Universität Ruperto Carola

zu Heidelberg

Fakultät für Mathematik und Informatik

A Numerical Method for Nonlinear Robust Optimal Control with Implicit Discontinuities and an Application to Powertrain Oscillations

Diplomarbeit

vorgelegt von

cand. math. Christian Kirches

aus

Kandel in Rheinland-Pfalz

betreut durch

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

Oktober 2006

A Numerical Method for Nonlinear Robust Optimal Control with Implicit Discontinuities and an Application to Powertrain Oscillations

Christian Kirches

Interdisciplinary Center for Scientific Computing (IWR) · University of Heidelberg
Im Neuenheimer Feld 368 · D-69120 Heidelberg · christian.kirches@iwr.uni-heidelberg.de

12th October, 2006

Abstract

This diploma thesis describes an approximative algorithm for robust nonlinear optimal control of problems described in terms of implicitly discontinuous ODE models. The algorithm has been embedded into the state-of-the-art optimal control software package *MUSCOD-II* developed in the work group of *Prof. Dr. Dr. h.c. Hans Georg Bock* at the *Ruperto Carola University of Heidelberg*.

Nonlinear equality- and inequality-constrained optimal control problems with uncertain parameters are addressed by an approximate robust formulation based on linearization of the uncertainty set. Sparsity-preserving formulations of the underlying nonlinear problem are presented.

The algorithm is complemented by a new integrator allowing for the treatment of implicitly defined discontinuities of non-stiff ODE models. An efficient method for the generation of second-order sensitivities is described. Proper discontinuity updates of first- and second-order sensitivities are derived and implemented.

The presented algorithms are applied to the problem of powertrain oscillations. Fast changes of the engine torque typically occurring during the acceleration of a car induce very specific oscillations of the car's powertrain. As these oscillations degrade the driving comfort, an effort is made to come up with engine control schemes that realize oscillation-free acceleration while maintaining high responsiveness.

An ODE model of the powertrain found in a *Mercedes C-Class* has been developed and improved in cooperation with *DaimlerChrysler AG* in *Stuttgart-Untertürkheim, Germany*. A software tool simplifying the process of parameter estimation has been developed in order to identify unknown parameters from real-world measurements taken on a test track.

Off-line optimal control scenarios are employed to evaluate to what extent the observed powertrain oscillations could ideally be diminished. The obtained engine control schemes allow for virtually oscillation-free acceleration of the car.

Focussing on a mechanical end-stop in the model, estimated uncertainties of the model parameters are taken into account. Powertrain acceleration schemes that gently touch the end-stop when switching from coasting mode to traction mode are obtained. These schemes show robustness against deviations in parameters, powertrain states, and applied engine controls.

Mathematics Subject Classification (2000): 62F, 62L, 62P, 65D, 65K, 65L, 90C, 93B.

Eine numerische Methode zur robusten nichtlinearen Optimalsteuerung bei impliziten Unstetigkeiten mit Anwendung auf Lastwechselschwingungen

Christian Kirches

Interdisziplinäres Zentrum für wissenschaftliches Rechnen (IWR) · Universität Heidelberg
Im Neuenheimer Feld 368 · D-69120 Heidelberg · christian.kirches@iwr.uni-heidelberg.de

12. Oktober 2006

Zusammenfassung

Diese Diplomarbeit beschreibt einen approximativen Algorithmus zur robusten nichtlinearen Optimalsteuerung von Problemen, welche durch gewöhnliche Differentialgleichungen mit impliziten Unstetigkeiten beschrieben werden können. Der Algorithmus wurde in die Optimalsteuerungssoftware *MUSCOD-II* integriert, welche in der Arbeitsgruppe von *Prof. Dr. Dr. h.c. Hans Georg Bock* an der *Ruprecht-Karls-Universität Heidelberg* entwickelt wird.

Nichtlineare gleichungs- und ungleichungsbeschränkte Optimalsteuerungsprobleme mit unsicheren Parametern werden durch eine auf Linearisierung des Unsicherheitsbereichs basierende approximative robuste Problemformulierung behandelt. Varianten dieser Formulierung werden beschrieben, welche die dünne Besetztheit von Ableitungsmatrizen des zugrunde liegenden nichtlinearen Problems beibehalten.

Der Algorithmus wird ergänzt durch einen neuen Integrator, welcher die Behandlung impliziter Unstetigkeiten in nicht-steifen gewöhnlichen Differentialgleichungen erlaubt. Eine effiziente Methode zur Erzeugung von Sensitivitäten zweiter Ordnung wird vorgestellt. Korrekte Aktualisierungen dieser Sensitivitäten bei impliziten Unstetigkeiten werden hergeleitet.

Die vorgestellten Algorithmen werden auf das Problem von Lastwechselschwingungen im KFZ-Antriebsstrang angewandt. Schnelle Änderungen des Motormoments, welche typischerweise während Beschleunigungsvorgängen auftreten, regen den Antriebsstrang zu spezifischen Schwingungen an. Da diese den Fahrkomfort beeinträchtigen, werden Anstrengungen zur Bestimmung von Motorkontrollschemata unternommen, welche schwingungsfreie Beschleunigung bei möglichst hoher Agilität gestatten.

Ein Modell des Antriebsstrangs einer *Mercedes C-Klasse* wurde in Zusammenarbeit mit der *DaimlerChrysler AG* in *Stuttgart-Untertürkheim* entwickelt und verbessert. Ein Werkzeug zur Vereinfachung der Parameteridentifizierung wurde entwickelt und ermöglicht die Bestimmung unbekannter Modellparameter und deren Unsicherheiten anhand von auf der Teststrecke aufgenommenen Messdaten.

Anhand von Optimalsteuerungs-Szenarien wird untersucht, inwieweit die beobachteten Schwingungen des Antriebsstrangs in verschiedenen Betriebssituationen gedämpft werden können. Die so erhaltenen Motorkontrollschemata gestatten eine nahezu schwingungsfreie Beschleunigung.

Während des Umlegens des Antriebsstrangs vom Schubbetrieb in den Zugbetrieb ist eine mechanische Nebenbedingung einzuhalten. Robuste Motorkontrollschemata, welche diese Bedingung erfüllen, werden berechnet. Dabei werden geschätzte Unsicherheiten in den Modellparametern, den aktuellen Systemzuständen sowie den applizierten Motorkontrollen berücksichtigt.

Acknowledgements

This diploma thesis has been prepared in the *Simulation & Optimization* work group at the *Interdisciplinary Center for Scientific Computing (IWR)* of the *University of Heidelberg*.

I wish to thank Professor Dr. Dr. h.c. Hans Georg Bock and Dr. Johannes P. Schlöder for granting me the unique opportunity to participate in the scientific work of their work group. Various courses and seminars, given by them as well as by Dr. Moritz Diehl and Dr. Ekaterina Kostina, attracted me to the highly interesting work of their group.

