^{\tau}}$, which appear in the least-squares objective (6.20) and in the optimization criteria of the objective (6.36) in OCP (6.35), are calculated based on a first solution of a non-weighted least-squares OCP (6.18) with regularization parameter $\gamma^{\mathrm{R}}$ set to zero. The results are listed in Table 10.2.

**Table 10.2:** Diagonal entries of the matrices $\boldsymbol{W^{\dot{\tau}}}$ and $\boldsymbol{W^{\tau}}$, which correspond to the given joint and motion around the given axis.

| Joint | Axis | $(w_{\mathrm{k}}^{\dot{\tau}})^{-2}$ | $(w_{\mathrm{k}}^{\tau})^{-2}$ |
|---|---|---|---|
| Hips | $y$ | $5.8 \cdot 10^5$ | $1.2 \cdot 10^3$ |
| Hips | $x$ | $1.7 \cdot 10^5$ | $6.8 \cdot 10^2$ |
| Hips | $z$ | $4.3 \cdot 10^5$ | $2.6 \cdot 10^2$ |
| Knees | $y$ | $8.0 \cdot 10^5$ | $3.2 \cdot 10^3$ |
| Ankle | $y$ | $6.3 \cdot 10^5$ | $4.2 \cdot 10^3$ |
| Ankle | $x$ | $2.8 \cdot 10^5$ | $1.0 \cdot 10^3$ |
| Ankle | $z$ | $5.1 \cdot 10^5$ | $2.9 \cdot 10^3$ |

## 10.2 Reconstruction of Cerebral Palsy Gait Model using Motion Capture Data

In this section we present the numerical results for the dynamics reconstruction of the derived CP gait model from chapter 6 achieved by solving the least-squares OCP (6.18). We use the setting and initialization described in the previous section and incorporate motion capture data at given equidistant measurement times provided by the HEIDELBERG MOTIONLAB [173], which was processed as explained in subsection 6.2.6 and subsection 6.2.7. The measurement time grid is chosen to be the same as the multiple shooting grid, such that given structures in the arising quantities of the Direct Multiple Shooting Method and the applied SQP algorithm can be exploited efficiently. Within the reconstruction the duration parameters $d_1, d_2$, and the lower and upper

**Table 10.3:** Fixed values of fixed stage duration parameters and model parameters defining the lower and upper bounds on the range of motion in both knees in the dynamics reconstruction of OCP (6.18).

| fixed stage duration | | |
|---|---|---|
| $d_1^*$ | 0.500 | |
| $d_2^*$ | 0.533 | |
| | | |
| fixed model parameters | knee_r | knee_l |
| $\overline{\beta}_{\mathrm{i}}$ | 1.75 | 1.65 |
| $\underline{\beta}_{\mathrm{i}}$ | 1.3 | 1.2 |

bounds on the range of motion in both knees $\underline{\beta}_{\mathrm{i}}, \overline{\beta}_{\mathrm{i}}, \mathrm{i} = \{\text{knee\_r, knee\_l}\}$ are set to fixed measured values, see Table 10.3. The remaining model parameters $\hat{\boldsymbol{p}}$, which include curvature parameters $\kappa_{\mathrm{i}} = \underline{\kappa}_{\mathrm{i}} = \overline{\kappa}_{\mathrm{i}}, \mathrm{i} = \{\text{knee\_r, knee\_l}\}$ and damping parameters $\boldsymbol{\delta}$, are estimated by solving the OCP (6.18). Its initial guesses are listed in Table 10.5. As guesses for the damping parameters around the $y$-axis we choose values based on a reduced muscle model of Millard et al. [126, 127]. For the other damping parameters we choose a small initial guess as we expect them to be small, and for the curvature parameters we use the same values as in [151].

The solution of the dynamics reconstruction for the CP gait model is summarized in Table 10.4, and in Table 10.5 the estimated model parameters are listed. According to the latter table, the damping parameters around the $x$- and $z$- axes are relatively small. The damping parameters around the $y$- axes and the curvature parameters stay close to the initial guesses. With $\boldsymbol{\Psi}^* = 0.5 \cdot 1.799625 \cdot 10^{-1}$ the least-squares objective takes a small value in the solution of OCP (6.18), such that the gait model of the patient with CP from chapter 6 can be reproduced within an acceptable quality under consideration of given motion capture data performing a

**Table 10.4:** Computational result of least-squares OCP (6.18) with the Direct Multiple Shooting Method in MUSCOD-II with settings summarized in Table 10.1.

| OCP (6.18) | Reconstruction (R) |
|---|---|
| objective $\Psi^*$ | $0.5 \cdot 1.799625 \cdot 10^{-1}$ |
| constraint violation | $1.07 \cdot 10^{-6}$ |
| kkt tol | $9.03 \cdot 10^{-6}$ |

full gait cycle. In appendix B.1 the corresponding optimal differential states and the optimal controls are illus-

**Table 10.5:** Estimated values of damping parameters $\boldsymbol{\delta}$ and curvature parameters $\kappa_i = \underline{\kappa}_i = \overline{\kappa}_i, i = \{\text{knee\_r}, \text{knee\_l}\}$ in the dynamics reconstruction of OCP (6.18). The initial guesses are listed in brackets, where $^{**}$ denotes values based on [126, 127].

| | $y-$component | | $x-$component | $z-$component |
|---|---|---|---|---|
| damping | | | | |
| $\boldsymbol{\delta}_{\text{hip\_r}}$ | 3.6476 | $(3.66)^{**}$ | $4.7835 \cdot 10^{-2}$ (0.1) | $6.5159 \cdot 10^{-1}$ (0.1) |
| $\boldsymbol{\delta}_{\text{knee\_r}}$ | 2.1527 | $(2.15)^{**}$ | | |
| $\boldsymbol{\delta}_{\text{ankle\_r}}$ | 1.1973 | $(1.18)^{**}$ | $4.9979 \cdot 10^{-3}$ (0.1) | $1.4597 \cdot 10^{-1}$ (0.1) |
| $\boldsymbol{\delta}_{\text{hip\_l}}$ | 3.6584 | $(3.66)^{**}$ | $4.9959 \cdot 10^{-3}$ (0.1) | $9.4413 \cdot 10^{-2}$ (0.1) |
| $\boldsymbol{\delta}_{\text{knee\_l}}$ | 2.1653 | $(2.15)^{**}$ | | |
| $\boldsymbol{\delta}_{\text{ankle\_l}}$ | 1.1982 | $(1.18)^{**}$ | $1.3234 \cdot 10^{-1}$ (0.1) | $8.4823 \cdot 10^{-2}$ (0.1) |
| curvature | | | | |
| $\kappa_{\text{knee\_r}}$ | $3.0626 \cdot 10^{1}$ | (30.0) | | |
| $\kappa_{\text{knee\_l}}$ | $3.0514 \cdot 10^{1}$ | (30.0) | | |

trated. In Figure B.5 one can see that the generalized positions in the solution of the dynamics reconstruction reproduces the reference data very well. Because of the fact that in our gait model several constraints at the feet have to be fulfilled, the biggest deviations can be noticed in the angles at the ankle joints. Furthermore, because of enforcing periodicity constraints, there exist some variations. However, the gait of a patient with CP can still be reproduced within an acceptable range. It should be noted, that the *"measurements"* shown in Figure B.5 are not the measurements used in the least-squares OCP formulation, but provided Vicon angles and calculated knee angles under consideration of (6.15). In OCP (6.18) we choose positions of selected points of the rigid multibody system model in the global reference frame at each time frame of the gait cycle as measurements $\boldsymbol{\eta}^{\text{P}} \in \mathbb{R}^{n_{\text{meas}}}$. See subsection 6.2.6 and subsection 6.2.7 for a more detailed discussion.

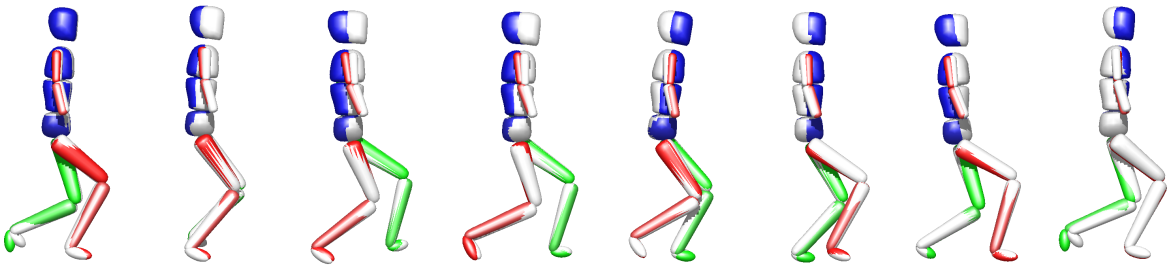

**Figure 10.1:** Visualization of solution of the dynamics reconstruction (colored multibody system model) and comparison with given motion capture data (grey multibody system model) at various time points.

## 10.3 Numerical Analysis of Gait Syntheses for CP Gait Model

In this section we analyze the multi-stage OCP (6.35) from section 6.4 in view of its application in the proposed Bilevel Inverse OCP formulation (6.42). An important aspect is if the stated OCP formulation can cover the main characteristics of the pathological gait of a patient with CP. We solve OCPs with differently weighted objective functions and compare the results with the dynamics reconstruction. In subsection 6.4.1 the objective function is defined in (6.36), where we choose a combination of four optimization criteria on each model stage. In the following we summarize the contributions for each model stage j = 1,2 of every optimization criterion and denote:

- $\boldsymbol{\phi}_1$ as the stability criterion summarizing $\boldsymbol{\phi}_{11}^{\mathrm{L}}$ (6.37) and $\boldsymbol{\phi}_{21}^{\mathrm{L}}$ (6.38),
- $\boldsymbol{\phi}_2$ as the criterion for minimization of the mechanical work summarizing $\boldsymbol{\phi}_{\mathrm{j2}}^{\mathrm{L}}$ (6.39) for j = 1,2,
- $\boldsymbol{\phi}_3$ as the mechanical effort criterion summarizing $\boldsymbol{\phi}_{\mathrm{j3}}^{\mathrm{L}}$ (6.40) for j = 1,2, and
- $\boldsymbol{\phi}_4$ as the criterion for minimization of the integrated squared torque derivatives summarizing $\boldsymbol{\phi}_{\mathrm{j4}}^{\mathrm{L}}$ (6.41) for j = 1,2.

In total, we perform five gait syntheses for a CP patient. We use the setting and initialization described in section 10.1, and fix the model parameters to the estimated values from Table 10.5 and to fixed bounds on the range of motion in both knees from Table 10.3. In all calculation the duration of each single support phase is optimized.

S denotes the solution of OCP (6.35), where only the corresponding weight $\alpha_1$ of the stability criterion $\boldsymbol{\phi}_1$ is set to $10^3$ with a regularization term $\alpha_4 = 10^{-4}$ to avoid redundancies in the controls. With MW, E, and TD we denote the solutions, where the weights of the corresponding criteria for minimization of mechanical work $\alpha_2$, of mechanical effort $\alpha_3$, and of integrated squared torque derivatives $\alpha_4$, respectively, are set to $10^{-1}$ in each case. Apart from a small regularization in the solutions MW and E, all other weights are set to zero. Furthermore, we perform one calculation denoted by C, where all four criteria contribute in OCP (6.35) with the corresponding weights set to $\alpha_1 = 10^3$, $\alpha_2 = 10^{-1}$, $\alpha_3 = 10^{-1}$, and $\alpha_4 = 10^{-1}$. The five different settings are summarized in Table 10.6. Note, that we choose distinct values for the weights to compensate for the different order of magnitudes in each optimization criterion.

**Table 10.6:** Five selected settings of the objective weights in the OCPs (6.35) denoted by S, MW, E, TD, and C.

| OCP (6.35) | S | MW | E | TD | C |
|---|---|---|---|---|---|
| $\alpha_1$ | $10^3$ | 0.0 | 0.0 | 0.0 | $10^3$ |
| $\alpha_2$ | 0.0 | $10^{-1}$ | 0.0 | 0.0 | $10^{-1}$ |
| $\alpha_3$ | 0.0 | 0.0 | $10^{-1}$ | 0.0 | $10^{-1}$ |
| $\alpha_4$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-1}$ | $10^{-1}$ |

In Table 10.7 the results of all five OCP solutions performed with MUSCOD-II [108] are summarized and in appendix B.2 the optimal differential states and controls are illustrated and compared to the results of the dynamics reconstruction. Of course, so far the chosen values for the weights for the contributions of varying optimization criteria are not optimal in the sense of a minimized deviation to given motion capture data. However, the results for the selected OCP solutions are already in a similar range as the reconstruction data and motivate us to use the proposed OCP formulation in a Bilevel Inverse OCP framework as stated in section 6.6 for future investigations. For the solutions S, MW, and E the optimal duration parameters are comparable to the measured and fixed duration parameters in R, see Table 10.7 and Table 10.3. In contrast to this, the phase durations in E are very short. For the solution C with contributions of all criteria in a comparable order of magnitude the duration of each phase is smaller than for the reconstructed data.

**Table 10.7:** Computational results of five solutions of selected OCPs (6.35) denoted by S, MW, E, TD, and C as defined in Table 10.6. We applied the Direct Multiple Shooting Method in MUSCOD-II with settings summarized in Table 10.1.

| OCP (6.35) | S | MW | E | TD | C |
|---|---|---|---|---|---|
| objective $\Phi^*$ | $5.8484 \cdot 10^{-3}$ | $2.4106 \cdot 10^{-1}$ | $1.7744 \cdot 10^{-2}$ | $1.7409 \cdot 10^{-2}$ | $7.9137 \cdot 10^{-1}$ |
| const. viol. | $1.27 \cdot 10^{-5}$ | $2.35 \cdot 10^{-5}$ | $1.43 \cdot 10^{-5}$ | $7.72 \cdot 10^{-6}$ | $2.28 \cdot 10^{-5}$ |
| kkt tol | $6.84 \cdot 10^{-7}$ | $6.29 \cdot 10^{-6}$ | $9.07 \cdot 10^{-7}$ | $7.77 \cdot 10^{-7}$ | $5.06 \cdot 10^{-6}$ |
| stage dur. | | | | | |
| $d_1^*$ | $4.61 \cdot 10^{-1}$ | $5.15 \cdot 10^{-1}$ | $1.95 \cdot 10^{-1}$ | $7.23 \cdot 10^{-1}$ | $3.57 \cdot 10^{-1}$ |
| $d_2^*$ | $4.85 \cdot 10^{-1}$ | $4.95 \cdot 10^{-1}$ | $1.58 \cdot 10^{-1}$ | $4.65 \cdot 10^{-1}$ | $3.00 \cdot 10^{-1}$ |

In Table 10.8 we furthermore evaluate each optimization criterion for all five OCPs solutions S, MW, E, TD, and C, and compare it to the solution of the dynamics reconstruction R. As expected, the minimal values of the optimization criteria $\boldsymbol{\phi}_\mathrm{i}, \mathrm{i} = 2,3,4$, are taken at the corresponding solutions MW, E, and TD, respectively. Contrary to this, in the minimization of the stability criterion $\boldsymbol{\phi}_1$, which is a distance measure and denoted by S, the value is slightly bigger than in the result for the combined objective in C. However, both values are in the range of $10^{-6}$, and, hence, comparable. In the solution MW the evaluated stability criterion $\boldsymbol{\phi}_1$ is in the same order of magnitude as for the reconstruction data in R. However, in sum the solution S shows a similar distribution for all optimization criteria. Because both objectives – the stability criterion and the least-squares objective in R – are distance measures, which are regularized with the same factor, the regularity might affect this similarity. However, to obtain more information into what happens *"inside"* the patient it is desirable to incorporate other measures and, therefore, varying criteria, such as the ones chosen here for minimization of mechanical work, mechanical effort, and squared torque derivatives. In the minimization TD the values for the mechanical work and mechanical effort are relatively high and comparable to the ones for the solution R. Whereas, when each of the latter criteria $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$ is minimized in MW and E, it also effects the optimization criterion $\boldsymbol{\phi}_4$, such that the value is higher as in the solutions S and R.