I am deeply indebted to Dr. Moritz Diehl, who now is a Professor at the Katholieke Universiteit Leuven. He did an extraordinary and outstanding job as my supervisor while I was working on this thesis. His ideas and suggestions strongly influenced my work. The continuous interest and enthusiasm I could always feel when discussing my work with him have been a constant source of motivation.

I thank Dr. Bernd B. Schneider of *DaimlerChrysler AG* in *Stuttgart-Untertürkheim* for many fruitful discussions, and for his hospitality during our visits to his work group in Stuttgart. His detailed feedback on my work was always very welcome and much appreciated. It was a thrilling experience to find my work being of interest to one of the world's largest automotive companies.

I am grateful to Professor Dr. Moritz Diehl, Dr. Sebastian Sager, and Dr. Bernd B. Schneider for proof-reading this thesis and for their helpful hints and suggestions for its improvement.

I wish to extend my thanks to all other members of the *Simulation & Optimization* work group. They all helped to create the friendly, cooperative, and inspiring atmosphere that at all times made it a pleasure to be a member of the team.

I would like to express my gratitude to my girlfriend Simone Evke de Groot. Thank you for your love, support and motivation, for bearing and crossing the distance, and for all the visions we share.

Finally, I wish to thank my parents, Ulrike and Claus Kirches, for their continuous backing of my interest in mathematics and computer science. It is your support that enabled me to pursue and enjoy my studies in Heidelberg. Without your love and encouragement, this work would not have been possible.

Contents

Chapter 1

Introduction

1.1 Goals and Highlights

This thesis aims at combining theory and algorithms from two areas of active research: Optimization of implicitly discontinuous systems, and optimization under uncertainty. We present theory and algorithms from both areas, applied to optimal control of ODE systems. In order to demonstrate the applicability and effectiveness of the presented numerical methods, all described algorithms have been implemented. In cooperation with *DaimlerChrysler AG* in *Stuttgart-Untertürkheim*, *Germany*, they are employed to treat a guiding example in automotive engineering.

Implicit Discontinuities and Switches

When we represent real-world problems in terms of systems of ordinary differential equations (ODEs), we must require the involved model functions to exhibit sufficient smoothness. This is to ensure that the systems can be accurately solved by appropriate numerical algorithms such as one-step or multi-step ODE solvers.
Many phenomena in mechanics, physics, and chemistry, however, feature inherent discontinuities. One may think here, e.g., of mechanical end-stops, friction phenomena, rebounds, overflowing of vessels, or changes in the phase of matter. We may also desire to exchange parts of the model in a discontinuous manner whenever the system satisfies certain *implicit* conditions. An example might be the automatic selection of gears in a vehicle's powertrain, depending on the engine's work load. This is to be clearly distinguished from *explicit* switches which are driven and decided by an external source (e.g., a human being, an off-line optimizer, etc.), and which we may regard as binary *control functions* applied to the system.
Consequentially, classical methods for the solution of ODE systems need to be extended by a facility for *event detection* that has the ability to detect, locate, and treat such implicit switch conditions.
In this thesis, we present an efficient numerical method for the exact treatment of implicit discontinuities and switches in non-stiff ODE models. As opposed to explicit switches, the generation of sensitivities of the ODE system's solution with respect to model parameters requires special attention whenever an implicit switch takes place.

Parameter Estimation

When we design a system of ordinary differential equations to model real-world problems that demand for optimization, the fundamental laws of nature dictate and allow to derive the structure and dependencies of such a system. What we usually cannot directly derive from these laws are the *values* of involved model parameters. Here, one may think of all kinds of physical quantities such as mass, velocity, torque, coefficients of friction and damping, temperature, pressure, etc. We may have at hand measurement devices for some of these quantities, allowing us to obtain *observed data*. Others, such as acceleration, torque, or damping coefficients may prove hard to observe on the real-world object of interest.

It is the domain and purpose of parameter estimation methods to obtain values for such unknown and possibly impossible-to-observe model parameters by comparing parts of the model's behavior to available observed data, and by making inferences about better parameterizations from these comparisons. Having found a parameterization that is in some sense optimal for the given model and observed data, we may in turn evolve the model to better represent reality.
In this thesis, we present the theory of estimating parameters using least-squares fits against observed data. In addition, we present possibilities to express the confidence we may have in the found parameterizations by determining estimated uncertainties for the estimated nominal parameter values.

Robust Optimal Control

Assume we have modeled a controllable process using an ODE system. In optimal control, we are interested in determining the optimal way to control this process in order to achieve a predefined goal. This goal could be minimizing a cost, minimizing the time required to reach a certain process state, maximizing the profit or some other quantity, etc. In addition, we will often require the controlled process to respect certain constraints. Velocity, heat, pressure, or volume limits which must stay within predefined ranges come to mind here.

We note, however, that the optimal control we obtained from the solution of such a problem is valid only for the very set of model parameters we used when we computed that control. We were ignoring the fact that in reality these parameters may, and most often will, vary and differ from the values we assumed. Even worse, critical constraints may be satisfied only under the assumption that the model's parameterization is precise.

Now, the ability to estimate uncertainty of model parameters – if observed data for the model is available – is perfectly complemented by optimization strategies that take these uncertainties into account. A first step is to analyze the effects of the computed control under variations of the model parameters as indicated by the estimated uncertainties. It is vitally important at least to quantify both the extent to which we may miss the prescribed goal, and the violation of the critical constraints, under a given uncertainty of the parameterization. An even more desirable ability, of course, is to formulate and solve a *robust* optimal control problem that takes the estimated uncertainties into account. Employing the optimal control scheme resulting from the solution of such a problem, the controlled process should exhibit *robustness* against deviations of the model parameters. The severity of the previously described and analyzed effects should be visibly diminished.

In this thesis we present a bi-level problem formulation for solving equality- and inequality-constrained nonlinear problems under uncertainties. A computationally feasible approach based on a linearization of the uncertainty sets is presented and transferred to optimal control of uncertain ODE systems. We describe two advantageous possibilities for generating required sensitivities of the ODE system's solution with respect to the uncertain parameters.

A highlight of this thesis is the efficient combination of implicitly discontinuous ODE models with techniques for optimal control of uncertain ODE systems. For robust optimal control, we require sensitivities of second order of the ODE system's solution, whose treatment during an implicit switch poses a challenge. We describe a new numerical method which has not been available prior to this work, that enabled us to treat optimal control problems of uncertain ODE systems with implicitly defined discontinuities and switches.

Software Implementations

All numerical methods presented in this thesis have actually been implemented within the optimal control software package *MUSCOD-II* (Diehl et al. [12], Leineweber [38]).

We implemented a new explicit error-controlled continuous Runge-Kutta method named *RKFSWT* (**R**unge-**K**utta-**F**ehlberg with **swit**ches) for the solution of non-stiff ODE systems

that is capable of detecting and treating implicit model discontinuities and switches. This new integrator has been used for all optimal control computations presented in this thesis.

We developed a new software tool named *QuickFit* for easy parameter estimation on initial-value problems. The tool is now being actively used at *DaimlerChrysler AG* in *Stuttgart*. It works in conjunction with either the developed Runge-Kutta integrator *RKFSWT*, or the implicit BDF method *DAESOL* (Bauer [2]).

The numerical method for robust optimal control to be presented has been implemented in the new *ROBUST* framework within *MUSCOD-II* [12, 38], and has been used for all robust optimal control computations in this thesis.

1.2 A Guiding Example: Powertrain Oscillations

As a guiding example for the demonstration of the numerical methods to be presented, we tackle the problem of oscillations in a car's powertrain. The problem of powertrain parametrization, identification of uncertainties, optimization, and robustification against the identified uncertainties will guide us through this thesis. Each method we present will be applied to this guiding example to demonstrate its actual applicability and effectiveness.

The power train comprises all drive-related car components from the engine to the wheels. As shown in Fig. 1.1 these parts usually encompass a flywheel, the gearbox, the cardan shaft, the axle drive, and the side shafts together with wheels and tires.

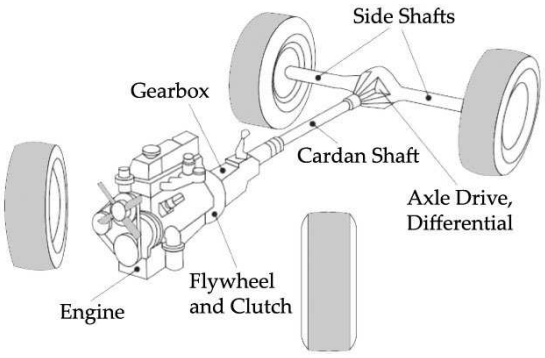


Fig. 1.1: Selected parts of a typical powertrain (Stelzer [55]).

Problem Description

Fast changes of the motor torque typically occur when a car is being accelerated. They induce very specific oscillations of the car's powertrain. As these oscillations significantly degrade the driving comfort, engineers make an effort to come up with engine control schemes that result in oscillation-free acceleration while maintaining high responsiveness and agility.

In the past, engineers frequently obtained such control laws by trial and error, slowly building up experience that allowed them to tune the powertrain behavior of every new car type and model. There was, however, no guarantee for the optimality of the control schemes in terms of agility, responsiveness, or maximum reduction of the oscillations. Even more so, nothing could be said or done concerning the robustness of those schemes against deviations from the assumed powertrain specifications. This is where the *REI/EP* department of *DaimlerChrysler AG* has been working on improving the situation.

The Powertrain Model

An ODE model of the powertrain found in a *Mercedes C Class* has been constructed at *Daimler-Chrysler AG* and has been improved for the presented research topics. The model is described to some detail in the following section. Gear shift, clutch, and differential are not modeled since the conducted research is restricted to preselected gears. Only a single side shaft is included (*single-track model*), being sufficient for the modeling of straight-ahead driving. Six implicit discontinuities arise from the exact modeling of mechanical plays and friction phenomena. Their vital importance for the precise representation of the oscillation phenomena gives us one additional motivation for the use of an ODE solver with the capability to detect and treat implicit discontinuities.

Parameter Estimation and Model Identification

We apply the algorithm for parameter and uncertainty estimation that enables us to fully identify a minimal powertrain model from real-world measurements taken on the test track. As a result we obtain a very satisfying representation of the real powertrain's behavior.

Controlling Powertrain Oscillations

To demonstrate the nonlinear optimal control methods, an optimization problem is formulated to measure and minimize oscillations of the powertrain observed during acceleration of the car. A mechanical constraint is motivated that must be satisfied at a certain implicitly defined point in time. Acceleration scenarios starting in different initial system states are evaluated. Using the presented numerical methods, optimal engine control schemes are obtained that diminish powertrain oscillations to a very satisfying extent.

Robust Acceleration into Traction Mode

We focus on the problem of accelerating the powertrain from *coasting mode* into *traction mode*. The sensitivity of the mechanical constraint with respect to deviations in several selected parameters and system states is evaluated. The results clearly motivate the need for robust optimal engine control schemes. Such schemes are obtained by solving an appropriate robust optimal control problem using the presented numerical method.

1.3 Outline of the Thesis

Chapter 2 The employed ODE model of a powertrain, designed at *DaimlerChrysler AG* in *Stuttgart-Untertürkheim*, is described [55].

Chapter 3 The second chapter briefly reviews the theory of nonlinear programming. We present the sequential quadratic programming (SQP) method [27, 28, 49, 43] as an efficient and reliable method for the solution of general nonlinear programs such as those obtained from multiple-shooting discretization of ODE optimal control problems [6, 7]. For the efficient solution of constrained least-squares problems, a generalization of the constrained Gauß-Newton (CGN) method [6, 52] is presented.

Chapter 4 This chapter presents an explicit Runge-Kutta integrator [15, 20, 37, 51] capable of treating implicit discontinuities (switches) in non-stiff ODE models. The efficient detection of such switches requires the implementation of a *continuous extension* [16, 20, 44] to this popular ODE solver. The guiding example features nonlinear springs with a play as instances of implicit discontinuities. In addition, it will be seen in Chapter 6 that the switch detection feature gives us considerable additional freedom in formulating optimal and robust optimal control problems.

Chapter 5 Deals with parameter estimation techniques for initial-value problems (IVPs). We motivate least-squares fits against observed data as maximum likelihood estimators assuming Gaussian distributed measurement errors [43]. Additional statistical information available from the solution of these problems allows us to obtain confidence information in the form of uncertainty sets for estimated parameter values [1, 6].
Convincing results for the guiding example — the powertrain oscillations problem — obtained from fits to real-world measurement data observed on a test track are presented. We derive a reduced non-stiff powertrain model to be used for subsequent optimization tasks.

Chapter 6 A class of nominal optimal control problems, and their solution using the direct multiple shooting method [6, 7] is reviewed in Chapter 6. The powertrain oscillation problem setup is discussed, and computational results are presented. We obtain engine control schemes that allow for virtually oscillation-free acceleration of the powertrain.

Chapter 7 Focuses on a class of nonlinear min-max problems for robust optimal control [3], and presents a computationally feasible approximation to this problem class that is based on linearization of the uncertainty sets [13, 14, 35, 40]. Subsequently, we elaborate on a new algorithm that allows for the solution of this class of problems within the optimal control software package *MUSCOD-II*. Emphasis is put on the efficient generation of first- and second- order sensitivities [6, 13, 42] and their interaction with implicit discontinuities present in the ODE model.
Finally, we turn again to the guiding example, this time focusing on the problem of accelerating the powertrain from coasting to traction mode, while respecting an internal rotation speed constraint. Using the presented algorithm, engine control schemes satisfying this requirement that exhibit some robustness against uncertainty in the model parameters, the powertrain's state, and the applied engine controls are obtained.

Chapter 8 The final chapter contains a summary of the presented theory, numerical methods, and results for the guiding example. We mention several interesting open questions that require further research, and show extension points for the current software implementations.