**Table 10.8:** Values of the contributions of each optimization criterion $\boldsymbol{\phi}_i$ in the OCP solutions denoted by S, MW, E, TD, and C as defined in Table 10.6. The results are compared to the values of the dynamics reconstruction denoted by R.

| | S | MW | E | TD | C | R |
|---|---|---|---|---|---|---|
| $\boldsymbol{\phi}_1$ | $4.47 \cdot 10^{-6}$ | $8.29 \cdot 10^{-3}$ | $1.70 \cdot 10^{-2}$ | $1.08 \cdot 10^{-2}$ | $3.29 \cdot 10^{-6}$ | $2.26 \cdot 10^{-3}$ |
| $\boldsymbol{\phi}_2$ | $2.05 \cdot 10^{1}$ | 2.40 | $1.13 \cdot 10^{1}$ | $1.82 \cdot 10^{1}$ | 4.16 | $2.16 \cdot 10^{1}$ |
| $\boldsymbol{\phi}_3$ | $1.32 \cdot 10^{1}$ | 5.95 | 1.77 | $1.30 \cdot 10^{1}$ | 3.12 | $1.41 \cdot 10^{1}$ |
| $\boldsymbol{\phi}_4$ | $1.38 \cdot 10^{1}$ | 7.28 | 6.93 | $1.74 \cdot 10^{-1}$ | $6.05 \cdot 10^{-1}$ | $1.53 \cdot 10^{1}$ |

All synthesized CP gaits for solutions S, MW, E, TD, and C with settings listed in Table 10.6 are visualized in Figure 10.2, Figure 10.3, Figure 10.4, Figure 10.5, and Figure 10.6, respectively. With the introduced limited range of motion in the knees and the patient-specific knee axes, the typical pathological gait, or so-called *crouched* gait, can already be modeled sufficiently well with the multi-stage OCP formulation (6.35). If we take a look at the optimal generalized positions as illustrated in Figure B.5, the optimal trajectories $\boldsymbol{x}_9(t)$ and $\boldsymbol{x}_{16}(t)$ of the knee angles vary for all solutions. For the right knee angle $\boldsymbol{x}_9(t)$ of OCP solutions S and E the values are close to the dynamics reconstruction R. The trajectory of the left knee angle $\boldsymbol{x}_{16}(t)$ is captured best for solution S. A further interesting aspect is that the typical movement of the pelvis in direction of the $y$-axis ($\boldsymbol{x}_1(t)$ in Figure B.5) with its $S$-shape in the dynamics reconstuction R also appears for OCP solutions MW and TD. For the solution S, where we minimize the stability criterion, this shifting of the pelvis is not observed. The other optimal trajec-

tories in S capture the reconstruction data reasonably well. Similarly to the solution S, in the solution C, which minimizes a selected combination of weighted optimization criteria, the *S*-shape is not observed. In the OCP solution E, where the mechanical effort is minimized, especially the movement of the pelvis in direction of the $z$-axis differs to all other solutions. Almost no *bouncing* of the pelvis is observed and the corresponding optimal trajectory denoted by $\boldsymbol{x}_2(t)$ in Figure B.5 does not change much over time. On the contrary, for the solution TD, a comparably large change can be observed. In the solution MW, where the mechanical work contributes mainly to the objective function, this *bouncing* of the pelvis can be captured sufficiently well but lies slightly higher as in the dynamics reconstruction R and OCP solution S. The optimal generalized active torques for the dynamics reconstruction R and OCP solution S have a similar pattern. For all other solutions the appearing generalized active torques show mostly smaller deflections. For a detailed comparison of the other optimal differential states and controls we refer to Figure B.5 - Figure B.8 in appendix B.2. In sum, with the different OCP solutions S, MW, E, TD, and C different aspects of the dynamics reconstruction are captured, such as the *S*-shape in the movement of the pelvis or the typical crouched gait.

### 10.3.1 Summary and Outlook

The results in the reconstruction of given motion capture data in section 10.2 for the least-squares OC model as derived in section 6.3 motivates us that the developed rigid multibody system model of the CP patient in chapter 6 with the underlying dynamics can capture the main characteristic of the pathological gait. The analysis of gait syntheses in section 10.3 by solving multi-stage OCPs with differently weighted optimization criteria, show that a wide range of motion can be produced with various characteristics. Hence, we are optimistic that with an optimal weighting of the objective function the gait pattern of a patient with CP can be captured within an acceptable accuracy with more insight on what happens *inside* the patient. This optimal weighting can be determined by solving the Bilevel Inverse OCP (6.42) from section 6.6, with given motion capture data, e.g., under consultation of the developed DISIMFAS from chapter 4. The choice can be underpinned by the fact, that in the performed calculations on the lower level OCPs almost all inequality constraints as well as stated bounds where inactive. Due to the reasons in section 6.6 the investigation of the DISIMFAS for solving the Bilevel Inverse OCP (6.42) is left for future research.

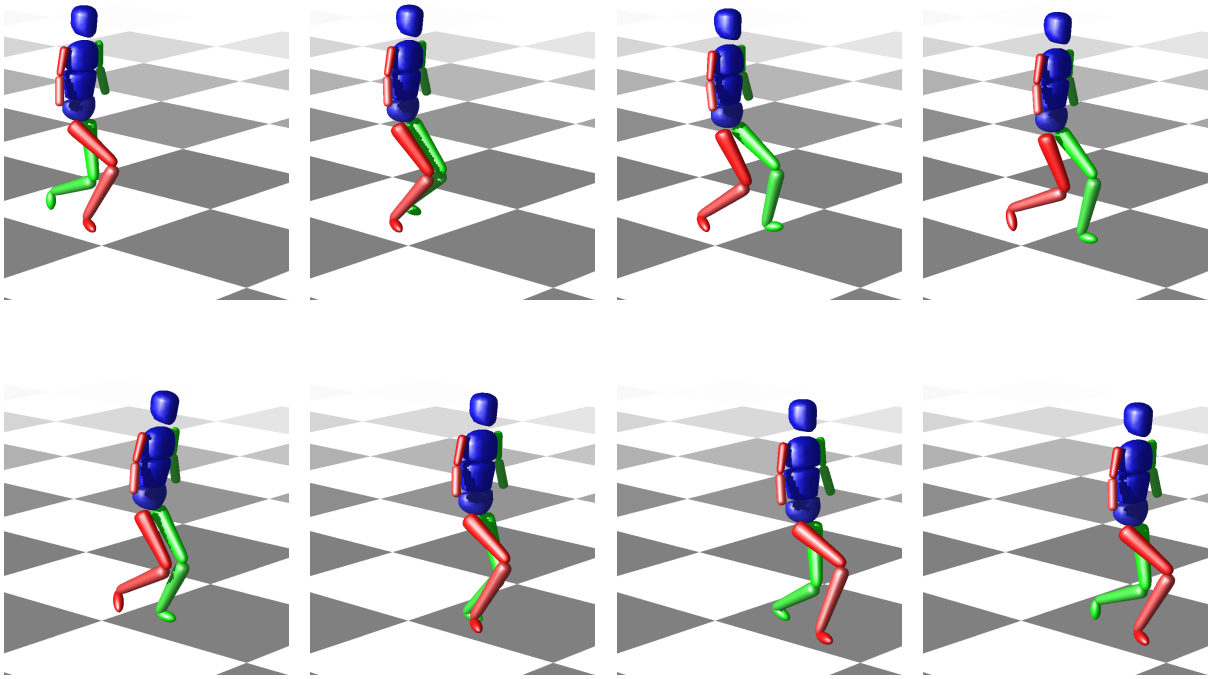

**Figure 10.2:** Visualization of synthesized CP gait as solution of the multi-stage OCP (6.35) for setting denoted by S from Table 10.6. Snapshots are arranged in reading direction, with initial position at upper left corner, and final position at lower right corner.

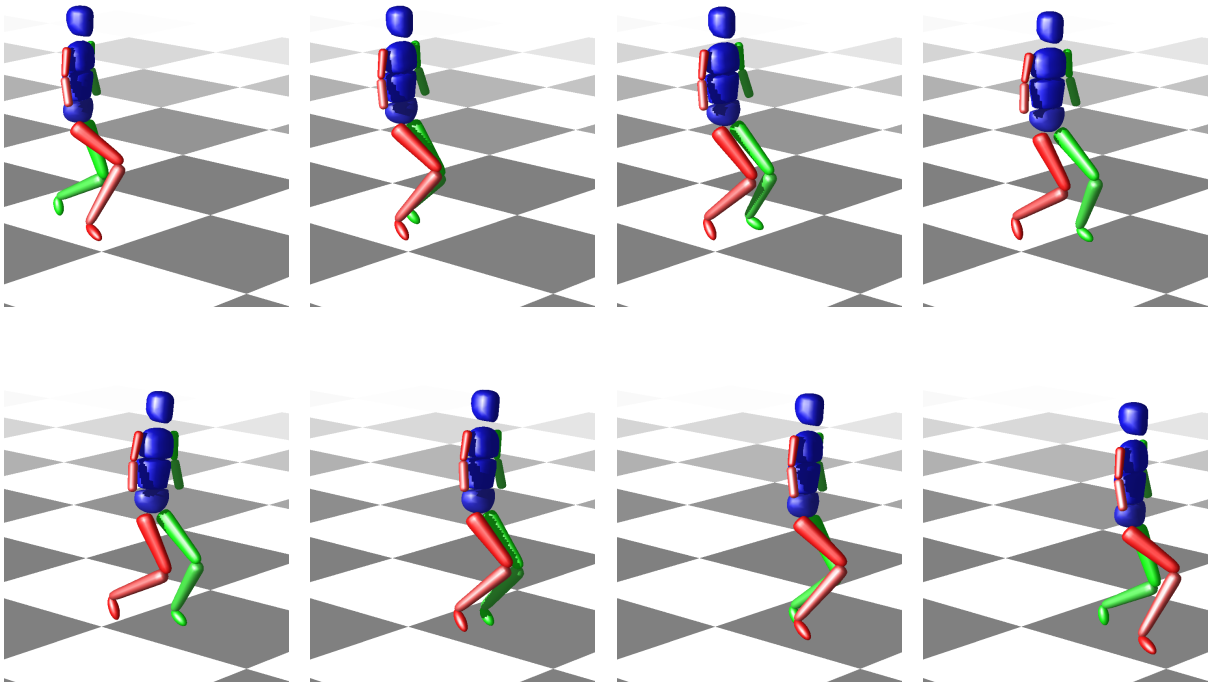

**Figure 10.3:** Visualization of synthesized CP gait as solution of the multi-stage OCP (6.35) for setting denoted by MW from Table 10.6. Snapshots are arranged in reading direction, with initial position at upper left corner, and final position at lower right corner.

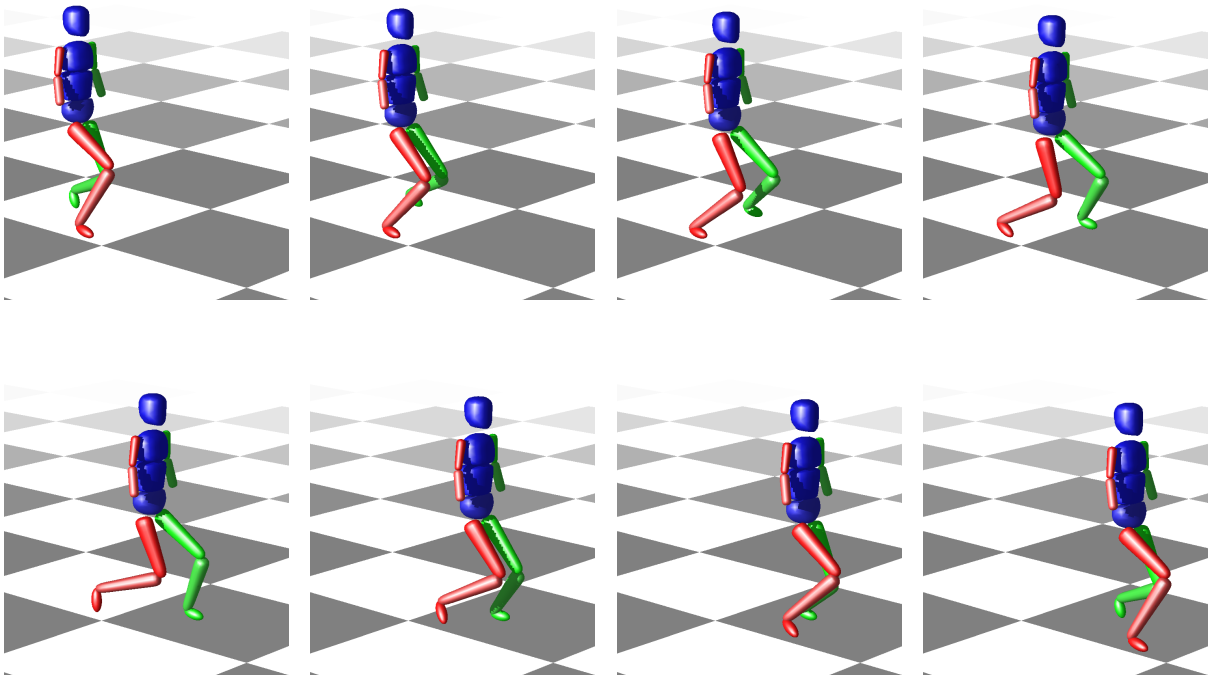

**Figure 10.4:** Visualization of synthesized CP gait as solution of the multi-stage OCP (6.35) for setting denoted by E from Table 10.6. Snapshots are arranged in reading direction, with initial position at upper left corner, and final position at lower right corner.

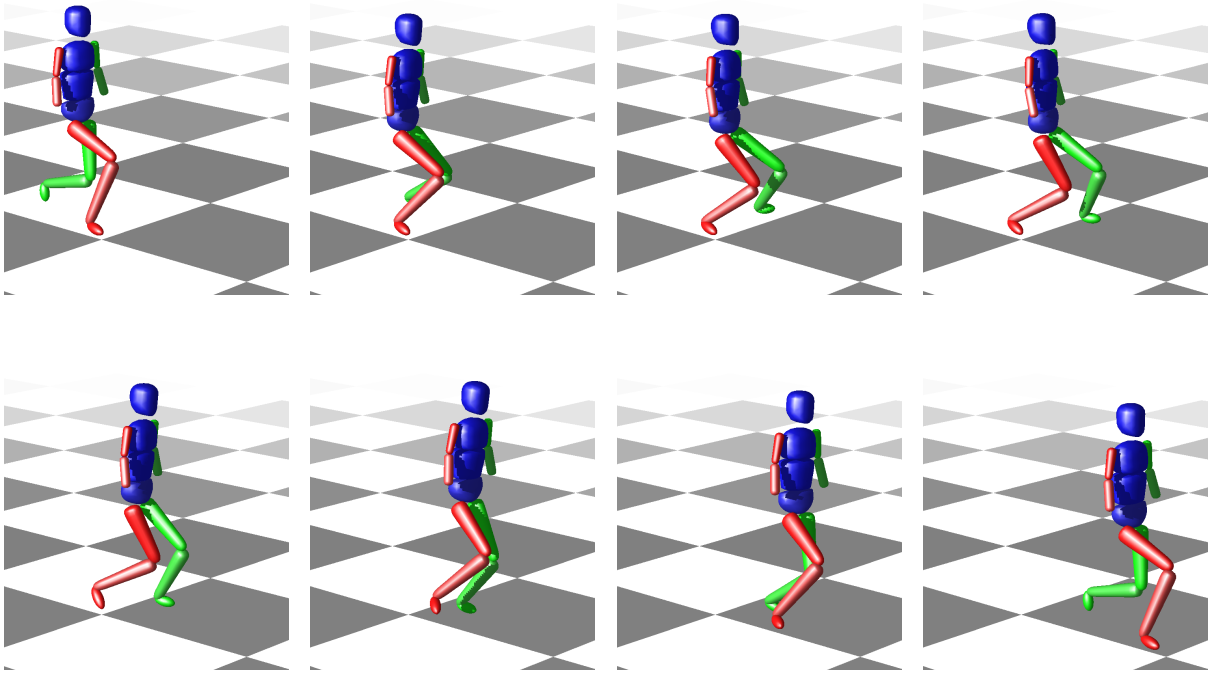

**Figure 10.5:** Visualization of synthesized CP gait as solution of the multi-stage OCP (6.35) for setting denoted by TD from Table 10.6. Snapshots are arranged in reading direction, with initial position at upper left corner, and final position at lower right corner.

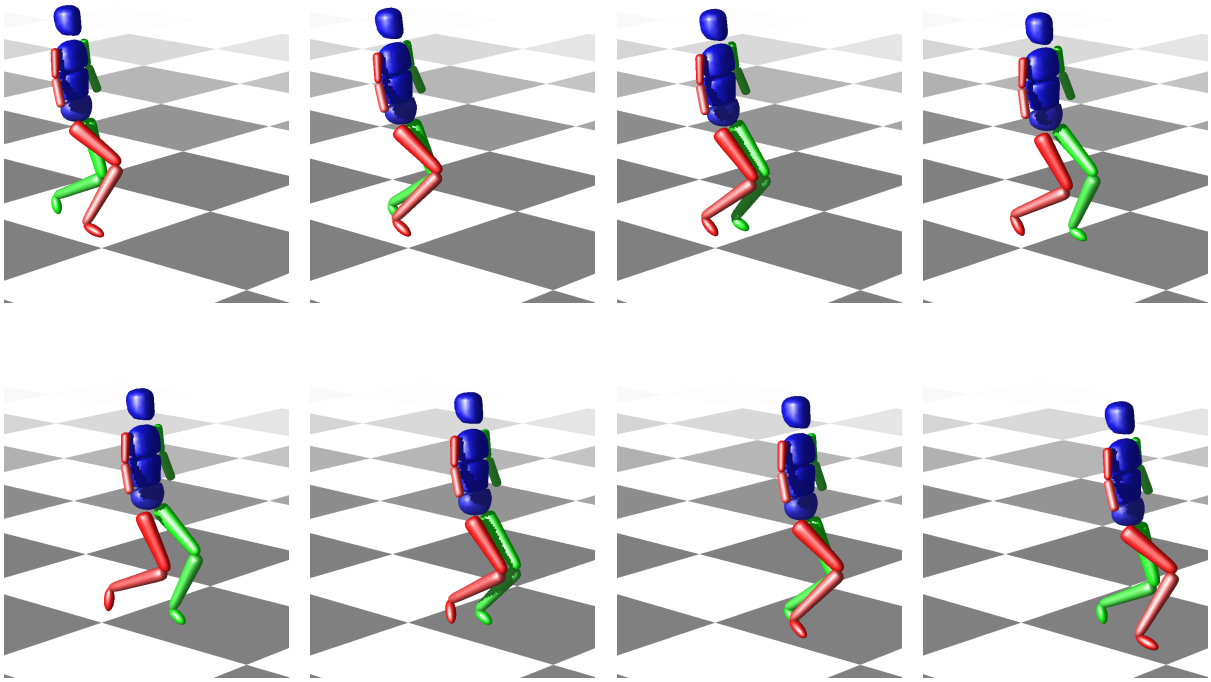

**Figure 10.6:** Visualization of synthesized CP gait as solution of the multi-stage OCP (6.35) for setting denoted by C from Table 10.6. Snapshots are arranged in reading direction, with initial position at upper left corner, and final position at lower right corner.