Appendices The three appendices to this thesis focus on software architecture and implementation topics, and serve as brief user's manuals for those interested in using the new extensions to the optimal control software package *MUSCOD-II* [38].

In Appendix A a description and guide to the new parameter estimation tool *QuickFit* can be found. While universally applicable to initial-value problems for ODE and DAE models, this tool was created for *DaimlerChrysler AG* in order to enable and simplify the process of estimating parameters in present and upcoming powertrain models.

Appendix B intends to serve as a user's guide to the newly implemented implicit switch extension to the optimal control software package *MUSCOD-II*.

Appendix C does the same for the robust optimal control extension to *MUSCOD-II*.

Lists of all figures, listings, and tables can be found in the back of the thesis. An extensive nomenclature, a bibliography, and an index of the most important keywords completes this work.

Chapter 2

The Powertrain Model

This chapter describes an improved version of the powertrain model designed and implemented in *Simulink* at *DaimlerChrysler AG* in *Stuttgart-Untertürkheim*. It consists of 8 differential states, and uses 31 model parameters, many of which are unknown in advance.
The joint work resulted in several improvements over the original model described by Stelzer [55], which include a reformulation of the ODE system to achieve better numerical stability when applying internal numerical differentiation (IND, cf. Bock [4, 5, 6]) to the model, the addition of four extra outputs to be fitted against measurement data, and the introduction of the dual-mass flywheel's end stop into the model.

2.1 Model Parts

We briefly describe equations for the individual parts of the powertrain in order to motivate the resulting ODE system. While improvements are already incorporated, this section is guided by Stelzer [55], which may be consulted for a more detailed discussion.

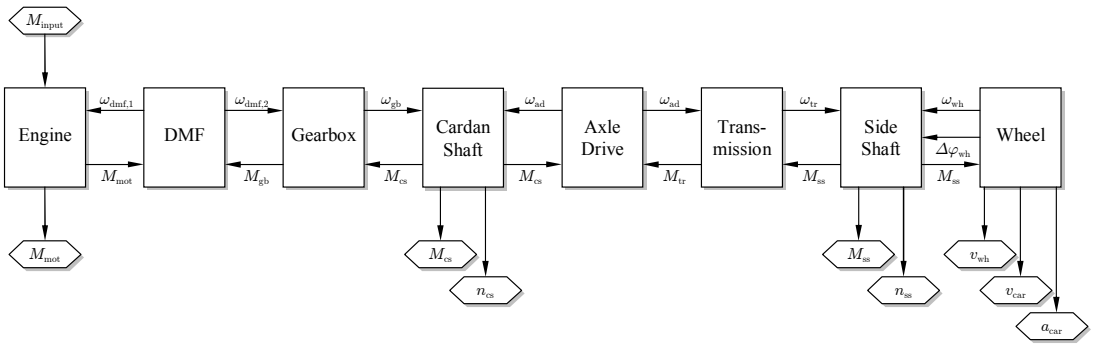


Fig. 2.1: Signal flow in the powertrain model.

Engine

Depending on the scenario — parameter estimation or optimal control — the effective engine torque M_{input} is supplied to the model as external data obtained through real-world measurements (parameter estimation), or it is the solution to be found for an optimal control problem.

The engine torque M_{mot} results from a reduction by a braking torque M_{fric} due to friction:

$$M_{\text{mot}} := M_{\text{input}} - M_{\text{fric}}, \tag{2.1a}$$

$$M_{\text{fric}} := d_{\text{mot}}\, \omega_{\text{dmf},1}(t). \tag{2.1b}$$

Dual-Mass Flywheel

The dual-mass flywheel's (DMF) purpose is to act as a low-pass filter for the engine torque, which shows peaks from the individual ignitions. While these peaks are not present in the observed data sets actually used, it is nonetheless crucial to include the dual-mass flywheel in the model in order to properly account for delays and oscillations applied to the torque signal as it propagates through the powertrain.

The DMF's primary mass $J_{\mathrm{dmf},1}$ is fixed to the engine's mass J_{mot} and is accelerated by the engine torque M_{mot}, while braked by friction $M_{\mathrm{fric},1}$ and the spring's self-aligning torque M_{spring}.

$$\begin{aligned} (J_{\mathrm{dmf},1} + J_{\mathrm{mot}})\ \dot{\omega}_{\mathrm{dmf},1}(t) &= M_{\mathrm{mot}} - M_{\mathrm{fric},1} - M_{\mathrm{spring}}, && \text{(2.2a)} \\ M_{\mathrm{fric},1} &:= d_{\mathrm{dmf},1}\ \omega_{\mathrm{dmf},1}(t). && \text{(2.2b)} \end{aligned}$$

Consequently, the spring accelerates the secondary mass $J_{\mathrm{dmf},2}$ which is braked by friction $M_{\mathrm{fric},2}$ and the retransmitted torque M_{gb} that back-propagates from the cardan shaft.

$$\begin{aligned} J_{\mathrm{dmf},2}\ \dot{\omega}_{\mathrm{dmf},2}(t) &= M_{\mathrm{spring}} - M_{\mathrm{fric},2} - M_{\mathrm{gb}}, && \text{(2.3a)} \\ M_{\mathrm{fric},2} &:= d_{\mathrm{dmf},2}\ \omega_{\mathrm{dmf},2}(t). && \text{(2.3b)} \end{aligned}$$

The spring features two different spring constants k^{+}_{dmf} and k^{-}_{dmf} which are selected depending on the spring torsion's direction, and it includes a play p_{dmf} as well as an additional friction term.

$$\begin{aligned} \Delta\dot{\varphi}_{\mathrm{dmf}}(t) &= \omega_{\mathrm{dmf},1}(t) - \omega_{\mathrm{dmf},2}(t), && \text{(2.4a)} \\ M_{\mathrm{spring}} &:= d_{\mathrm{dmf}}\ \Delta\dot{\varphi}_{\mathrm{dmf}}(t) + k_{\mathrm{dmf}}\big(\Delta\varphi_{\mathrm{dmf}}(t)\big), && \text{(2.4b)} \\ k_{\mathrm{dmf}}(\Delta\varphi) &:= \begin{cases} k^{-}_{\mathrm{dmf}}(\Delta\varphi + p_{\mathrm{dmf}}) & \text{if} \quad \Delta\varphi < -p_{\mathrm{dmf}}, \\ 0 & \text{if} \quad |\Delta\varphi| \leqslant p_{\mathrm{dmf}}, \\ k^{+}_{\mathrm{dmf}}(\Delta\varphi - p_{\mathrm{dmf}}) & \text{if} \quad \Delta\varphi > +p_{\mathrm{dmf}}. \end{cases} && \text{(2.4c)} \end{aligned}$$

The spring's torsion $|\Delta\varphi_{\mathrm{dmf}}|$ also has an end-stop $\Delta\varphi_{\max}$. An implementation of both $k(\Delta\varphi)$ and the end-stop that overcomes the introduced non-differentiability is described in Section 2.2.

Gearbox

Since the simulation of gear shifts is not of interest, the gearbox can effectively be reduced to a simple transmission of the DMF's secondary side angular velocity $\omega_{\mathrm{dmf},2}$. M_{gb} is the corresponding back-transmission of the cardan shaft's braking torque M_{cs}, increased by some engine speed- and gear-dependent loss $M_{\mathrm{loss,gb}}$ due to friction, see Eq. (2.16) and Fig. 2.2 on page 19.

$$\begin{aligned} \omega_{\mathrm{gb}}(t) &:= \frac{1}{i_{\mathrm{gb}}}\ \omega_{\mathrm{dmf},2}(t), && \text{(2.5a)} \\ M_{\mathrm{gb}} &:= \begin{cases} \frac{1}{i_{\mathrm{gb}}\eta_{\mathrm{gb}}}\ M_{\mathrm{cs}} + M_{\mathrm{loss,gb}}\big(\omega_{\mathrm{dmf},2}(t)\big) & \text{if} \quad \Delta\varphi_{\mathrm{dmf}}(t) < 0, \\ \frac{\eta_{\mathrm{gb}}}{i_{\mathrm{gb}}}\ M_{\mathrm{cs}} - M_{\mathrm{loss,gb}}\big(\omega_{\mathrm{dmf},2}(t)\big) & \text{if} \quad \Delta\varphi_{\mathrm{dmf}}(t) \geqslant 0. \end{cases} && \text{(2.5b)} \end{aligned}$$

Cardan Shaft

In this complete version of the powertrain model the cardan shaft is included as a massless elasticity with damping. Thus the cardan shaft's moment of inertia is covered by the neighbouring moments $J_{\mathrm{dmf},2}$ and J_{ag}. As the cardan shafts used in cars manufactured by the *Mercedes Car Group* are comparatively stiff, it will be seen that this part of the powertrain has only a negligible influence on the oscillation phenomena.

$$M_{\mathrm{cs}} := c_{\mathrm{cs}}\Delta\varphi_{\mathrm{ad}}(t) + d_{\mathrm{cs}}\Delta\dot{\varphi}_{\mathrm{ad}}(t). \qquad \text{(2.6)}$$

Axle Drive

The axle drive is accelerated by the cardan shaft's torque M_{cs} while braked by friction $M_{\mathrm{fric,ad}}$ and the back-transmitted torque M_{tr} from the side shaft.

$$\begin{aligned} J_{\mathrm{ad}}\ \dot{\omega}_{\mathrm{ad}}(t) &= M_{\mathrm{cs}} - M_{\mathrm{fric,ad}} - M_{\mathrm{tr}}, && \text{(2.7a)} \\ M_{\mathrm{fric,ad}} &:= d_{\mathrm{ad}}\omega_{\mathrm{ad}}(t) && \text{(2.7b)} \\ \Delta\dot{\varphi}_{\mathrm{ad}}(t) &= \omega_{\mathrm{gb}}(t) - \omega_{\mathrm{ad}}(t). && \text{(2.7c)} \end{aligned}$$

Transmission

Since the model only includes a single side shaft, the differential is omitted and only the corresponding transmission is included in the model. Just like the gearbox, this transmission also includes a speed-dependent torque loss $M_{\mathrm{loss,tr}}$, see Eq. (2.16) and Fig. 2.3.

$$\begin{aligned} \omega_{\mathrm{tr}}(t) &:= \frac{1}{i_{\mathrm{tr}}}\ \omega_{\mathrm{ad}}(t), && \text{(2.8a)} \\ M_{\mathrm{tr}} &:= \begin{cases} \frac{1}{i_{\mathrm{tr}}\eta_{\mathrm{tr}}}\ M_{\mathrm{ss}} + M_{\mathrm{loss,tr}}\big(\omega_{\mathrm{ad}}(t)\big) & \text{if} \quad \Delta\varphi_{\mathrm{ss}}(t) < 0, \\ \frac{\eta_{\mathrm{tr}}}{i_{\mathrm{tr}}}\ M_{\mathrm{ss}} - M_{\mathrm{loss,tr}}\big(\omega_{\mathrm{ad}}(t)\big) & \text{if} \quad \Delta\varphi_{\mathrm{ss}}(t) \geqslant 0. \end{cases} && \text{(2.8b)} \end{aligned}$$

Side Shaft

Like the cardan shaft, the side shafts are modelled as mass-less elasticities. Only a single side shaft is included in the present model (*single-track model*). Like in the DMF, the side shaft spring features a play p_{ss} whose implementation is discussed in Section 2.2.

$$\begin{aligned} M_{\mathrm{ss}} &:= k_{\mathrm{ss}}\big(\Delta\varphi_{\mathrm{wh}}(t)\big) + d_{\mathrm{ss}}\Delta\dot{\varphi}_{\mathrm{wh}}(t), && \text{(2.9a)} \\ k_{\mathrm{ss}}(\Delta\varphi) &:= \begin{cases} c_{\mathrm{ss}}(\Delta\varphi + p_{\mathrm{ss}}) & \text{if} \quad \Delta\varphi < -p_{\mathrm{ss}}, \\ 0 & \text{if} \quad |\Delta\varphi| \leqslant p_{\mathrm{ss}}, \\ c_{\mathrm{ss}}(\Delta\varphi - p_{\mathrm{ss}}) & \text{if} \quad \Delta\varphi > +p_{\mathrm{ss}}. \end{cases} && \text{(2.9b)} \end{aligned}$$

Wheels

The last part of the model includes the wheel's moment of inertia J_{wh} as well as the translation of the angular forces within the powertrain into acceleration of the car. Tyre slip is crucial to the model's correctness as it reduces both the effective accelerating force and the oscillations observed in the side shaft.
The wheel is accelerated by the side shaft's torque M_{ss}, while braked by friction and the tangential force F_{acc} acting on the tyre.

$$\begin{aligned} J_{\mathrm{wh}}\ \dot{\omega}_{\mathrm{wh}}(t) &= M_{\mathrm{ss}} - M_{\mathrm{fric,wh}} - r_{\mathrm{tyre}}F_{\mathrm{acc}}, && \text{(2.10a)} \\ M_{\mathrm{fric,wh}} &:= d_{\mathrm{wh}}\omega_{\mathrm{wh}}(t), && \text{(2.10b)} \\ \Delta\dot{\varphi}_{\mathrm{wh}}(t) &= \omega_{\mathrm{tr}}(t) - \omega_{\mathrm{rad}}(t). && \text{(2.10c)} \end{aligned}$$

Due to tyre slip the wheel's speed may differ from the vehicle's speed, which is therefore introduced as an additional model state. The car is accelerated by the force F_{acc} while braked by rolling friction F_{roll}, air resistance F_{air} and downhill force F_{down}.