## 10.4 Case Study: Identification of Weights for Optimal CP Gait by Deep Neural Networks

In the case study of this section, we investigate the proposed approach from section 6.7 for the special case of varying only two instead of four objective weights when solving various OCPs (6.35). More specifically, the weight $\alpha_2$ that corresponds to the optimization criterion (6.39) minimizing the absolute mechanical work, and the weight $\alpha_3$ that corresponds to the criterion (6.40) minimizing the mechanical effort are considered. The other objective weights are fixed to the values $\alpha_1 = 10^3$ and $\alpha_4 = 10^{-1}$. In addition, for the varying objective weights with $\alpha_i = \alpha_i^{'} \cdot 10^{-1}$, $i = 2,3$ we ensure that

$$\alpha_2^{'} + \alpha_3^{'} = 1 \qquad \text{and} \qquad \alpha_i^{'} \geq 0,\ i = 2,3. \tag{10.1}$$

In the following descriptions we mainly consider weight $\alpha_3^{'}$ when solving various OCPs (6.35) to provide simulated training data for approximating the DNN (6.47). For varying values of objective weight $\alpha_3^{'}$, weight $\alpha_2^{'}$ is defined by (10.1).

Before the trained DNN $\hat{\boldsymbol{\Phi}}_{\text{DNN}}$ is used to obtain estimated objective weight $\hat{\boldsymbol{\alpha}} = \alpha_3^{'}$ for given measurements $\boldsymbol{\eta}$ in the offline phase of algorithm 4, in the following we give a detailed description on the simulation of the training data, and the DNN set-up and training in the online phase.

### 10.4.1 Simulation of Training Data

For training the approximated DNN, $n_S = 101$ simulated data pairs (6.46) are generated by solving OCPs (6.35) with varying weights $\alpha_3^{'}$ with the software package MUSCOD-II in the setting as described in section 10.1. We solve OCPs with

$$\alpha_3^{'} = (s-1)\Delta_\alpha, \qquad \text{for } s = 1,\dots,n_S, \tag{10.2}$$

and grid width $\Delta_\alpha = 0.01$. To apply the supervised learning approach of section 6.7, for each weight $\boldsymbol{\alpha}_{(s)}^* = (\alpha_3^{'})_{(s)} \in [0,1]$ defined by (10.2) we consider $\boldsymbol{q}_{(s)}^* \in \mathbb{R}^{n_{\boldsymbol{\eta}}}$ with $n_{\boldsymbol{\eta}} = 340$ for $s \in \{1,\dots,n_S\}$ and use both quantities for constructing $n_S$ data pairs (6.46) for training. Therein, the vector $\boldsymbol{q}_{(s)}^*$ combines only a chosen subset of all generalized coordinates that are the solutions of OCPs (6.35) evaluated at given time points for varying objective weight. We choose a subset, where the focus lies on the translation of the pelvis joint (in $y$- and $z$- direction) and all rotations of the pelvis, the rotations in the hip joints (around $x$- and $z$- axes), and the rotations around the patient-specific knee axes. The other generalized coordinates are not considered to reduce the dimension of the simulated data without loosing too much information. In the future, the application of an attention mechanism within the approximated DNN can be included, such as, e.g., in the prominent publication "Attention is All you Need" by Vaswani et al. [168].

### 10.4.2 DNN Set-Up and Training

The DNN set-up and training described in this section is performed within an own `python` implementation using the KERAS library [35]. For the DNN approximation we choose a fully-connected feedforward network $\hat{\boldsymbol{\Phi}}_{\text{DNN}} : \mathbb{R}^{n_{\boldsymbol{\eta}}} \times \mathbb{R}^{n_{\boldsymbol{\theta}}} \to [0,1]$ with four layers, where the inputs are first normalized. The first three hidden layers are all dense with node dimensions $3 \cdot n_{\boldsymbol{\eta}}, 2 \cdot n_{\boldsymbol{\eta}}$, and $1 \cdot n_{\boldsymbol{\eta}}$. As activation, the ***ReLU*** function defined by (6.48) is used. To prevent overfitting we use a so called *dropout layer* in KERAS with a rate of 0.1 that drops out some inputs randomly during training. For the output layer we choose a *sigmoid* activation in KERAS without bias term instead of a linear transformation, because the sigmoid function always returns a value between 0 and 1, and, hence, the output lies in $[0,1]$ as expected. The simulated data pairs generated as previously described in subsection 10.4.1 are used to train, validate and test the DNN. The validation of the DNN model is performed in each iteration of the optimization method applied on problem (6.47) on the actual estimated network parameter set to provide a performance indication during training. The test set is then used for investigating the

performance of the already trained DNN. For training we use 73 data pairs, for validation 18, and for testing 10, which are selected randomly. Note that the three chosen index sets, which correspond to the training, validation, and testing sets are disjoint. As *loss* function (or objective function) the *mean squared error* option in KERAS is chosen, such that problem (6.49) is minimized during training. As optimizer the ADAMmethod [101] is set with a *learning rate* (or step size) of 0.001 and 100 *epochs* (or iterations).

If we denote a subset of the index set of all available simulated data pairs $\mathcal{S} = \{1, \dots, n_S\}$ with $\mathcal{S}' \subseteq \mathcal{S}$, the mean squared error on the given set is defined as

$$\frac{1}{|\mathcal{S}'|} \sum_{s \in \mathcal{S}'} \| \hat{\boldsymbol{\Phi}}_{\text{DNN}}(\boldsymbol{q}^*_{(s)}; \boldsymbol{\theta}) - \boldsymbol{\alpha}^*_{(s)} \|_2^2. \tag{10.3}$$

With randomly chosen training, validation and testing data out of all simulated data pairs, the solution of (6.49) on the training set for the estimation of the DNN (6.47) resulted in a loss function of $2.2759 \cdot 10^{-4}$ (mean squared error (10.3)) for the determined network parameter vector $\hat{\boldsymbol{\theta}}$. For the validation set a mean squared error (10.3) of $2.9827 \cdot 10^{-4}$ is achieved. This results indicate that an appropriate fit of the DNN model is accomplished, with only a slightly higher value of (10.3) for the validation. This is confirmed by the testing of the estimated DNN. Therein a small mean squared error (10.3) with a value of $7.1121 \cdot 10^{-4}$ indicates that the estimated weights $\hat{\boldsymbol{\alpha}}_{(s)} = \hat{\boldsymbol{\Phi}}_{\text{DNN}}(\boldsymbol{q}^*_{(s)}; \hat{\boldsymbol{\theta}})$ are close to the true values $\boldsymbol{\alpha}^*_{(s)}$, and suggests that our DNN model performs well on our testing set of the simulated data pairs. Although only providing a relatively small number of simulated data pairs compared to the dimension of the network parameters in the fully-connected feedforward network, the estimated DNN model provides reasonable results for the simulated data pairs. Note, that so far no noise in the simulated data is considered.

### 10.4.3 Identification of Weights via Trained DNN

In this section we synthesize measurements $(\boldsymbol{\eta})_{(s)} \in \mathbb{R}^{n_{\boldsymbol{\eta}}}, s \in \mathcal{S}'_{\text{test}}$ from the testing set denoted by $\mathcal{S}'_{\text{test}} \subset \mathcal{S}$ with independent, additive, and normally distributed errors $\boldsymbol{\varepsilon}_{(s)} \in \mathbb{R}^{n_{\boldsymbol{\eta}}}$ with zero mean and variances calculated for each component of the vector separately under consideration of the standard deviations for each component in the data set. The testing set is the same randomly selected set as previously used in subsection 10.4.2. But at this point we investigate the performance of the previously estimated DNN on synthesized measurements with added noise. We generate 10 distinct measurement vectors $\boldsymbol{\eta}_{(s)} \in \mathbb{R}^{n_{\boldsymbol{\eta}}}$ for a subset of all generalized coordinates evaluated at specific time points by applying

$$\boldsymbol{\eta}_{(s)} = \boldsymbol{q}^*_{(s)} + \boldsymbol{\varepsilon}_{(s)}, \qquad \text{for } s \in \mathcal{S}'_{\text{test}}. \tag{10.4}$$

To obtain the corresponding estimated weights $\hat{\boldsymbol{\alpha}}_{(s)}$ we propagate each simulated measurement vector $\boldsymbol{\eta}_{(s)}$ for $s \in \mathcal{S}'_{\text{test}}$ through the trained DNN. In Table 10.9 the results are summarized, where all weights are estimated reasonably well by the DNN. The solution vector $\boldsymbol{q}^*_{(s)}$ of the OCP (6.35) computed for true weight $\boldsymbol{\alpha}^*_{(s)}$ is then compared to the solution vector denoted by $\hat{\boldsymbol{q}}_{(s)} \in \mathbb{R}^{n_{\boldsymbol{\eta}}}$ comprising generalized coordinates of the solution of the OCP (6.35) with estimated weight $\hat{\boldsymbol{\alpha}}_{(s)}$ for each test data pair. This is evaluated by the relative error defined as

$$\boldsymbol{q}^{\text{rel}}_{(s)} = \frac{\left\| \hat{\boldsymbol{q}}_{(s)} - \boldsymbol{q}^*_{(s)} \right\|_2}{\| \boldsymbol{q}^*_{(s)} \|_2}, \qquad \text{for } s \in \mathcal{S}'_{\text{test}}. \tag{10.5}$$

This relative error for the reconstruction of the generalized coordinates is listed in Table 10.9. It can be observed that the resulting DNN performs reasonably well on the testing set with added noise.

**Table 10.9:** Result of identification of weights $\hat{\boldsymbol{\alpha}}_{(\mathrm{s})}$ in the CP gait model via DNN for one particular choice of the testing data set with index s and comparison to true values denoted by $\boldsymbol{\alpha}^*_{(\mathrm{s})}$. The discrepancy of both values is listed, as well as the relative error $\boldsymbol{q}^{\mathrm{rel}}_{(\mathrm{s})}$ defined by (10.5) of the reconstruction of the generalized coordinates by solving the OCPs (6.35) for the corresponding weights.

| Index s | True Value $\boldsymbol{\alpha}^*_{(\mathrm{s})}$ | Estimated Value $\hat{\boldsymbol{\alpha}}_{(\mathrm{s})}$ | Discrepancy $\lvert\hat{\boldsymbol{\alpha}}_{(\mathrm{s})} - \boldsymbol{\alpha}^*_{(\mathrm{s})}\rvert$ | Rel. Error Reconstr. $\boldsymbol{q}^{\mathrm{rel}}_{(\mathrm{s})}$ |
|---|---|---|---|---|
| 101 | 1.000 | 0.984 | $1.6 \cdot 10^{-2}$ | 1.33% |
| 82 | 0.810 | 0.814 | $4.0 \cdot 10^{-3}$ | 0.68% |
| 20 | 0.190 | 0.203 | $1.3 \cdot 10^{-2}$ | 1.30% |
| 17 | 0.160 | 0.157 | $3.0 \cdot 10^{-3}$ | 0.84% |
| 8 | 0.070 | 0.055 | $1.5 \cdot 10^{-2}$ | 2.54% |
| 35 | 0.340 | 0.350 | $1.0 \cdot 10^{-2}$ | 3.20% |
| 72 | 0.710 | 0.650 | $6.0 \cdot 10^{-2}$ | 2.89% |
| 41 | 0.400 | 0.428 | $2.8 \cdot 10^{-2}$ | 2.19% |
| 76 | 0.750 | 0.719 | $3.1 \cdot 10^{-2}$ | 1.00% |
| 9 | 0.080 | 0.073 | $7.0 \cdot 10^{-3}$ | 2.51% |

### 10.4.4 Summary and Outlook

In this section, we provide a first case study for the proposed supervised learning approach from section 6.7 for the identification of objective weight $\alpha_2$ that corresponds to the optimization criterion (6.39) minimizing the absolute mechanical work, and the weight $\alpha_3$ that corresponds to the criterion (6.40) minimizing the mechanical effort. The other objective weights as well as the usually unknown model parameters in the OCP (6.35) are fixed to specific values. Already for relatively small data sets for training, validation, and testing of the DNN as described in subsection 10.4.2, we could achieve reasonable results in subsection 10.4.3 for the identification of objective weights in the derived CP gait model from chapter 6, which is promising for future further development of this approach. Some limitations of the DNN based supervised learning approach have to be considered. For example, individual objective weights that correspond to the recorded gait of a CP patient can only be estimated, if *adequate* simulated data for training is available. In our first case study, where we exploited OCPs of the form (6.35) on a relatively small data set and for only two objective weights and their relative impact on the simulated CP gaits, so far, the trained DNN cannot be used for the data provided by the HEIDELBERG MOTIONLAB [173]. It should be considered that different combinations of objective weights may lead to the same (or very similar) measurements – especially if the optimization criteria are not chosen well. In the supervised learning approach we assumed that there exist a continuous function that maps measurements onto objective weights, see section 6.7. This may lead to difficulties in the identification of weights via DNNs and should be further investigated.

In future research the supervised learning approach from section 6.7 can be applied onto a broader range of simulated data sets. Then it can be investigated in more detail, if the reconstructed CP gait from section 6.3 can be reproduced within this approach, and if the gait of a CP patient can be classified. Together with the DISIMFAS for solving the Bilevel Inverse OCP (6.42) for a more detailed characterization of the CP gait model of a patient including the identification of usually unknown model parameters, classification of the gait as well as information for intervention planning for the physicians can be provided.

# Conclusion and Outlook

We developed mathematical models and numerical methods for solving Bilevel Inverse OCPs of constrained biomechanical multibody system models. With the results achieved in this thesis we can contribute to the establishment of a classification and diagnosis scheme for the pathological gait of CP patients. For this purpose we have developed a suitable biomechanical rigid multibody system for a patient with CP. We have shown that the underlying dynamics can capture the main characteristics of the pathological gait. Because of the common assumption that the human gait is a result of a person's decision guided by certain optimization criteria and constraints we have derived an OC model for the gait of a CP patient, where the dynamics serve as constraints.

A calibrated patient-specific gait model can be determined under consideration of given motion capture data from the HEIDELBERG MOTIONLAB. To achieve this goal, we have formulated a Bilevel Inverse OCP: on the upper level of this bilevel optimization problem we have a PE Problem, constrained by the lower level parametrized OCP describing the gait of a patient with CP. However, Bilevel Inverse OCPs are challenging and difficult to solve. We developed a new numerical method for solving these kinds of problems, where information from given measurements can be integrated by retaining a fixed structure of the active inequality constraints. With the newly developed software package PARDYNOPT we created a highly efficient implementation of the proposed DISIMFAS by exploiting given structures. The sustainable software establishment allows for future extensions and implementations of other mathematical methods by reusing already available software modules. The successful application of the DISIMFAS implemented in PARDYNOPT in selected case studies – a basic human-like walking motion included – gives hope for future investigations for the new derived CP gait model to provide a non-invasive tool for the physicians to see what happens *inside* the patient as our ultimate goal. A first successful study of a supervised learning approach based on DNNs for the identification of optimal weights in our developed CP gait model can support this intention. may also be an alternative, feasible approach for this aim.

The pathological gait of CP patients is still an ongoing field of research. Because our derived CP gait model is easily adjustable to other patients, as well as healthy subjects, with the derived mathematical method for solving Bilevel Inverse OCPs, this will allow to draw conclusions on the locomotor system condition of patients in the future by routinely obtainable motion capture data of their gait. Comparison of the optimization criteria and other individual parameters of this model of healthy subjects and CP patients will allow investigating the pathological condition and classify and stratify CP patients for surgery and alternative therapy management according to yet to be identified criteria. This is especially important because the exact causalities of the pathological gait of CP patients are still unknown and no general, factual approach to surgical intervention exists.