$$\begin{aligned} m_{\mathrm{car}}\ \dot{v}_{\mathrm{car}}(t) &= F_{\mathrm{acc}} - F_{\mathrm{roll}} - F_{\mathrm{air}} - F_{\mathrm{down}}, && \text{(2.11a)} \\ F_{\mathrm{roll}} &:= \mu_{\mathrm{roll}}\ m_{\mathrm{car}}\ g\ \cos\beta, && \text{(2.11b)} \\ F_{\mathrm{air}} &:= \tfrac{1}{2}\ v_{\mathrm{car}}^2(t)\ \varrho_{\mathrm{air}}\ A_{\mathrm{car}}\ c_{\mathrm{w}}, && \text{(2.11c)} \\ F_{\mathrm{down}} &:= m_{\mathrm{car}}\ g\ \sin\beta. && \text{(2.11d)} \end{aligned}$$

The tangential force F_{acc} is calculated from the normal force $F_{\mathrm{stat}} + F_{\mathrm{dyn}}$ by way of the ratio function μ.

$$F_{\mathrm{acc}} := (F_{\mathrm{stat}} + F_{\mathrm{dyn}})\ \mu\Big(s\big(\omega_{\mathrm{wh}}(t), v_{\mathrm{car}}(t)\big)\Big), \tag{2.12a}$$

$$F_{\mathrm{stat}} := m_{\mathrm{car}}\ g\ \left(\frac{l_{\mathrm{f}}}{l_{\mathrm{wb}}}\cos\beta + \frac{h_{\mathrm{mc}}}{l_{\mathrm{wb}}}\sin\beta\right), \tag{2.12b}$$

$$F_{\mathrm{dyn}} := m_{\mathrm{car}}\ \dot{v}_{\mathrm{car}}(t)\ \frac{h_{\mathrm{mc}}}{l_{\mathrm{wb}}}, \tag{2.12c}$$

The function μ models the ratio between tangential and normal force depending on the slip s, according to a popular approximation due to Pacejka and Bakker [47].

$$\mu(s) := \mu_{\max}\ \sin\big(c_{\mathrm{pac}}\ \arctan(b_{\mathrm{pac}}\ s)\big). \tag{2.13}$$

The slip s describes the difference between a wheel's tangential speed $r\ \omega$ and the car's speed v.

$$s(\omega, v) := \frac{r_{\mathrm{tyre}}\ \omega - v}{r_{\mathrm{tyre}}\ \omega}. \tag{2.14}$$

Finally, note that by way of F_{acc} Eq. (2.11) contains $\dot{v}_{\mathrm{car}}(t)$ on both sides. Solving for $\dot{v}_{\mathrm{car}}(t)$ splits F_{dyn} and yields

$$m_{\mathrm{car}}\left(1 - \frac{h_{\mathrm{mc}}}{l_{\mathrm{wb}}}\ \mu(s)\right)\ \dot{v}_{\mathrm{car}}(t) = F_{\mathrm{stat}}\ \mu(s) - F_{\mathrm{roll}} - F_{\mathrm{air}} - F_{\mathrm{down}}, \tag{2.15}$$

with $s := s\big(\omega_{\mathrm{wh}}(t), v_{\mathrm{car}}(t)\big)$.

2.2 Details of the Implementation

Gearbox and Transmission

Stelzer [55] approximated the nonlinear function $M_{\mathrm{loss,gb}}(\omega_{\mathrm{dmf},2})$ by performing a second-order polynomial fit (2.16) against a series of torque loss values at certain rotation speeds, known from the corresponding data sheets. The results are reprinted in Fig. 2.2 and Tab. 2.1.

$$M_{\mathrm{loss},*}\,(\omega_*) := \left(\lambda_2\left(\frac{60\ \mathsf{s}}{2\pi}\omega_*\right)^2 + \lambda_1\left(\frac{60\ \mathsf{s}}{2\pi}\omega_*\right) + \lambda_0\right)\quad \mathsf{Nm} \tag{2.16}$$

Just as in the gearbox, the torque loss $M_{\mathrm{loss,tr}}(\omega_{\mathrm{ad}})$ in the transmission is approximated using a second-order polynomial as shown in Fig. 2.3 and Tab. 2.2.

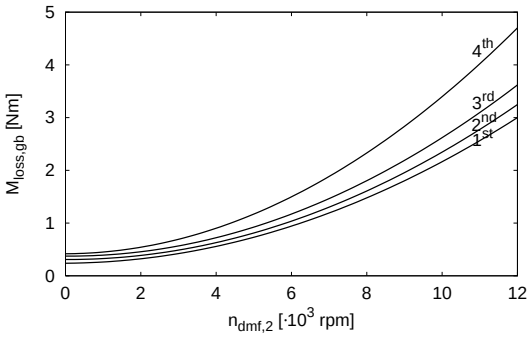


Fig. 2.2: Gearbox torque loss.

Gear	$\lambda_2\left[\cdot 10^{-8}\right]$	$\lambda_1\left[\cdot 10^{-6}\right]$	$\lambda_0\left[\cdot 10^{-1}\right]$
1^{st}	1.88	4.54	2.3859
2^{nd}	2.06	−2.3284	3.0905
3^{rd}	2.2916	−4.5643	3.7377
4^{th}	2.95	2.96	4.1826

Tab. 2.1: Gearbox torque loss polynomials.

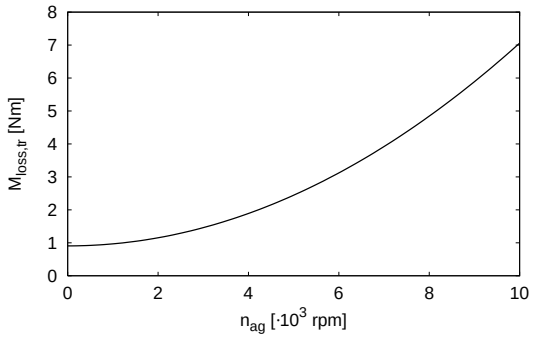


Fig. 2.3: Transmission torque loss.

$\lambda_2\left[\cdot 10^{-8}\right]$	$\lambda_1\left[\cdot 10^{-7}\right]$	$\lambda_0\left[\cdot 10^{-1}\right]$
6.1535	−1.4246	9.0689

Tab. 2.2: Torque loss polynomial coefficients for the transmission.

Non-Differentiabilities

The model equations presented in the previous section contain several non-differentiabilities:

1. Springs with a play, found in the DMF and the side shaft;
2. Loss of torque due to friction, depending on direction of the powertrain torsion's rates of change $\Delta\varphi_{\mathrm{dmf}}$ and $\Delta\varphi_{\mathrm{ss}}$.

Since the existence of higher-order derivatives is crucial to guarantee convergence of numerical algorithms for the solution of the ODE system, one may think of approximating the respective equations by smooth functions as shown in Eq. (2.17) and Fig. 2.4 in the case of the DMF,

$$\begin{aligned} k(\Delta\varphi) \quad = \quad & \pm\, k_{\mathrm{dmf}}^{\pm}\gamma \log\left(\exp\left(\frac{\pm\Delta\varphi - p_{\mathrm{dmf}}}{\gamma}\right)+1\right) \\ & \pm\, k_{2}^{\pm}\gamma_2 \log\left(\exp\left(\frac{\pm\Delta\varphi - \Delta\varphi_{\max}}{\gamma_2}\right)+1\right). \end{aligned} \tag{2.17}$$

This approach, however, is frequently found to be too inexact. In addition, the closer the non-differentiability is approximated, the more stiffness will be introduced. Explicit integrator methods such as the one presented in Chapter 4 may fail to solve the resulting approximative smooth ODE model due to its stiffness. A more sophisticated and exact method to treat this and other types of non-differentiabilities is discussed in Section 4.3.