# Appendix A Software Package: ParDynOpt

## A.1 Selected Initialization Methods in ParDynOpt

**Listing A.1:** Initialization methods in class `ProblemMs`

```
/**
 * \brief Initialize bounds on differential states.
 * \param xd_lb       lower bounds
 * \param xd_ub       upper bounds
 */
void ProblemMs::initialize_states_diff_bounds(dbl_t* xd_lb,
                                              dbl_t* xd_ub);

/**
* \brief Initialize initial values of differential states.
* \param xd_s         initial values
* \param xd_free_s    boolean array, true indicates free variables
* \param idx_stage    stage index
*/
void ProblemMs::initialize_states_diff_initial_values(dbl_t* xd_s,
                                                      bool_t* xd_free_s,
                                                      Index idx_stage = 0);

/**
* \brief Initialize final conditions on differential states.
* \param xd_f         final values
* \param xd_free_f    boolean array, true indicates free variables
* \param idx_stage    stage index
*/
void ProblemMs::initialize_states_diff_final_condition(dbl_t* xd_f,
                                                       bool_t* xd_free_f,
                                                       Index idx_stage);

/**
* \brief Initialize control parameters with same values on all nodes.
* \param q_ini        initial guesses
* \param q_lb         lower bounds
* \param q_ub         upper bounds
* \param q_free       boolean array, true indicates free variables
*/
void ProblemMs::initialize_control_parameters(dbl_t* q_ini,
                                              dbl_t* q_lb,
                                              dbl_t* q_ub,
                                              bool_t* q_free = nullptr);

/**
* \brief Initialize model parameters.
* \param mp_ini       initial guesses
* \param mp_lb        lower bounds
* \param mp_ub        upper bounds
* \param mp_free      boolean array, true indicates free variables
*/
void ProblemMs::initialize_model_parameters(dbl_t* mp_ini,
                                            dbl_t* mp_lb,
```

```
                                          dbl_t* mp_ub,
                                          bool_t* mp_free = nullptr);

/**
* \brief Initialize duration parameters.
* \param dp_ini       initial guesses
* \param dp_lb        lower bounds
* \param dp_ub        upper bounds
* \param dp_free      boolean array, true indicates free variables
*/
void ProblemMs::initialize_duration_parameters(dbl_t* dp_ini,
                                               dbl_t* dp_lb,
                                               dbl_t* dp_ub,
                                               bool_t* dp_free = nullptr);

/**
* \brief Initialize all measurements related to one stage.
* \param idx_stage    stage index
* \param meas         measurements
* \param sigma        standard deviation
*/
void ProblemMsParaOcp::initialize_all_measurements_on_all_nodes_of_stage(Index idx_stage,
                                                                         DMat* meas,
                                                                         Vec* sigma = nullptr);
```

## A.2 A Complete Example - The Rocket Car

In this section we give a complete example how to set up an OCP and a Bilevel Inverse OCP of the rocket car example as described in more detail in chapter 8. For convenience, we restate both problem formulations and start with the OCP as follows:

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}} \quad \boldsymbol{\Phi}(\cdot) = \alpha_1 \cdot d_1 - x_2(1) \tag{A.1a}$$

$$\text{s.t.} \quad \dot{x}_0(t) = d_1 x_1(t), \qquad t \in \mathcal{T}, \tag{A.1b}$$

$$\dot{x}_1(t) = d_1 \left(u_0(t) - p_0 x_1^2(t)\right) / \left(x_2(t) + \gamma\right), \quad t \in \mathcal{T}, \tag{A.1c}$$

$$\dot{x}_2(t) = -d_1 \varrho u_0^2(t), \qquad t \in \mathcal{T}, \tag{A.1d}$$

$$x_0(0) = 0, x_1(0) = 0, x_2(0) = 1, \tag{A.1e}$$

$$x_0(1) = 10, x_1(1) = 1, \tag{A.1f}$$

$$-10 \le x_0(t) \le 10, \qquad t \in \mathcal{T}, \tag{A.1g}$$

$$-10 \le x_1(t) \le 10, \qquad t \in \mathcal{T}, \tag{A.1h}$$

$$0 \le x_2(t) \le 10, \qquad t \in \mathcal{T}, \tag{A.1i}$$

$$-1 \le u_0(t) \le 1, \qquad t \in \mathcal{T}, \tag{A.1j}$$

$$0 \le d_1, \tag{A.1k}$$

with the quantities defined in Table 8.1, the objective weight $\alpha_1$ set to 0.06, and the model parameters $p_0$ to 0.1, $\gamma$ to 0.1, and $\varrho$ to 0.1. The corresponding Bilevel Inverse OCP can be formulated as

$$\min_{\substack{\alpha_1, p_0, \\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{d}}} \quad \boldsymbol{\Psi}(\cdot) = \frac{1}{2} \sum_{\mathrm{n}=0}^{\mathrm{n_m}-1} \sum_{\mathrm{k}=0}^{2} \frac{\left(x_{\mathrm{k}}(t_{\mathrm{n}}^{\mathrm{m}}) - \eta_{\mathrm{nk}}\right)^2}{\sigma_{\mathrm{k}}^2}$$

$$\text{s.t.} \quad (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \text{ solve OCP (A.1)},$$

$$0 \le \alpha_1 \le 10,$$

$$-20 \le p_0 \le 20,$$

where the upper level PE is constrained by the lower level OCP (A.1) and the objective weight $\alpha_1$ and the model parameter $p_0$ have to be determined by fitting the model to given measurements. For both problems an object of the `ProblemDescription` class has to be instantiated as described in Listing A.2.

**Listing A.2:** ProblemDescription for rocket car example

```
1
2  /*****************************************************************************/
3  //****** Problem Description ******//
4  /*****************************************************************************/
5
6  Dimension num_stages = 1;
7  Dimension* num_nodes = new Dimension[num_stages];
8  num_nodes[0] = 20;
9
10 Model** model_array = new Model*[num_stages];
11 Model* model = new ModelSolvInd("./librocket_car_dyn_model_para_ocp.so",
12                                  "rocket_car_para_ocp");
13 model_array[0] = model;
14
15 StageType* stage_type = new StageType[num_stages];
16 stage_type[0] = StageType::dynamic;
17
18 ProblemDescription* problem_desc = new ProblemDescription(model_array,
19                                                           num_stages,
20                                                           stage_type,
21                                                           num_nodes);
22
23 problem_desc->finalize_problem_description();
```

Beside the number of model stages with its type and the number of nodes per stage, an array of dynamic models for all stages have to be defined. In the actual implementation of PARDYNOPT dynamic models can be described via SOLVIND dynamic model descriptions. The user has to implement a class which inherits from the parent class `IDynamicModelDescription` and provides a constructor as shown in Listing A.3. In the constructor, the dimensions of differential states, controls, model parameters and duration parameters are defined. Furthermore, all previously implemented model functions are set with the corresponding output dimensions. Such a SOLVIND model can be enabled within the PARDYNOPT software package by calling the constructor of the `ModelSolvInd` class and providing the corresponding shared library, here the `"./librocket_car_dyn_model_para_ocp.so"` with its name `"rocket_car_para_ocp"` is used for the registration in SOLVIND.

**Listing A.3:** SOLVIND model for the rocket car example.

```
1  #include <cmath>
2  #include <cstdlib>
3  #include <sstream>
4
5  #include "sonic++.h"
6
7  #include "ind_compile_time_info.hpp"
8  #include "ind_dyn_model_description.hpp"
9
10 using namespace SolvInd;
11
12 template <typename T>
13 svLong rocket_car_rhs(TArgs_ffcn<T>& args, TDependency* depends)
14 {
15   T x, v, m;
16   T u;
```

```
17    T pfr;
18
19    double pcm = 0.1;
20    double pfc = 0.1;
21
22    // STATE: position
23    x = args.xd[0];
24    // STATE: speed
25    v = args.xd[1];
26    // STATE: fuel mass
27    m = args.xd[2];
28    // CONTROL: control force
29    u = args.u[0];
30    // PARAMETER: friction
31    pfr = args.p[0];
32
33    args.rhs[0] = v;
34    args.rhs[1] = u / (m + pcm) - pfr * v * v / (m + pcm);
35    args.rhs[2] = -pfc * u * u;
36
37    return 0;
38  }
39
40  /*****************************************************************************/
41
42  template <typename T>
43  static svLong rocket_car_mfcn(SolvInd::TArgs_mfcn<T>& args, TDependency* depends)
44  {
45    args.mval[0] = args.p[1] * args.t[0] - args.xd[2];
46    return 0;
47  }
48
49  /*****************************************************************************/
50
51  template <typename T>
52  static svLong rocket_car_lsq(SolvInd::TArgs_lsqfcn<T>& args, TDependency* depends)
53  {
54    args.res[0] = args.xd[0];
55    args.res[1] = args.xd[1];
56    args.res[2] = args.xd[2];
57    return 0;
58  }
59
60  /*****************************************************************************/
61
62  class myDynModel : public IDynamicModelDescription
63  {
64   public:
65    myDynModel(const std::string& options = "");
66
67   private:
68    static FFactory createMe;
69    static IDynamicModelDescription::TRegisterTrigger myTrigger;
70  };
71
72  /*****************************************************************************/
73
74  IDynamicModelDescription::TRegisterTrigger myDynModel::myTrigger(
75                                                  std::string("rocket_car_para_ocp"),
76                                                  &myDynModel::createMe);
```

```
77
78  /*****************************************************************************/
79
80  IDynamicModelDescription* myDynModel::createMe(const std::string& options)
81  {
82    return new myDynModel(options);
83  }
84
85  /*****************************************************************************/
86
87  myDynModel::myDynModel(const std::string& options)
88    : IDynamicModelDescription()
89  {
90    m_dims.dim[Component_XD] = 3;
91    m_dims.dim[Component_P]  = 2;
92    m_dims.dim[Component_U]  = 1;
93    m_dims.dim[Component_H]  = 1;
94
95    m_dims.nTrajectories = 1;
96
97    m_functions.setFunction<Function_ffcn>(&rocket_car_rhs<double>);
98    m_functions.setFunction<Function_ffcn>(&rocket_car_rhs<adouble>);
99
100   m_functions.setFunction<Function_mfcn>(&rocket_car_mfcn<double>);
101   m_functions.setFunction<Function_mfcn>(&rocket_car_mfcn<adouble>);
102
103   m_functions.setFunction<Function_lsqfcn_s>(&rocket_car_lsq<double>);
104   m_functions.setFunction<Function_lsqfcn_s>(&rocket_car_lsq<adouble>);
105
106   m_functions.setFunction<Function_lsqfcn_i>(&rocket_car_lsq<double>);
107   m_functions.setFunction<Function_lsqfcn_i>(&rocket_car_lsq<adouble>);
108
109   m_functions.setFunction<Function_lsqfcn_e>(&rocket_car_lsq<double>);
110   m_functions.setFunction<Function_lsqfcn_e>(&rocket_car_lsq<adouble>);
111
112
113   m_fcnOutputDims.dim[Function_mfcn] = 1;
114   m_fcnOutputDims.dim[Function_ffcn] = m_dims.dim[Component_XD];
115
116   m_fcnOutputDims.dim[Function_lsqfcn_s] = m_dims.dim[Component_XD];
117   m_fcnOutputDims.dim[Function_lsqfcn_i] = m_dims.dim[Component_XD];
118   m_fcnOutputDims.dim[Function_lsqfcn_e] = m_dims.dim[Component_XD];
119
120 }
```

By default the Direct Multiple Shooting Method is implemented on a normalized equidistant multiple shooting grid defined by the given number of nodes with the SOLVIND interface for piecewise constant control discretization. Furthermore, the `NodeEvaluatorSolvInd` implementation for function evaluation and derivative generation within the SOLVIND suite and the `NlpSolverIpopt` interface for IPOPT is enabled. After calling `finalize_problem_description()`, a `ProblemDescription` object is initialized completely and can be used to set up an OCP or a Bilevel Inverse OCP in PARDYNOPT for the rocket car example.

### A.2.1 OCP for Rocket Car Example

In Listing A.4 we give an example how to set up a `ProblemMsOcp` object and start the solution process by calling `solve()` after finalized initialization.

**Listing A.4:** ProblemMsOcp for rocket car example

```
#include "utils/types.hxx"
#include "model/model_solvind.hxx"
#include "problem/problem_ms_ocp.hxx"
using namespace pdo;

int main(void)
{
  /*****************************************************************************************/
  //****** Problem Description ******//
  /*****************************************************************************************/

  // include listing with problem description for rocket car example


  /*****************************************************************************************/
  //****** Problem OCP with Multiple Shooting ******//
  /*****************************************************************************************/

  Problem* problem = new ProblemMsOcp(problem_desc);

  /*****************************************************************************************/
  // initialize states xd - s, f, lb, ub, free_s, free_f
  dbl_t xd_s[] = { 0.0, 0.0, 1.0};
  dbl_t xd_f[] = {10.0, 1.0, 0.0};

  dbl_t xd_lb[]  = {-10.0, -10.0,   0.0};
  dbl_t xd_ub[]  = { 10.0,  10.0,  10.0};

  bool_t xd_free_s[] = {false,  false,  false};
  bool_t xd_free_f[] = {false,  false,  true};


  problem->initialize_states_diff_bounds(xd_lb, xd_ub);
  problem->initialize_states_diff_initial_values(xd_s, xd_free_s);
  problem->initialize_states_diff_final_condition(xd_f, xd_free_f);
  /*****************************************************************************************/
  // initialize control parameters - ini, lb, ub, free
  dbl_t q_ini[] = {1.0};
  dbl_t q_lb[]  = {-1.0};
  dbl_t q_ub[]  = { 1.0};

  problem->initialize_control_parameters(q_ini, q_lb, q_ub);
  /*****************************************************************************************/
  // initialize model parameters - ini, lb, ub, free
  dbl_t mp_ini[] = {0.1, 0.06};
  dbl_t mp_lb[]  = {-20.0, 0.};
  dbl_t mp_ub[]  = { 20.0, 10.};
  bool_t mp_free[] = {false, false};

  problem->initialize_model_parameters(mp_ini, mp_lb, mp_ub, mp_free);
  /*****************************************************************************************/
  // initialize duration parameters dp - ini, lb, ub, free
  dbl_t dp_ini[] = {1.0};
  dbl_t dp_lb[]  = {0.02};
  dbl_t dp_ub[]  = {10.0};
  bool_t dp_free[] = {true};

  problem->initialize_duration_parameters(dp_ini, dp_lb, dp_ub, dp_free);
  /*****************************************************************************************/

```

```
61    problem->finalize_initialization();
62
63    problem->solve();
64
65    /*****************************************************************************/
66 }
```

In the OCP we consider the model parameter $p_0$ set to 0.1 and the objective weight $\alpha_1$ to 0.06. They correspond to `mp_ini[] = 0.1, 0.06` in the PARDYNOPT framework and in the SOLVIND model to `p[0]` and `p[1]`, respectively. Note that the least-squares function as described in the SOLVIND model in Listing A.3 does not enter the OCP, but is an integral part in the following section, where this function describes the objective of the PE problem.

### A.2.2 Bilevel Inverse OCP for Rocket Car Example

Similar to the previous OCP in Listing A.5, we give an example how to set up a `ProblemMsParaOcpActiveSetFix` object to start the solution process for the Bilevel Inverse OCP with the DISIMFAS described in chapter 4 by calling `solve()` after finalized initialization.

**Listing A.5:** Solve ProblemMsParaOcpActiveSetFix instance for rocket car example

```
1  #include "utils/types.hxx"
2  #include "model/model_solvind.hxx"
3  #include "problem/problem_ms_para_ocp_active_set_fix.hxx"
4  using namespace pdo;
5
6  int main(void)
7  {
8    /*****************************************************************************/
9    //****** Problem Description ******//
10   /*****************************************************************************/
11
12   // include listing with problem description for rocket car example
13
14
15   /*****************************************************************************/
16   //****** Problem Bilevel Inverse OCP on Fixed Active Set with Multiple Shooting *******//
17   /*****************************************************************************/
18
19   Problem* problem = new ProblemMsParaOcpActiveSetFix(problem_desc);
20
21   /*****************************************************************************/
22   // initialize states xd - s, f, lb, ub, free_s, free_f
23   dbl_t xd_s[] = { 0.0, 0.0, 1.0};
24   dbl_t xd_f[] = {10.0, 1.0, 0.0};
25
26   dbl_t xd_lb[]  = {-10.0, -10.0,   0.0};
27   dbl_t xd_ub[]  = { 10.0,  10.0,  10.0};
28
29   bool_t xd_free_s[] = {false,  false,  false};
30   bool_t xd_free_f[] = {false,  false,  true};
31
32
33   problem->initialize_states_diff_bounds(xd_lb, xd_ub);
34   problem->initialize_states_diff_initial_values(xd_s, xd_free_s);
35   problem->initialize_states_diff_final_condition(xd_f, xd_free_f);
36   /*****************************************************************************/
37   // initialize control parameters - ini, lb, ub, free
38   dbl_t q_ini[] = {1.0};
```

```
39      dbl_t q_lb[]  = {-1.0};
40      dbl_t q_ub[]  = { 1.0};
41
42      problem->initialize_control_parameters(q_ini, q_lb, q_ub);
43      /*****************************************************************************************/
44      // initialize model parameters - ini, lb, ub, free
45      dbl_t mp_ini[] = {0.1, 0.5};
46      dbl_t mp_lb[]  = {-20.0, 0.};
47      dbl_t mp_ub[]  = { 20.0, 10.};
48      bool_t mp_free[] = {false, false};  // parameters are fixed in the OCP model!!!
49
50      problem->initialize_model_parameters(mp_ini, mp_lb, mp_ub, mp_free);
51      /*****************************************************************************************/
52      // initialize duration parameters dp - ini, lb, ub, free
53      dbl_t dp_ini[] = {1.0};
54      dbl_t dp_lb[]  = {0.02};
55      dbl_t dp_ub[]  = {10.0};
56      bool_t dp_free[] = {true};
57
58      problem->initialize_duration_parameters(dp_ini, dp_lb, dp_ub, dp_free);
59      /*****************************************************************************************/
60      // initialize measurements
61      Dimension num_lsq = model->mModelDescription->get_out_dim_lsq();
62      DMat* meas = new DMat(num_lsq, num_nodes[0]);
63
64      // get measurements by user defined function
65      get_measurements(meas);
66
67      // states can also be initialized with measurements if preferred
68      // problem->initialize_states_diff_values_with_measurements_on_all_nodes_of_stage(0, meas);
69
70      Vec* sigma = new Vec(num_lsq);
71      (*sigma)(0) = 0.05 * 5.0;
72      (*sigma)(1) = 0.05 * 1.0;
73      (*sigma)(2) = 0.05 * 0.9;
74
75      problem->initialize_all_measurements_on_all_nodes_of_stage(0, meas, sigma);
76      /*****************************************************************************************/
77      /*****************************************************************************************/
78
79      problem->finalize_initialization();
80
81      problem->solve();
82
83      /*****************************************************************************************/
84  }
```

After instantiation of a `ProblemMsParaOcpActiveSetFix` object by calling its constructor and passing the previously defined `ProblemDescription` object, all quantities related to the lower level OCP are treated in the same way as in the previous paragraph. In addition to this, given measurements have to be initialized in the Bilevel Inverse OCP, which enter the least-squares objective. Note, that although the parameters that include the objective weight $\alpha_1$ and the model parameter $p_0$ are free variables in the upper level PE problem, these are set to be fixed in the lower level OCP formulation of Listing A.5. In the Single-Stage Bilevel Inverse OCP in subsection 8.1.2, where we choose generated pseudo-measurements from an OCP with the objective weight $\alpha_1$ set to 0.06 and the model parameter $p_0$ set to 0.1, no active working set has to be specified by the user. In the Two-Stage and Three-Stage formulations as described in section 8.1, the fixed working sets, which are defined by the active lower or upper bounds of the dircretized control parameters, can be set in the `WorkingSet` of the instantiated problem. With the method `mFix->set_status_of_index(Index index, WorkingSetStatus`

`status)` the user can choose between `WorkingSetStatus::upper` and `WorkingSetStatus::lower`, see Listing A.6, where `index` defines the index of the NLP variables vector related to the OCP. In the future a more user friendly interface will be provided.

**Listing A.6:** Setting fixed working set.

```
#include "working_set/working_set_bounds.hxx"
using namespace pdo;

int main(void)
{
  ...
  // specify working set
  WorkingSet* bounds =
        dynamic_cast <ProblemMsParaOcpActiveSetFix*> (problem)->mWorkingSetModelBounds;
  bounds->mFix->set_status_of_index(0, WorkingSetStatus::upper);
  bounds->mFix->set_status_of_index(4, WorkingSetStatus::lower);

}
```

# Appendix B CP Gait Model

## B.1 Optimal Differential States and Controls of Dynamics Reconstruction

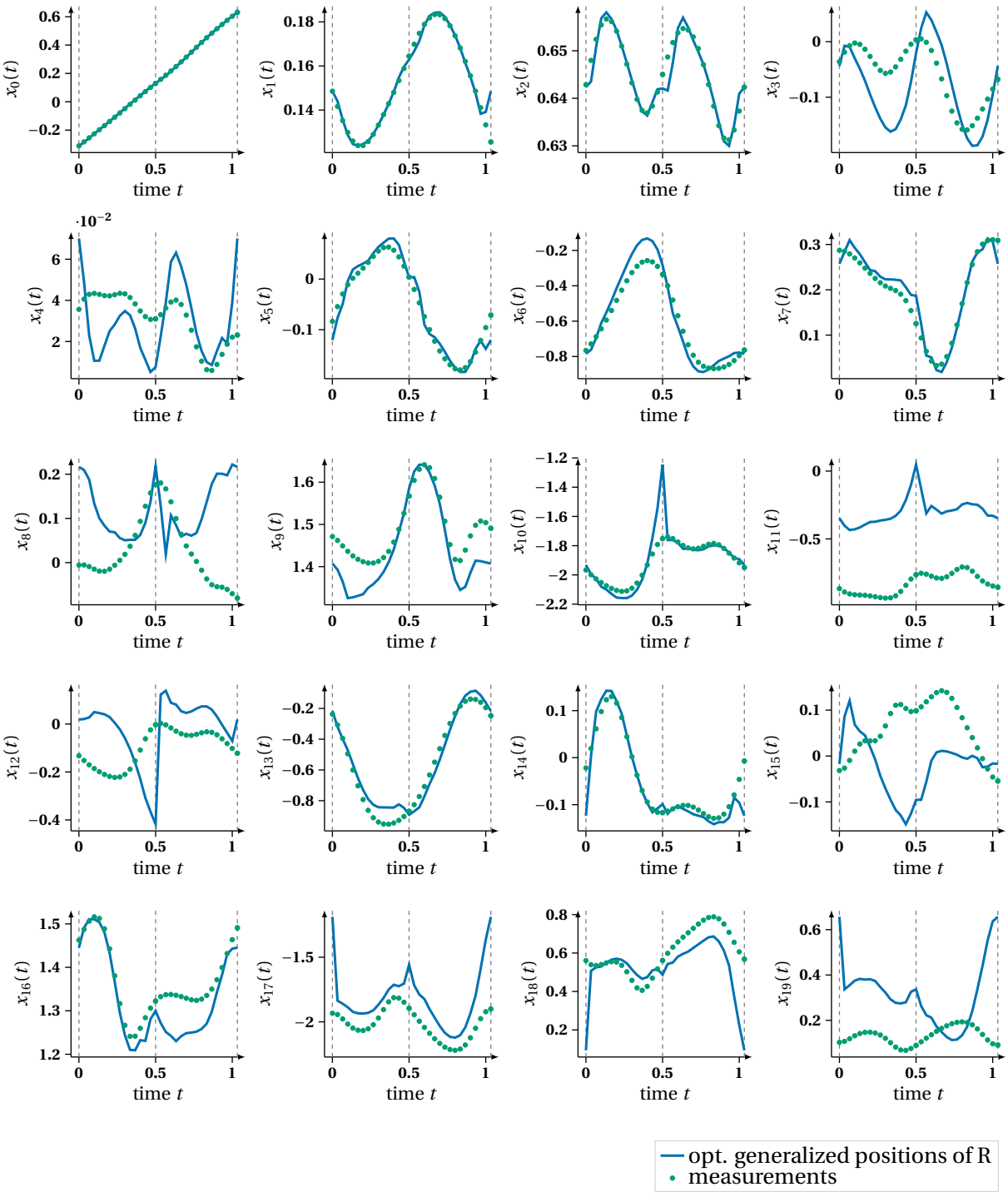


**Figure B.1:** This figure depicts the 20 optimal generalized positions $x_0(t) - x_{19}(t)$ in the solution of the least-squares OCP (6.18) denoted by R. This result is compared to some given measurements. These measurements include the processed Vicon angles and calculated knee angles from given Euler angles as described in section 10.2 and subsection 6.2.6.

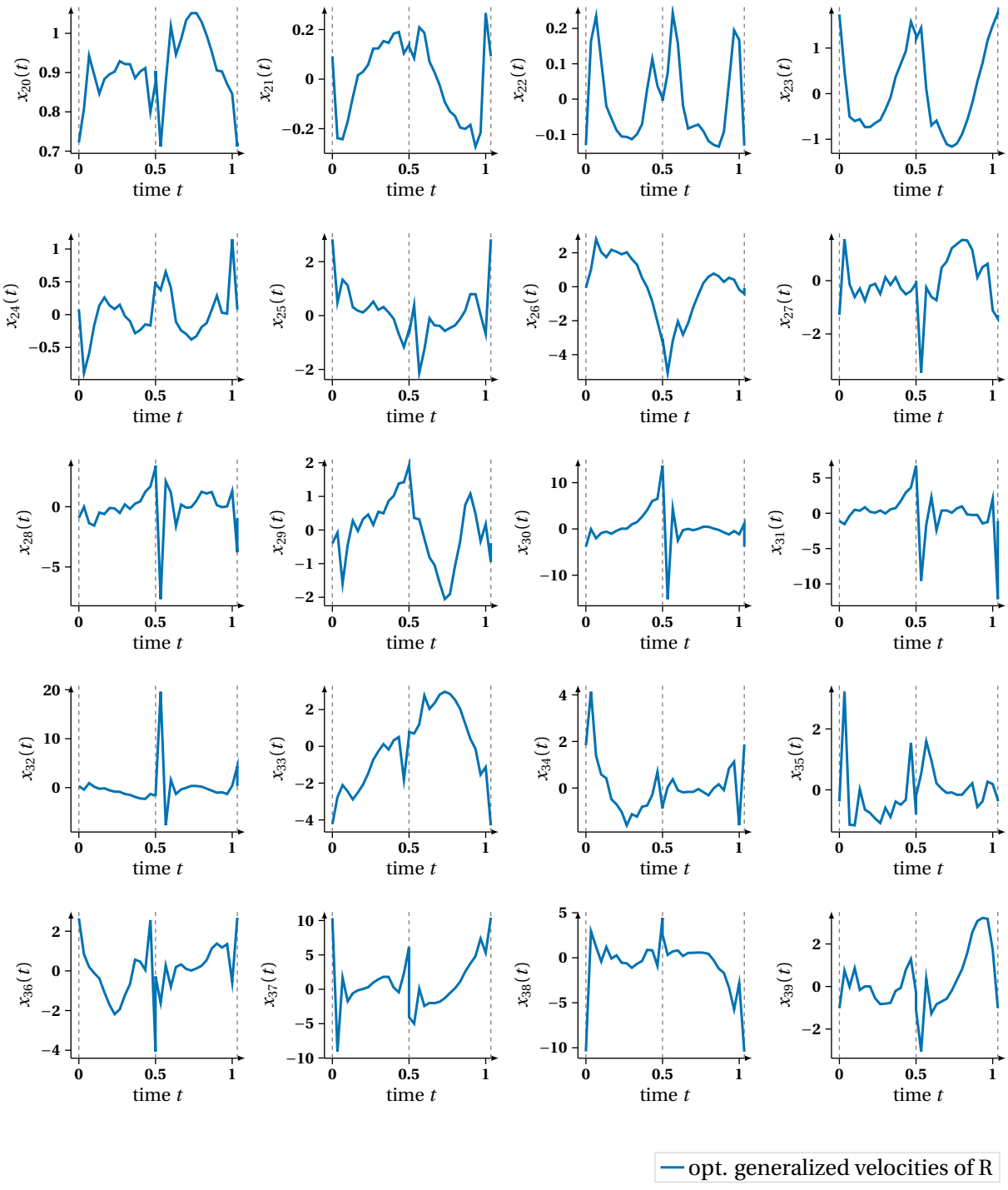


**Figure B.2:** This figure depicts the 20 optimal generalized velocities $x_{20}(t) - x_{39}(t)$ in the solution of the least-squares OCP (6.18) denoted by R.

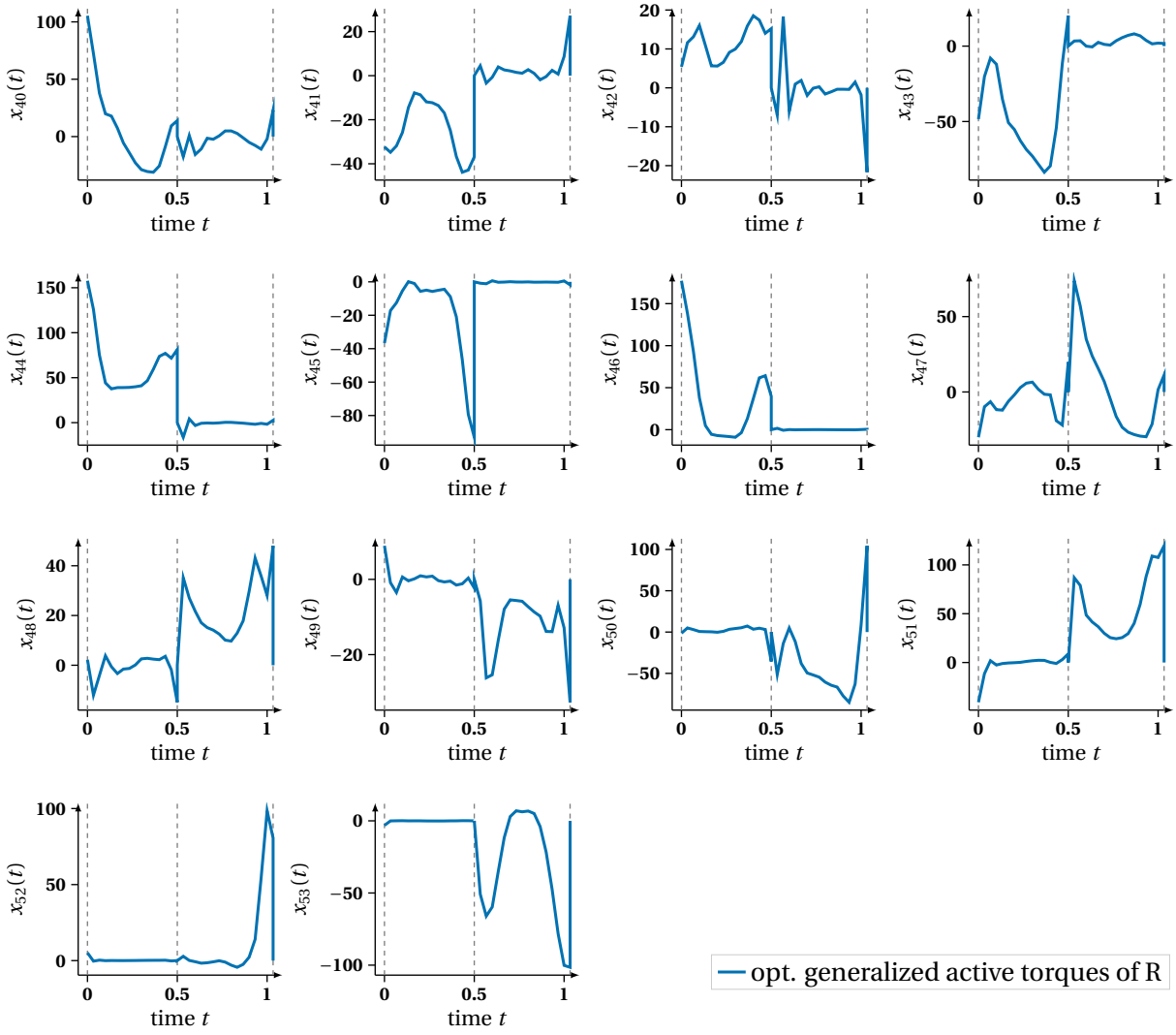


**Figure B.3:** This figure depicts the 14 optimal generalized active torques $x_{40}(t) - x_{53}(t)$ in the solution of the least-squares OCP (6.18) denoted by R.

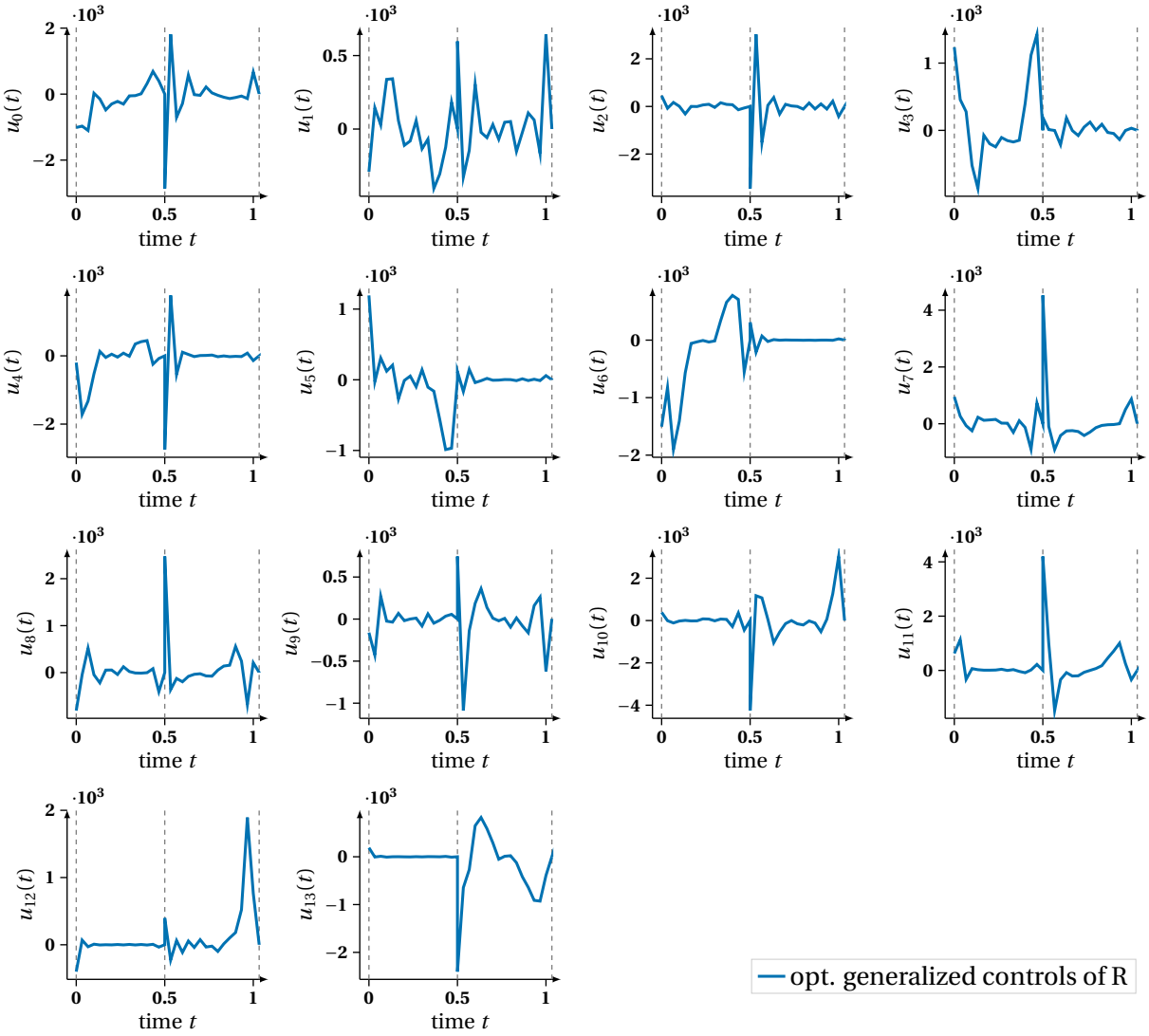


**Figure B.4:** This figure depicts the 14 optimal controls $u_0(t) - u_{13}(t)$ in the solution of the least-squares OCP (6.18) denoted by R.