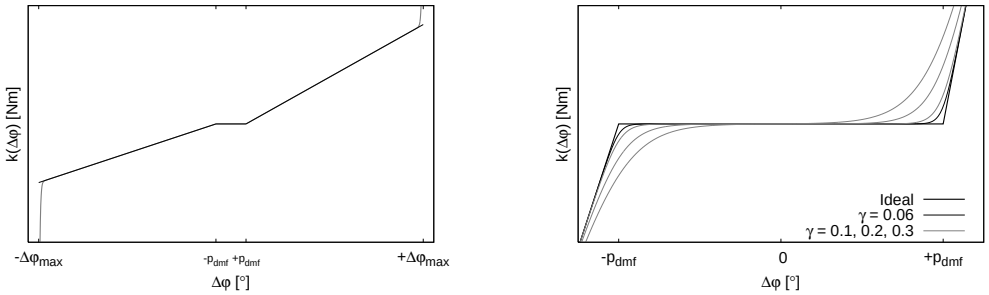


Fig. 2.4: Spring torsion dependent friction in the dual-mass flywheel.

2.3 Summary and Reference

In this section we collect model equations, parameters, and outputs from the previous one. The system of ordinary differential equations is rewritten in Eq. (2.18) to fit the canonic form. A comprehensive list of all introduced parameters is given in Tab. 2.4. Model outputs that can be compared to observed data are listed in Tab. 2.3.

System of Ordinary Differential Equations

In Eq. (2.18) we summarize the ordinary differential equations from the previous one and reformulate them to fit the canonic form.

$$\dot{\omega}_{\mathrm{dmf},1} = \frac{1}{J_{\mathrm{mot}} + J_{\mathrm{dmf},1}} \left(M_{\mathrm{mot}} - M_{\mathrm{fric},1} - M_{\mathrm{spring}} \right), \tag{2.18a}$$

$$\dot{\omega}_{\mathrm{dmf},2} = \frac{1}{J_{\mathrm{dmf},2}} \left(M_{\mathrm{spring}} - M_{\mathrm{fric},2} - M_{\mathrm{gb}} \right), \tag{2.18b}$$

$$\Delta\dot{\varphi}_{\mathrm{dmf}} = \omega_{\mathrm{dmf},1} - \omega_{\mathrm{dmf},2}, \tag{2.18c}$$

$$\dot{\omega}_{\mathrm{ad}} = \frac{1}{J_{\mathrm{ad}}} \left(M_{\mathrm{cs}} - M_{\mathrm{fric,ad}} - M_{\mathrm{tr}} \right), \tag{2.18d}$$

$$\Delta\dot{\varphi}_{\mathrm{ad}} = \omega_{\mathrm{gb}} - \omega_{\mathrm{ad}}, \tag{2.18e}$$

$$\dot{\omega}_{\mathrm{wh}} = \frac{1}{J_{\mathrm{wh}}} \left(M_{\mathrm{ss}} - M_{\mathrm{fric,wh}} - r_{\mathrm{tyre}} F_{\mathrm{acc}} \right), \tag{2.18f}$$

$$\Delta\dot{\varphi}_{\mathrm{wh}} = \omega_{\mathrm{tr}} - \omega_{\mathrm{wh}}, \tag{2.18g}$$

$$\dot{v}_{\mathrm{car}} = \frac{1}{m_{\mathrm{car}} \left(1 - \frac{h_{\mathrm{mc}}}{l_{\mathrm{wb}}} s \right)} \left(F_{\mathrm{stat}}\, \mu(s) - F_{\mathrm{roll}} - F_{\mathrm{air}} - F_{\mathrm{down}} \right). \tag{2.18h}$$

Observable Model Outputs

The following table lists outputs of the model that are fitted to observed data in order to obtain estimates of the unknown model parameters.

Output	Description	Value	Unit
n_{mot}	Engine speed.	$\left(\frac{60\ \mathsf{s}}{1\ \mathsf{min}} \frac{1}{2\pi}\right) \omega_{\mathrm{dmf},1}$	min^{-1}
M_{cs}	Cardan shaft torque.	M_{cs}	Nm
n_{cs}	Cardan shaft revolutions per second.	$\left(\frac{60\ \mathsf{s}}{1\ \mathsf{min}} \frac{1}{2\pi}\right) \omega_{\mathrm{gb}}$	min^{-1}
M_{ss}	Side shaft torque.	M_{ss}	Nm
n_{ss}	Side shaft revolutions per second.	$\left(\frac{60\ \mathsf{s}}{1\ \mathsf{min}} \frac{1}{2\pi}\right) \omega_{\mathrm{tr}}$	min^{-1}
v_{wh}	Rear wheel velocity.	$\left(\frac{3600\ \mathsf{s}}{1000\ \mathsf{m}}\right) r_{\mathrm{tyre}}\, \omega_{\mathrm{wh}}$	$\mathsf{km/h}$
v_{car}	Car velocity.	$\left(\frac{3600\ \mathsf{s}}{1000\ \mathsf{m}}\right) v_{\mathrm{car}}$	$\mathsf{km/h}$
a_{car}	Car acceleration.	$\dot{v}_{\mathrm{car}}$	$\mathsf{m/s^2}$

Tab. 2.3: List of observable model outputs.

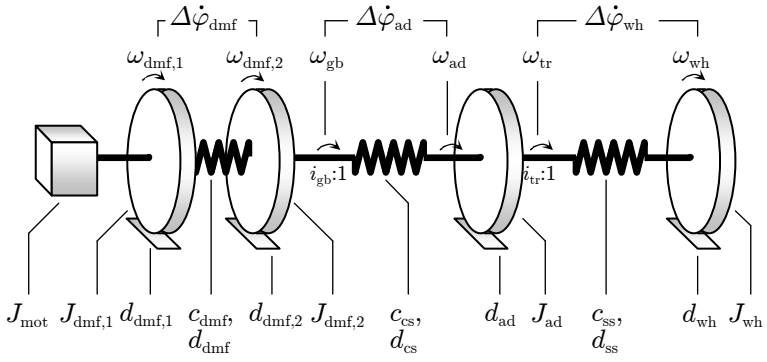


Fig. 2.5: Schematic of the powertrain model.

Model Parameters

Tab. 2.4 summarizes all model parameters along with their unit and a brief description. Unknown parameters to be included in the parameter estimation problem presented in Chapter 5 are marked. Crosses in parentheses indicate selected parameters which get included in the estimation problem to improve the quality of fit although a nominal value is known. For non-disclosure reasons, however, these values are not listed here.