## B.2 Optimal Differential States and Controls of CP Gait Synthesis

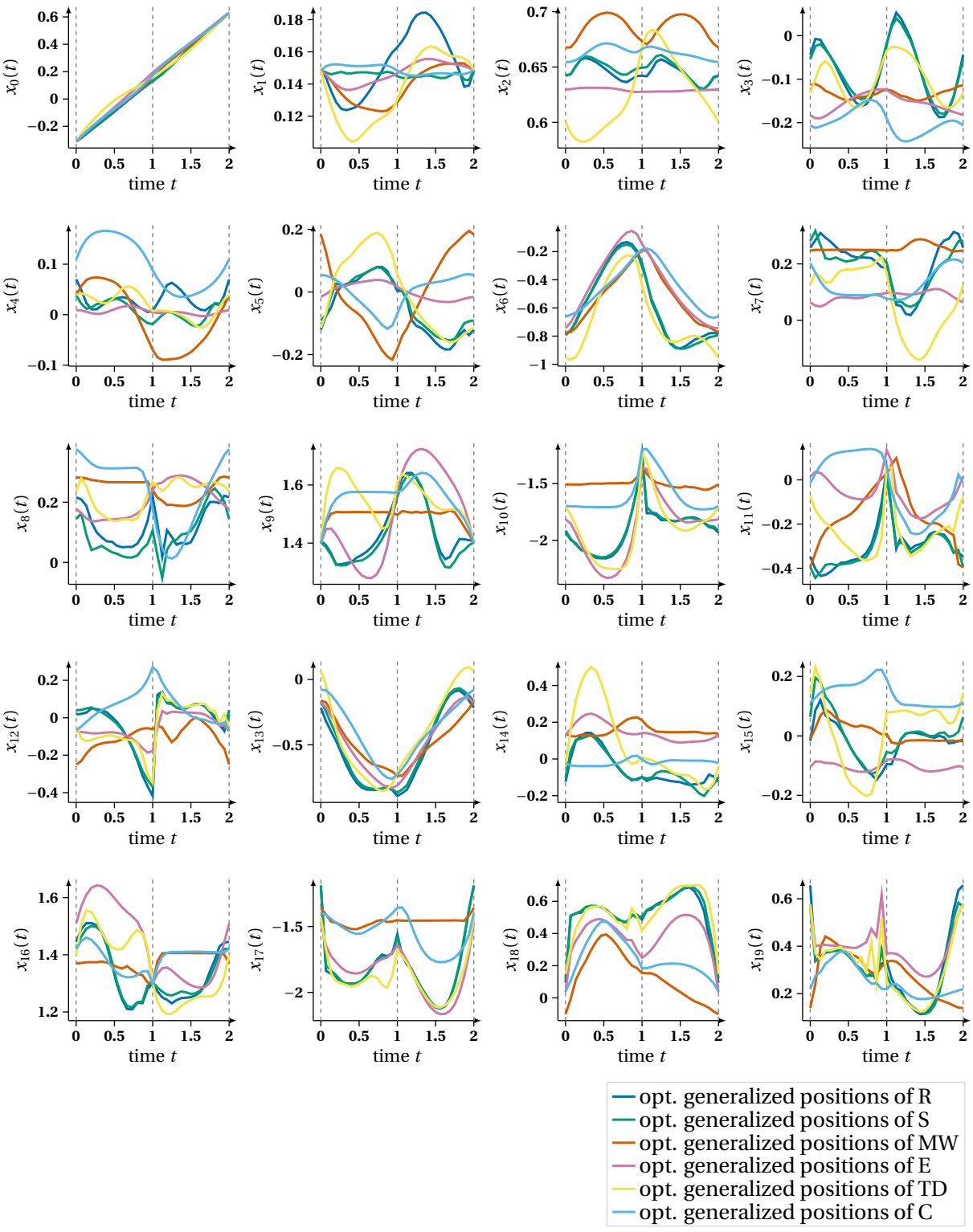


**Figure B.5:** This figure depicts the 20 optimal generalized positions $x_0(t) - x_{19}(t)$ of five solutions of OCP (6.35) on normalized stage durations with differenty weighted objective denoted by S, MW, E, TD and C, as defined in Table 10.6. This results are compared to the corresponding optimal states of the dynamics reconstruction (R) (for a discussion, see section 10.3).

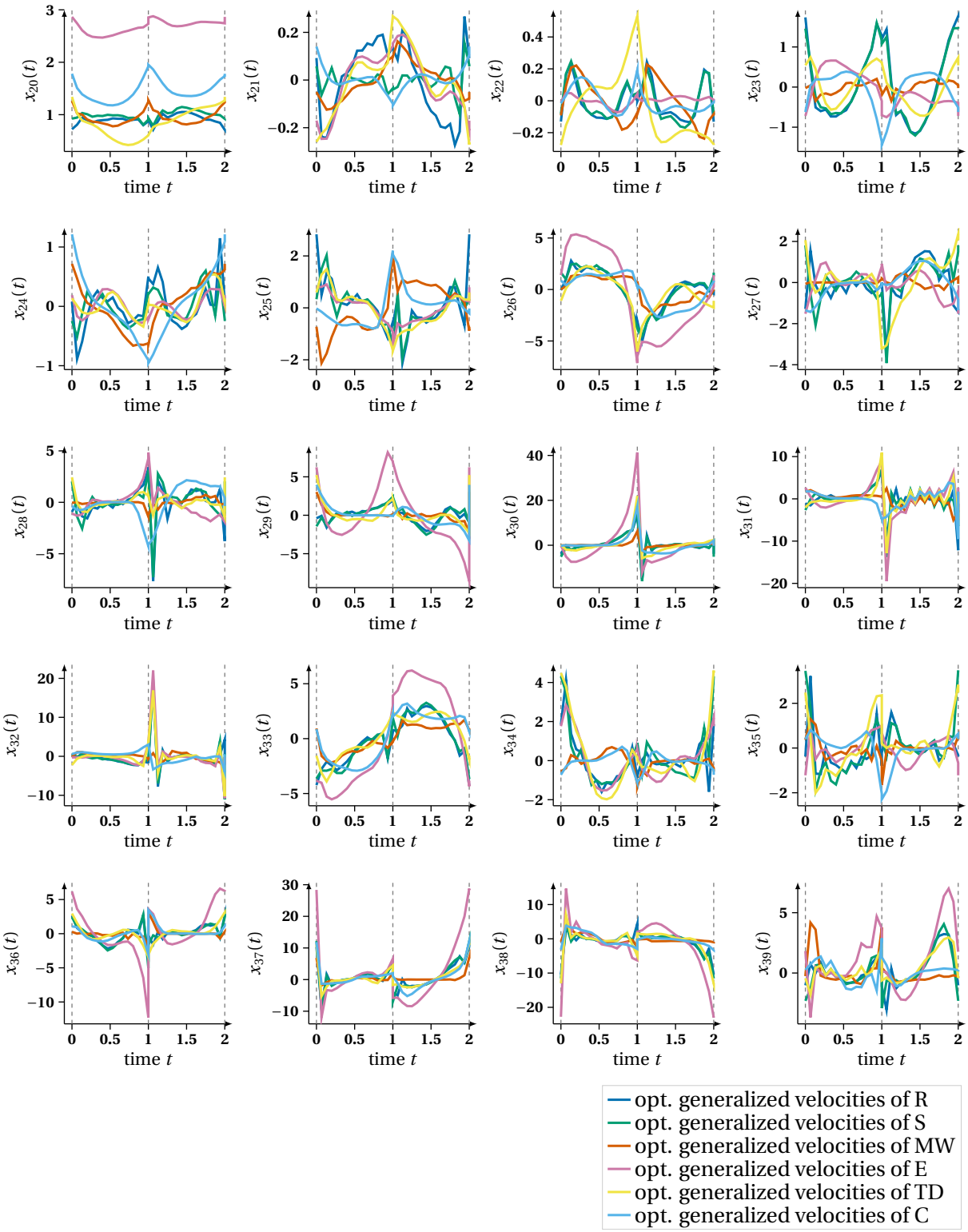


**Figure B.6:** This figure depicts the 20 optimal generalized velocities $x_{20}(t) - x_{39}(t)$ of five solutions of OCP (6.35) on normalized stage durations with differenty weighted objective denoted by S, MW, E, TD and C, as defined in Table 10.6. This results are compared to the corresponding optimal states of the dynamics reconstruction (R).

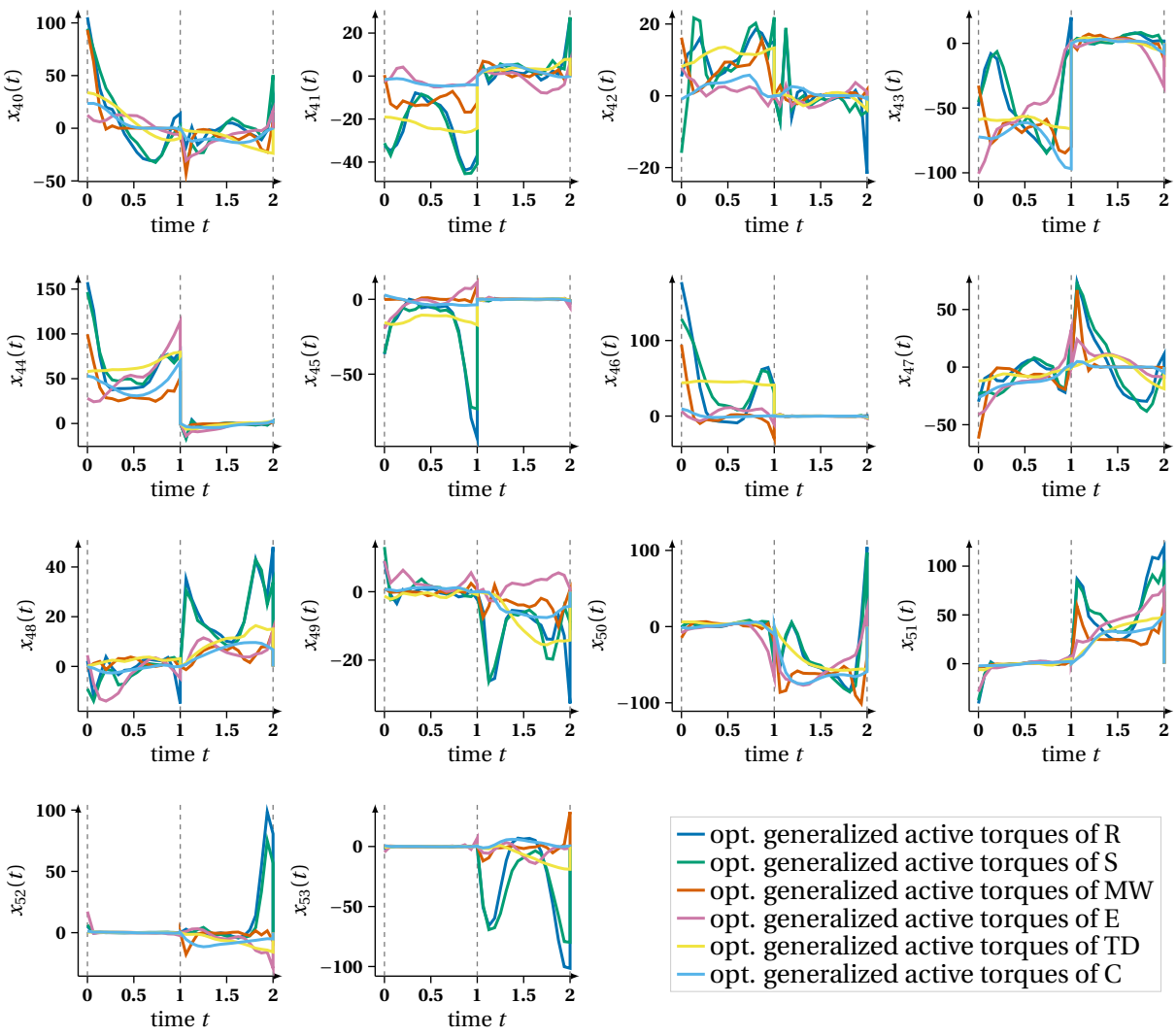


**Figure B.7:** This figure depicts the 14 optimal generalized active torques $x_{40}(t) - x_{53}(t)$ of five solutions of OCP (6.35) on normalized stage durations with differenty weighted objective denoted by S, MW, E, TD and C, as defined in Table 10.6. This results are compared to the corresponding optimal states of the dynamics reconstruction (R).

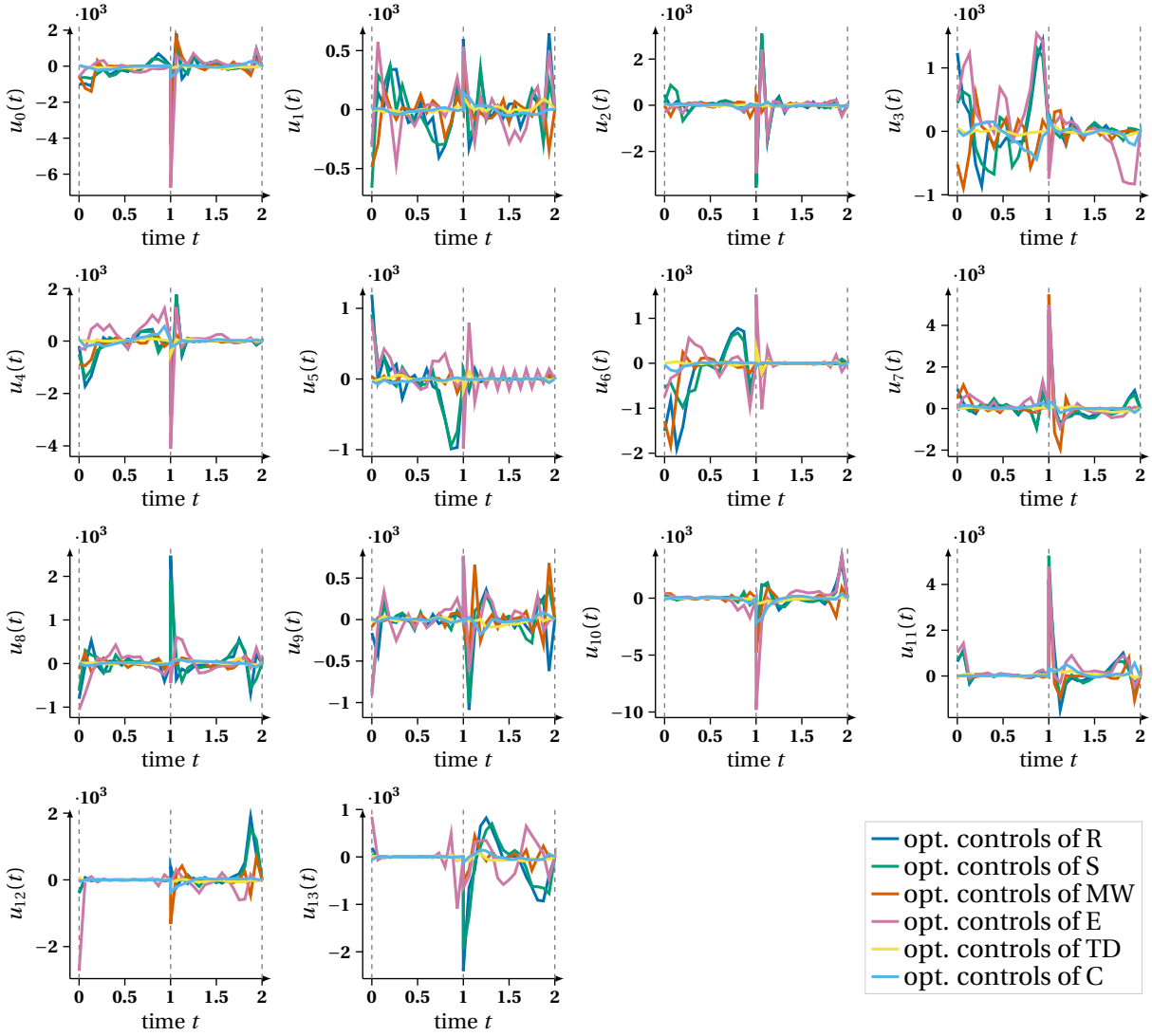


**Figure B.8:** This figure depicts the 14 optimal controls $u_0(t) - u_{13}(t)$ of five solutions of OCP (6.35) on normalized stage durations with differenty weighted objective denoted by S, MW, E, TD and C, as defined in Table 10.6. This results are compared to the optimal controls of the dynamics reconstruction (R).

# Bibliography


[1] Inria. humans toolbox. 2005. URL http://www.inrialpes.fr/bipop/software/humans/.

[2] Vicon motion systems. 2013. URL http://www.vicon.com.

[3] Btk. biomechanical toolkit. 2014. URL https://code.google.com/archive/p/b-tk/.

[4] B. M. Afkham, J. Chung, and M. Chung. Learning regularization parameters of inverse problems via deep neural networks. *Inverse Problems*, 37(10):105017, 2021. DOI: 10.1088/1361-6420/ac245d.

[5] J. Albersmeyer. *Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems*. PhD thesis, University Heidelberg, 2010. URL http://www.ub.uni-heidelberg.de/archiv/11651/.

[6] S. Albrecht. *Modeling and Numerical Solution of Inverse Optimal Control Problems for the Analysis of Human Motions*. Dissertation, TU Munich, 2013.

[7] S. Albrecht and M. Ulbrich. Mathematical programs with complementarity constraints in the context of inverse optimal control for locomotion. *Optimization Methods and Software*, 32(4):670–698, 2017. DOI: 10.1080/10556788.2016.1225212.

[8] S. Albrecht, C. Passenberg, M. Sabotka, A. Peer, M. Buss, and M. Ulbrich. *Optimization Criteria for Human Trajectory Formation in Dynamic Virtual Environments*, volume 6192, pages 257 – 262. Springer, 2010.

[9] S. Albrecht, M. Leibold, and M. Ulbrich. A bilevel optimization approach to obtain optimal cost functions for human arm movements. *Numerical Algebra, Control and Optimization*, 2(1):105–127, 2012. DOI: 10.3934/naco.2012.2.105.

[10] R. M. Alexander. The gaits of bipedal and quadrupedal animals. *The International J. of Robotics Research*, 3(2):49–59, 1984.

[11] R. M. Alexander. Optima for animals. *Princeton Univ. Press.*, 1996.

[12] F. Aller, M. Harant, and K. Mombaur. Optimization of dynamic sit-to-stand trajectories to assess whole-body motion performance of the humanoid robot reem-c. *Frontiers in Robotics and AI*, 9:898696, 2022. DOI: 10.3389/frobt.2022.898696.

[13] D. Anderson, M. Madigan, and M. Nussbaum. Maximum voluntary joint torque as a function of joint angle and angular velocity: Model development and application to the lower limb. *Journal of biomechanics*, 40:3105–13, 2007. DOI: 10.1016/j.jbiomech.2007.03.022.

[14] R. Andreani, L. D. Secchin, and P. J. S. Silva. Convergence properties of a second order augmented lagrangian method for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 28(3):2574–2600, 2018. DOI: 10.1137/17m1125698.

[15] M. Anitescu. Global convergence of an elastic mode approach for a class of mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 16(1):120–145, 2005. DOI: 10.1137/040606855.

[16] M. Anitescu. On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 15(4):1203–1236, 2005. DOI: 10.1137/s1052623402401221.

[17] M. Anitescu, P. Tseng, and S. J. Wright. Elastic-mode algorithms for mathematical programs with equilibrium constraints: global convergence and stationarity properties. *Mathematical Programming*, 110(2): 337–371, 2006. DOI: 10.1007/s10107-006-0005-4.

[18] S. Armand, G. Decoulon, and A. Bonnefoy-Mazure. Gait analysis in children with cerebral palsy. *EFORT Open Reviews*, 1:448–460, 2016. DOI: 10.1302/2058-5241.1.000052.

[19] J. F. Bard. *Practical Bilevel Optimization: Algorithms And Applications*. Springer New York, NY, 2000. ISBN 978-1-4419-4807-6. DOI: 10.1007/978-1-4757-2836-1.

[20] F. Benita and P. Mehlitz. Bilevel optimal control with final-state-dependent finite-dimensional lower level. 26:718–752, 2016.

[21] M. Benko and H. Gfrerer. An SQP method for mathematical programs with complementarity constraints with strong convergence properties. *Kybernetika*, pages 169–208, 2016. DOI: 10.14736/kyb-2016-2-0169.

[22] H. Y. Benson, A. Sen, D. F. Shanno, and R. J. Vanderbei. Interior-point algorithms, penalty methods and equilibrium problems. *Computational Optimization and Applications*, 34(2):155–182, 2006. DOI: 10.1007/s10589-005-3908-8.

[23] H. G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K. H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, pages 102–125. Springer Berlin Heidelberg, 1981. ISBN 978-3-642-68220-9.

[24] H. G. Bock. *Recent Advances in Parameteridentification Techniques for O.D.E.*, pages 95–121. Birkhäuser Boston, Boston, MA, 1983. ISBN 978-1-4684-7324-7. DOI: 10.1007/978-1-4684-7324-7_7.

[25] H. G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*. PhD thesis, University Bonn, 1987.

[26] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984. DOI: 10.1016/s1474-6670(17)61205-9.

[27] H. G. Bock, E. Kostina, M. Sauter, J. P. Schlöder, and M. Schlöder. Numerical methods for diagnosis and therapy design of cerebral palsy by bilevel optimal control of constrained biomechanical multi-body systems. In M. Hintermüller, R. Herzog, C. Kanzow, M. Ulbrich, and S. Ulbrich, editors, *Non-Smooth and Complementarity-Based Distributed Parameter Systems: Simulation and Hierarchical Optimization*, pages 21–41. Springer International Publishing, Cham, 2022. ISBN 978-3-030-79393-7. DOI: 10.1007/978-3-030-79393-7_2.

[28] P. Brown and J. McPhee. A 3d ellipsoidal volumetric foot–ground contact model for forward dynamics. *Multibody System Dynamics*, 42, 2018. DOI: 10.1007/s11044-017-9605-4.

[29] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. ISSN 0272-4960. DOI: 10.1093/imamat/6.1.76.

[30] R. H. Byrd, N. I. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Mathematical Programming*, 100(1):27–48, 2003. ISSN 1436-4646. DOI: 10.1007/s10107-003-0485-4.

[31] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. On the convergence of successive linear-quadratic programming algorithms. *SIAM Journal on Optimization*, 16(2):471–489, 2005. DOI: 10.1137/S1052623403426532.

[32] H. Chen, H. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark cstr. *Proceedings of the 3rd European Control Conference, Rome-Italy.*, pages 3247–3252, 1995.

[33] Y. Chen and M. Florian. The nonlinear bilevel programming problem: Formulations, regularity and optimality conditions. *Optimization*, 32:193–209, 1995.

[34] Y. Chen and Z. Wan. A new smoothing method for mathematical programs with complementarity constraints based on logarithm-exponential function. *Mathematical Problems in Engineering*, 2018:1–11, 2018. DOI: 10.1155/2018/5056148.

[35] F. Chollet et al. Keras. `https://keras.io`, 2015.

[36] C. Clason, Y. Deng, P. Mehlitz, and U. Prüfert. optimal control problems with control complementarity constraints. *Preprint SPP1962-081*, 2018.

[37] D. Clever and K. D. Mombaur. An inverse optimal control approach for the transfer of human walking motions in constrained environment to humanoid robots. In *Robotics: Science and Systems*, 2016.

[38] D. Clever, R. M. Schemschat, M. L. Felis, and K. Mombaur. Inverse optimal control based identification of optimality criteria in whole-body human walking on level ground. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1192–1199, 2016. DOI: 10.1109/BIOROB.2016.7523793.

[39] J. J. Craig. *Introduction to robotics*. Pearson Education, Harlow, 3. new internat. edition, 2014. ISBN 1-292-04004-1 and 978-1-292-04004-2.

[40] P. de Leva. Adjustments to zatsiorsky-seluyanov's segment inertia parameters. *Journal of Biomechanics*, 29(9):1223–1230, 1996. ISSN 0021-9290. DOI: https://doi.org/10.1016/0021-9290(95)00178-6.

[41] S. Delp, F. Anderson, A. Arnold, P. Loan, A. Habib, C. John, E. Guendelman, and D. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering, IEEE Transactions on*, 54:1940 – 1950, 2007. DOI: 10.1109/TBME.2007.901024.

[42] V. DeMiguel, M. P. Friedlander, F. J. Nogales, and S. Scholtes. A two-sided relaxation scheme for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization*, 16(2):587–609, 2005. DOI: 10.1137/04060754x.

[43] S. Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, 2002.

[44] S. Dempe and J. Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Math. Program.*, 131:37–48, 2012. DOI: 10.1007/s10107-010-0342-1.

[45] S. Dempe and A. Zemkoho. The bilevel programming problem: reformulations, constraint qualifications and optimality conditions. *Math. Program.*, 138:447, 2013. DOI: 10.1007/s10107-011-0508-5.

[46] S. Dempe, V. Kalashnikov, G. A. Pérez-Valdés, and N. Kalashnykova. *Theory, Algorithms and Applications to Energy Networks*. Springer-Verlag Berlin Heidelberg, 2015.

[47] F. Dobson, M. Morris, R. Baker, and K. Graham. Gait classification in children with cerebral palsy: A systematic review. *Gait & posture*, 25:140–52, 2007. DOI: 10.1016/j.gaitpost.2006.01.003.

[48] L. Döderlein. *Die infantilen Zerebralparesen: Diagnostik, konservative und operative Therapie.* Springer, 2nd edition, 2015. DOI: 10.1007/978-3-642-35319-2.

[49] A. L. Emonds. *Towards a simulator tool for predicting sprinting and long jump motions with and without running-specific prostheses.* PhD thesis, University Heidelberg, 2023. URL http://nbn-resolving.de/urn:nbn:de:bsz:16-heidok-328478.

[50] F. Facchinei, H. Jiang, and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 85:107–134, 1999. DOI: 10.1007/s10107990015a.

[51] R. Featherstone. *Rigid Body Dynamics Algorithms.* Springer New York, NY, 2008. ISBN 978-0-387-74314-1. DOI: 10.1007/978-1-4899-7560-7.

[52] M. L. Felis. *Modeling emotional aspects in human Locomotion.* PhD thesis, University Heidelberg, 2015. URL http://www.ub.uni-heidelberg.de/archiv/19319.

[53] M. L. Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41:495–511, 2017. DOI: 10.1007/s10514-016-9574-0.

[54] M. L. Felis and K. Mombaur. Synthesis of full-body 3-d human gait using optimal control methods. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, page 1560–1566. IEEE Press, 2016. DOI: 10.1109/ICRA.2016.7487294.

[55] M. Flegel. *Constraint Qualifications and Stationarity Concepts for Mathematical Programs with Equilibrium Constraints.* PhD thesis, University Würzburg, 2005. URL url:http://nbn-resolving.org/urn:nbn:de:bvb:20-opus-12453.

[56] M. L. Flegel and C. Kanzow. A fritz john approach to first order optimality conditions for mathematical programs with equilibrium constraints. *Optimization*, 52.3:277–286, 2003. DOI: 10.1080/0233193031000120020.

[57] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970. ISSN 0010-4620. DOI: 10.1093/comjnl/13.3.317.

[58] R. Fletcher. *Practical methods of optimization.* A Wiley-Interscience Publication. Wiley, Chichester [u.a.], 2. edition, 2001. ISBN 0-471-49463-1 and 0-471-91547-5 and 978-0-471-91547-8 and 978-0-471-49463-8.

[59] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002. DOI: 10.1007/s101070100244.

[60] R. Fletcher and S. Leyffer. Solving mathematical programs with complementarity constraints as nonlinear programs. *Optimization Methods and Software*, 19(1):15–40, 2004. ISSN 1055-6788 and 1029-4937.

[61] R. Fletcher, S. Leyffer, D. Ralph, and S. Scholtes. Local convergence of sqp methods for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization*, 17(1):259–286, 2006. DOI: 10.1137/S1052623402407382.

[62] Freepik. 2023. URL https://www.freepik.com/.

[63] M. Fukushima and P. Tseng. An implementable active-set algorithm for computing a b-stationary point of a mathematical program with linear complementarity constraints. *SIAM Journal on Optimization*, 12(3):724–739, 2002. DOI: 10.1137/s1052623499363232.

[64] M. Fukushima, Z.-Q. Luo, and J.-S. Pang. A globally convergent sequential quadratic programming algorithm for mathematical programs with linear complementarity constraints. *Computational Optimization and Applications*, 10.1:5–34, 1998. ISSN 1573-2894. DOI: 10.1023/A:1018359900133.

[65] J. Gage and B. Russman. *Gait Analysis in Cerebral Palsy*. Clinics in Developmental Medicine (Mac Keith Press). Cambridge University Press, 1991. ISBN 9780521412773.

[66] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman. The simplest walking model: Stability, complexity, and scaling. *Journal of biomechanical engineering*, 120:281–8, 1998.

[67] C. Geiger and C. Kanzow. *Theorie und Numerik Restringierter Optimierungsaufgaben*. 2002. ISBN 978-3-540-42790-2. DOI: 10.1007/978-3-642-56004-0.

[68] H. Gfrerer. Optimality conditions for disjunctive programs based on generalized differentiation with application to mathematical programs with equilibrium constraints. *SIAM Journal on Optimization*, 24.2:898–931, 2014. DOI: 10.1137/130914449.

[69] H. Gfrerer and J. J. Ye. New constraint qualifications for mathematical programs with equilibrium constraints via variational analysis. *SIAM Journal on Optimization*, 27(2):842–865, 2017. DOI: 10.1137/16M1088752.

[70] G. Giallombardo and D. Ralph. Multiplier convergence in trust-region methods with application to convergence of decomposition methods for MPECs. *Mathematical Programming*, 112(2):335–369, 2006. DOI: 10.1007/s10107-006-0020-5.

[71] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. DOI: 10.1137/S0036144504446096.

[72] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Emerald, Bingley, 2008. ISBN 0-12-283952-8 and 978-0-12-283952-8.

[73] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970. DOI: 10.1090/S0025-5718-1970-0258249-6.

[74] A. Griewank and A. Walther. *Evaluating derivatives*. SIAM, Philadelphia, 2. edition, 2008. ISBN 978-0-898716-59-7.

[75] S. Gulati and V. Sondhi. Cerebral palsy: An overview. *The Indian Journal of Pediatrics*, 85:1–11, 2017. DOI: 10.1007/s12098-017-2475-1.

[76] L. Guo and J. J. Ye. Necessary optimality conditions for optimal control problems with equilibrium constraints. *SIAM Journal on Control and Optimization*, 54(5):2710–2733, 2016. DOI: 10.1137/15M1013493.

[77] E. Haber and L. Tenorio. Learning regularization functionals - a supervised training approach. *Inverse Problems*, 19:611, 2003. DOI: 10.1088/0266-5611/19/3/309.

[78] J. P. Halloran, M. Ackermann, A. Erdemir, and A. J. van den Bogert. Concurrent musculoskeletal dynamics and finite element analysis predicts altered gait patterns to reduce foot tissue loading. *Journal of Biomechanics*, 43(14):2810–2815, 2010. ISSN 0021-9290. DOI: 10.1016/j.jbiomech.2010.05.036. URL https://www.sciencedirect.com/science/article/pii/S0021929010003234.

[79] M. Harant, M. Näf, and K. Mombaur. Multibody dynamics and optimal control for optimizing spinal exoskeleton design and support. *Multibody System Dynamics*, 57:389–411, 2023. DOI: 10.1007/s11044-023-09877-w.

[80] K. Hatz. *Efficient numerical methods for hierarchical dynamic optimization with application to cerebral palsy gait modeling*. PhD thesis, University Heidelberg, 2014. URL http://www.ub.uni-heidelberg.de/archiv/16803.

[81] K. Hatz, J. P. Schlöder, and H. G. Bock. Estimating parameters in optimal control problems. *SIAM Journal on Scientific Computing*, 34(3):A1707–A1728, 2012. DOI: 10.1137/110823390.

[82] R. Herzog, C. Meyer, and G. Wachsmuth. B- and strong stationarity for optimal control of static plasticity with hardening. *SIAM Journal on Optimization*, 23(1):321 – 352, 2013.

[83] M. R. Hestenes. *Calculus of Variations and Optimal Control Theory*. John Wiley & Sons, New York, 1966.

[84] M. Hintermüller and J. Kopacka. Mathematical programs with complementarity constraints in function space: C- and strong stationarity and a path-following algorithm. *SIAM Journal on Optimization*, 20(2): 868 – 902, 2009.

[85] M. Hintermüller and T. Surowiec. First order optimality conditions for elliptic mathematical programs with equilibrium constraints via variational analysis. *SIAM Journal on Optimization*, 21(4):1561–1593, 2011.

[86] M. Hintermüller, B. Mordukhovich, and T. Surowiec. Several approaches for the derivation of stationarity conditions for elliptic mpecs with upper-level control constraints. *Mathematical Programming*, 146(1-2): 555 – 582, 2014.

[87] K.-L. Ho Hoang, S. Wolf, and K. Mombaur. Benchmarking stability of bipedal locomotion based on individual full body dynamics and foot placement strategies - application to impaired and unimpaired walking. *Frontiers in Robotics and AI*, 5, 2018. DOI: 10.3389/frobt.2018.00117.

[88] T. Hoheisel, C. Kanzow, and A. Schwartz. Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. *Mathematical Programming Series A*, 137.1–2:257–288, 2013. DOI: 10.1007/s10107-011-0488-5.

[89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

[90] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. ISSN 0893-6080. DOI: 10.1016/0893-6080(90)90005-6.

[91] X. Hu and D. Ralph. Convergence of a penalty method for mathematical programming with complementarity constraints. *Journal of Optimization Theory and Applications*, 123.2:365–290, 2004. DOI: 10.1007/s10957-004-5154-0.

[92] Y. Hu and K. Mombaur. Analysis of human leg joints compliance in different walking scenarios with an optimal control approach**the research leading to these results has received funding from the european union seventh framework programme (fp7/2007 - 2013) under grant agreement n. 611909 (koroibot), www.koroibot.eu. *IFAC-PapersOnLine*, 49(14):99 – 106, 2016. DOI: 10.1016/j.ifacol.2016.07.992.

[93] A. F. Izmailov, M. V. Solodov, and E. I. Uskov. Global convergence of augmented lagrangian methods applied to optimization problems with degenerate constraints, including problems with complementarity constraints. *SIAM Journal on Optimization*, 22(4):1579–1606, 2012. DOI: 10.1137/120868359.

[94] A. Jain. *Robot and Multibody Dynamics: Analysis and Algorithms*. 2012. ISBN 978-1-4419-7266-8. DOI: 10.1007/978-1-4419-7267-5.

[95] H. Jiang and D. Ralph. Smooth SQP methods for mathematical programs with nonlinear complementarity constraints. *SIAM Journal on Optimization*, 10(3):779–808, 2000. DOI: 10.1137/s1052623497332329.

[96] A. Kadrani, J.-P. Dussault, and A. Benchakroun. A new regularization scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 20(1):78–103, 2009. DOI: 10.1137/070705490.

[97] C. Kanzow and A. Schwartz. Mathematical programs with equilibrium constraints: Enhanced fritz john-conditions. *SIAM Journal on Optimization*, 20:2730–2753, 2010.

[98] C. Kanzow and A. Schwartz. A new regularization method for mathematical programs with complementarity constraints with strong convergence properties. *SIAM Journal on Optimization*, 23(2):770–798, 2013. DOI: 10.1137/100802487.

[99] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master thesis, Department of Mathematics, University of Chicago, 1939.

[100] W. Khalil. *Modeling, identification & control of robots*. Kogan Page Science paper edition. Kogan Page Science, London, 2004. ISBN 1-281-98541-4.

[101] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[102] C. Kirches, S. Sager, H. G. Bock, and J. P. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 31(2):137–153, 2010. DOI: 10.1002/oca.892.

[103] C. Kirches, J. Larson, S. Leyffer, and P. Manns. Sequential linearization method for bound-constrained mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 32(1):75–99, 2022. DOI: 10.1137/20M1370501.

[104] M. Knauer. *Bilevel-Optimalsteuerung mittels hybrider Lösungsmethoden am Beispiel eines deckengeführten Regalbediengerätes in einem Hochregallager*. Dissertation, Universität Bremen, 2009.

[105] M. Knauer and C. Büskens. Bilevel optimization of container cranes. *In Progress in Industrial Mathematics at ECMI 2008*, pages 913 – 918, 2010.

[106] K. Krigger. Cerebral palsy: An overview. *American family physician*, 73:91–100, 2006.

[107] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492, Berkeley, 1951. University of California Press.

[108] D. B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. PhD thesis, University Heidelberg, 1998.

[109] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects. *Computers & Chemical Engineering*, 27(2):157–166, 2003. ISSN 0098-1354. DOI: 10.1016/S0098-1354(02)00158-8.

[110] D. B. Leineweber, A. Schäfer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization: Part II: Software aspects and applications. *Computers & Chemical Engineering*, 27(2):167–174, 2003. ISSN 0098-1354. DOI: 10.1016/S0098-1354(02)00195-3.

[111] F. Lenders. *Numerical Methods for Mixed-Integer Optimal Control with Combinatorial Constraints*. PhD thesis, University Heidelberg, 2018.

[112] F. L. Lewis, D. Vrabie, and V. L. Syrmos. *Optimal Control*. John Wiley and Sons, Hoboken, 2012.

[113] S. Leyffer. *Complementarity constraints as nonlinear equations: Theory and numerical experience*, pages 169–208. Springer US, Boston, MA, 2006. ISBN 978-0-387-34221-4. DOI: 10.1007/0-387-34221-4_9.

[114] S. Leyffer and T. S. Munson. A globally convergent filter method for mpecs. *Preprint ANL/MCS-P1457-0907, Argonne National Laboratory, Mathematics and Computer Science Division*, 2007.

[115] S. Leyffer, G. Lopez-Calva, and J. Nocedal. Interior methods for mathematical programs with complementarity constraints. *SIAM JOURNAL ON OPTIMIZATION*, 17(1):52–77, 2006. ISSN 1052-6234 and 1095-7189.

[116] G. Lin and M. Fukushima. New relaxation method for mathematical programs with complementarity constraints. *Journal of Optimization Theory and Applications*, 118(1):81–116, 2003. DOI: 10.1023/a:1024739508603.

[117] G.-H. Lin and M. Fukushima. A modified relaxation scheme for mathematical programs with complementarity constraints. *Annals of Operations Research*, 133(1-4):63–84, 2005. DOI: 10.1007/s10479-004-5024-z.

[118] X. Liu and J. Sun. Generalized stationary points and an interior-point method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 101(1), 2004. DOI: 10.1007/s10107-004-0543-6.

[119] Z. Luo, J. Pang, and D. Ralph. *Mathematical Programs with Equlibrium Constraints*. Cambridge University Press, 1996. DOI: 10.1017/CBO9780511983658.

[120] O. Mangasarian and S. Fromovitz. The fritz john necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, 17(1):37–47, 1967. ISSN 0022-247X. DOI: 10.1016/0022-247X(67)90163-1.

[121] P. Mehlitz. Necessary optimality conditions for a special class of bilevel programming problems with unique lower level solution. *Optimization*, 66(10):1533–1562, 2017. DOI: 10.1080/02331934.2017.1349123.

[122] P. Mehlitz and G. Wachsmuth. Weak and strong stationarity in generalized bilevel programming and bilevel optimal control. *Optimization*, 65(5):907–935, 2016. DOI: 10.1080/02331934.2015.1122007.

[123] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992. DOI: 10.1137/0802028.

[124] A. Meyer. *Numerical Solution of Optimal Control Problems with Explicit and Implicit Switches*. PhD thesis, University Heidelberg, 2020. URL `http://www.ub.uni-heidelberg.de/archiv/27701`.

[125] M. Millard, T. Uchida, A. Seth, and S. Delp. Flexing computational muscle: Modeling and simulation of musculotendon dynamics. *Journal of Biomechanical Engineering*, 135:021005, 2013. DOI: 10.1115/1.4023390.

[126] M. Millard, M. Sreenivasa, and K. Mombaur. Predicting the motions and forces of wearable robotic systems using optimal control. *Frontiers in Robotics and AI*, 4:41, 2017. DOI: 10.3389/frobt.2017.00041.

[127] M. Millard, A. L. Emonds, M. Harant, and K. Mombaur. A reduced muscle model and planar musculoskeletal model fit for the simulation of whole-body movements. *Journal of Biomechanics*, 89:11–20, 2019. ISSN 0021-9290. DOI: 10.1016/j.jbiomech.2019.04.004. URL `https://www.sciencedirect.com/science/article/pii/S0021929019302568`.

[128] K. Mombaur. *Stability optimization of open-loop controlled walking robots*. PhD thesis, University Heidelberg, 2001. URL `http://www.ub.uni-heidelberg.de/archiv/1796`.

[129] K. Mombaur and D. Clever. *Inverse Optimal Control as a Tool to Understand Human Movement*, pages 163–186. Springer International Publishing, Cham, 2017. ISBN 978-3-319-51547-2. DOI: 10.1007/978-3-319-51547-2_8.

[130] K. Mombaur, A. Truong, and J.-P. Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous Robots*, 28(3):369–383, 2010. ISSN 1573-7527. DOI: 10.1007/s10514-009-9170-7.

[131] K. Mombaur, H. Vallery, Y. Hu, J. Buchli, P. Bhounsule, T. Boaventura, P. M. Wensing, S. Revzen, A. D. Ames, I. Poulakakis, and A. Ijspeert. Chapter 4 - control of motion and compliance. In M. A. Sharbafi and A. Seyfarth, editors, *Bioinspired Legged Locomotion*, pages 135–346. Butterworth-Heinemann, 2017. ISBN 978-0-12-803766-9. DOI: 10.1016/B978-0-12-803766-9.00006-3.

[132] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer series in operation research and financial engineering. Springer, New York, NY, second edition, 2006. ISBN 978-1-4939-3711-0 and 0-387-30303-0 and 978-0-387-30303-1.

[133] J. V. Outrata. Optimality conditions for a class of mathematical programs with equilibrium constraints. *Mathematics of Operations Research*, 24(3):627–644, 1999. DOI: 10.1287/moor.24.3.627.

[134] J. V. Outrata. A generalized mathematical program with equilibrium constraints. *SIAM Journal on Control and Optimization*, 38(5):1623–1638, 2000. DOI: 10.1137/S0363012999352911.

[135] R. Palisano, P. Rosenbaum, W. SD, D. Russell, W. EP, and B. Galuppi. Development and reliability of a system to classify gross motor function in children with cerebral palsy. *Developmental medicine and child neurology*, 39:214–23, 1997. DOI: 10.1111/dmcn.1997.39.issue-4.

[136] J.-S. Pang and M. Fukushima. Complementarity constraint qualifications and simplified b-stationarity conditions for mathematical programs with equilibrium constraints. *Computational Optimization and Applications*, 13.1:111–136, 1999. ISSN 1573-2894. DOI: 10.1023/A:1008656806889.

[137] C. Panteliadis and H. Strassburg. *Cerebral Palsy: Principles and Management*. Thieme Publishers Series. Georg Thieme, 2004. ISBN 9783131400215.

[138] K. J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, University of Bonn, 1981.

[139] L. Pontryagin, V. Boltyanski, R. Gamkrelidze, and E. Mischenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962.

[140] A. Potschka. Handling path constraints in a direct multiple shooting method for optimal control problems. Diploma thesis, University Heidelberg, 2006.

[141] A. Potschka, H. G. Bock, and J. P. Schlöder. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software*, 24(2):237–252, 2009. DOI: 10.1080/10556780902753098.

[142] A. U. Raghunathan and L. T. Biegler. Mathematical programs with equilibrium constraints (MPECs) in process engineering. *Computers & Chemical Engineering*, 27(10):1381–1392, 2003. DOI: 10.1016/s0098-1354(03)00092-9.

[143] D. Ralph and S. J. Wright. Some properties of regularization and penalization schemes for mpecs. *Optimization Methods and Software*, 19(5):527–556, 2004. DOI: 10.1080/10556780410001709439.

[144] L. Ren, D. Howard, L. Ren, C. Nester, and L. Tian. A generic analytical foot rollover model for predicting translational ankle kinematics in gait simulation studies. *Journal of Biomechanics*, 43(2):194–202, 2010. ISSN 0021-9290. DOI: 10.1016/j.jbiomech.2009.09.027.

[145] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22 (3):400–407, 1951.

[146] J. Rodda and K. Graham. Classification of gait patterns in spastic hemiplegia and spastic diplegia: A basis for a management algorithm. *European journal of neurology : the official journal of the European Federation of Neurological Societies*, 8 Suppl 5:98–108, 2001. DOI: 10.1046/j.1468-1331.2001.00042.x.

[147] P. Rosenbaum. Erratum: A report: The definition and classification of cerebral palsy. *Developmental Medicine and Child Neurology*, 49:8–14, 2007. DOI: 10.1111/j.1469-8749.2007.00480.x.

[148] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[149] H. Scheel and S. Scholtes. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Math. Oper. Res.*, 25:1–22, 2000. DOI: 10.1287/moor.25.1.1.15213.

[150] J. P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*. PhD thesis, University Bonn, 1988.

[151] M. Schlöder. *Numerical methods for optimal control of constrained biomechanical multi-body systems appearing in therapy design of cerebral palsy*. PhD thesis, University Heidelberg, 2022. URL `http://nbn-resolving.de/urn:nbn:de:bsz:16-heidok-313074`.

[152] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61, 2014. DOI: 10.1016/j.neunet.2014.09.003.

[153] S. Scholtes. Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 11:918–936, 2001. DOI: 10.1137/S1052623499361233.

[154] S. Scholtes. Combinatorial structures in nonlinear programming. *Tech. rep. University of Cambridge*, 2002. URL `http://www.optimization-online.org/DB%5CHTML/2002/05/477.html`.

[155] S. Scholtes. Nonconvex structures in nonlinear programming. *Operations Research*, 52.3:368–383, 2004. DOI: 10.1287/opre.1030.0102.

[156] S. Scholtes and M. Stöhr. Exact penalization of mathematical programs with equilibrium constraints. *SIAM Journal on Control and Optimization*, 37(2):617–652, 1999. DOI: 10.1137/s0363012996306121.

[157] A. A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, 4. edition, 2013. DOI: 10.1017/CBO9781107337213.

[158] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970. DOI: 10.2307/2004840.

[159] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming. Modeling and theory*. 2009. DOI: 10.1137/1.9780898718751.

[160] K. Shimizu, Y. Ishizuka, and J. F. Bard. *Nondifferentiable and two-level mathematical programming*. Kluwer Academic, Dordrecht, 1997.

[161] A. Sommer. *Numerical methods for parameter estimation in dynamical systems with noise*. PhD thesis, University Heidelberg, 2017. URL `http://nbn-resolving.de/urn:nbn:de:bsz:16-heidok-225894`.

[162] S. Steffensen and M. Ulbrich. A new relaxation scheme for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization*, 20(5):2504–2539, 2010. DOI: 10.1137/090748883.

[163] O. Stein. Lifting mathematical programs with complementarity constraints. *Mathematical Programming*, 131(1-2):71–94, 2010. DOI: 10.1007/s10107-010-0345-y.

[164] M. C. Steinbach. Numerische Berechnung optimaler Steuerungen für Industrieroboter. Diploma thesis, University Heidelberg, 1987.

[165] M. Stöhr. *Nonsmooth trust region methods and their applications to mathematical programs with equilibrium constraints*. Dissertation. University of Karlsruhe, 2000.

[166] S. Vajda and R. Isaacs. Differential games. a mathematical theory with applications to warfare and pursuit, control and optimization. *The Mathematical Gazette*, 51(375):80, 1967. DOI: 10.2307/3613661.

[167] M. van der Krogt, C. Doorenbosch, and J. Harlaar. The effect of walking speed on hamstrings length and lengthening velocity in children with spastic cerebral palsy. *Gait & posture*, 29:640–4, 2009. DOI: 10.1016/j.gaitpost.2009.01.007.

[168] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[169] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006. DOI: 10.1007/s10107-004-0559-y.

[170] A. Walther. Getting started with ADOL-C. *Combinatorial Scientific Computing*, 2009. DOI: 10.1201/b11644-8.

[171] A. Walther and A. Griewank. Getting started with adol-c. In U. Naumann and O. Schenk, editors, *Combinatorial Scientific Computing*, chapter 7, pages 181–202. Chapman-Hall CRC Computational Science, 2012.

[172] J. Winters, T F, J. R. Gage, and R. Hicks. Gait patterns in spastic hemiplegia in children and young adults. *J Bone Joint Surg Am*, 69(3):437–41, 1987.

[173] S. Wolf. Heidelberg motionlab. *Heidelberg University Hospital, Department Orthopedic Surgery*, 2023. URL `http://www.heidel-motionlab.de/`.

[174] J. Ye. Optimization conditions for bilevel programming problems. *Optimization*, 33:9 – 27, 1995.

[175] J. Ye. Constraint qualifications and necessary optimality conditions for optimization problems with variational inequality constraints. *SIAM Journal on Optimization*, 10(4):943–962, 2000. DOI: 10.1137/S105262349834847X.

[176] J. Ye. Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints. *Journal of Mathematical Analysis and Applications*, 307:350–369, 2005. DOI: 10.1016/j.jmaa.2004.10.032.

# List of Figures

# List of Tables

# List of Acronyms

**AD** Automatic Differentiation.

**BDF** Backward Differentiation Formulas.

**BFGS** Broyden–Fletcher–Goldfarb–Shanno.

**CoM** Center of Mass.

**CP** Cerebral Palsy.

**CQ** Constraint Qualification.

**DAE** Differential Algebraic Equation.

**DNN** Deep Neural Network.

**DOF** Degree of Freedom.

**END** External Numerical Differentiation.

**GMFCS** Gross Motor Function Classification System.

**IND** Internal Numerical Differentiation.

**IVP** Initial Value Problem.

**KKT** Karush-Kuhn-Tucker.

**LICQ** Linear Independence Constraint Qualification.

**MFCQ** Mangasarian-Fromowitz Constraint Qualification.

**MPCC** Mathematical Program with Complementarity Constraints.

**NLP** Nonlinear Programming Problem.

**OC** Optimal Control.

**OCP** Optimal Control Problem.

**ODE** Ordinary Differential Equation.

**PE** Parameter Estimation.

**QP** Quadratic Program.

**RNLP** Relaxed Nonlinear Programming Problem.

**SQP** Sequential Quadratic Programming.

**VDE** Variational Differential Equation.