Symbol	Description	Unit	Unknown
d_{mot}	Engine block damping coefficient.	$\mathsf{Nm\,s/_\circ}$	
J_{mot}	Engine block moment of inertia.	$\mathsf{kg\,m^2}$	
d_{dmf}	DMF common damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
$d_{\mathrm{dmf},1}$	DMF primary wheel damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
$d_{\mathrm{dmf},2}$	DMF secondary wheel damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
$J_{\mathrm{dmf},1}$	DMF primary wheel moment of inertia.	$\mathsf{kg\,m^2}$	×
$J_{\mathrm{dmf},2}$	DMF secondary wheel moment of inertia.	$\mathsf{kg\,m^2}$	×
k^+_{dmf}	DMF spring coefficient of resilience.	$\mathsf{Nm/_\circ}$	
k^-_{dmf}	DMF spring coefficient of resilience.	$\mathsf{Nm/_\circ}$	
p_{dmf}	DMF spring play.	$\circ$	
$\Delta\varphi_{\max}$	DMF spring torsion's hard limit.	$\circ$	
η_{gb}	Gearbox efficiency.	–	
i_{gb}	Gearbox transmission ratio.	–	
c_{cs}	Cardan shaft coefficient of friction.	$\mathsf{Nm/_\circ}$	×
d_{cs}	Cardan shaft damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
d_{ad}	Axle drive damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
J_{ad}	Axle drive damping coefficient.	$\mathsf{kg\,m^2}$	×
η_{tr}	Axle drive transmission efficiency.	–	
i_{tr}	Axle drive transmission ratio.	–	
c_{ss}	Side shaft coefficient of friction.	$\mathsf{Nm/_\circ}$	×
d_{ss}	Side shaft damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
p_{ss}	Side shaft play.	$\circ$	×
d_{wh}	Wheel damping coefficient.	$\mathsf{Nm\,s/_\circ}$	×
J_{wh}	Wheel moment of inertia.	$\mathsf{kg\,m^2}$	×
μ_{roll}	Tire's rolling friction coefficient.	–	(×)
r_{tyre}	Rear tire radius.	m	(×)
$\mu_{\max}$	Peak traction.	–	(×)
b_{pac}	Pacejka coefficient b.	–	(×)
c_{pac}	Pacejka coefficient c.	–	(×)
A_{car}	Car's abutting face.	$\mathsf{m^2}$	×
c_{w}	Car chassis drag coefficient.	–	(×)
h_{mc}	Mass center's height above ground.	m	
l_{wb}	Rear wheelbase.	m	
l_{f}	Front axle's distance to mass center.	m	
m_{car}	Car mass during measurements.	kg	(×)
β	Street's slope.	$\circ$	
ϱ_{air}	Air density.	$\mathsf{kg/m^3}$	(×)
g	Unit of acceleration.	$\mathsf{m/s^2}$	

Tab. 2.4: List of model parameters sorted by power train part.

Chapter 3

Nonlinear Programming Theory and Algorithms

In this chapter we give an overview over the theory of nonlinear programming, and presents two efficient numerical methods for the solution of constrained nonlinear problems (NLPs), and constrained nonlinear problems of least squares.
Initially we consider a class of equality- and inequality-constrained nonlinear problems. NLPs of the considered structure will arise from the discretization of continuous optimal control problems using the direct multiple-shooting method (cf. Bock [6], Bock and Plitt [7]) in Chapter 6. We present well-known local optimality conditions based on first- and second-order derivatives of the objective and constraints. Then the sequential quadratic programming (SQP) approach for the efficient and reliable solution of such problems is briefly presented.
Closely related is the constrained Gauß-Newton (CGN) method for the solution of constrained problems of least-squares, such as those arising from parameter estimation problems discussed in Chapter 5.

3.1 Nonlinear Programming Theory

In this section we consider a general nonlinear problem (NLP) of the form

Definition 3.1. *Nonlinear Problem*

$$\min_{\boldsymbol{x}\in\mathcal{D}} \quad f(\boldsymbol{x}) \tag{3.1a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}, \tag{3.1b}$$

$$\boldsymbol{h}(\boldsymbol{x}) \geqslant \boldsymbol{0}, \tag{3.1c}$$

where we assume $f \in \mathscr{C}^2(\mathcal{D}, \mathbb{R})$, $\boldsymbol{g} \in \mathscr{C}^2(\mathcal{D}, \mathbb{R}^{n_g})$, *and* $\boldsymbol{h} \in \mathscr{C}^2(\mathcal{D}, \mathbb{R}^{n_h})$. *The domain* $\mathcal{D}$ *is a subset of* $\mathbb{R}^{n_x}$.

We strive to identify a point $\boldsymbol{x}$ within the domain $\mathcal{D}$ that minimizes the objective function f (3.1) defined thereon. At the same time, we restrict the set of allowed points to the implicitly defined subset of the domain $\mathcal{D}$ that satisfies the given equality constraints g_i, $i = 1, \ldots, n_g$ (3.1a), and inequality constraints h_i, $i = 1, \ldots, n_h$ (3.1b).

A Note about Derivatives

Throughout this chapter, we consider all vector values to be column vectors unless otherwise noted. For some vector valued function $\boldsymbol{f}(\boldsymbol{x})$ we denote with $\nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x})$ the Jacobian of first partial derivatives with respect to the vector argument $\boldsymbol{x}$,

$$\nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x}) := \left[\frac{\partial f_i}{\partial x_j}\right]_{ij}, \qquad \begin{aligned} i &= 1, \ldots, n_f, \\ j &= 1, \ldots, n_x. \end{aligned}$$

Columns of this Jacobian hold the partial derivative of $\boldsymbol{f}$ with respect to the scalar component x_j of the argument $\boldsymbol{x}$. Row vectors of this Jacobian contain gradients $\nabla_{\boldsymbol{x}} f_i(\boldsymbol{x})$ of the scalar functions f_i. Consequentially, gradients of scalar functions are also considered to be row vectors.

3.1.1 Feasibility and Optimality

In order to verify that a candidate point $\boldsymbol{x} \in \mathcal{D}$ is a minimizer of problem (3.1a), a local characterization of the nearby feasible points is required.

Definition 3.2. ***Feasible Point, Feasible Set***
*A point $\boldsymbol{x} \in \mathcal{D}$ is called a **feasible point** iff the constraints (3.1a, 3.1b) are fulfilled. The set of all feasible points is referred to as the **feasible set** $\mathcal{F}$,*

$$\mathcal{F} := \left\{ \boldsymbol{x} \in \mathcal{D} \;\middle|\; \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \;\wedge\; \boldsymbol{h}(\boldsymbol{x}) \geqslant \boldsymbol{0} \right\} \subseteq \mathbb{R}^{n_x}. \tag{3.2}$$

Definition 3.3. ***Local and Global Minimizer***
*A feasible point $\boldsymbol{x} \in \mathcal{F}$ is a **local minimizer** of the problem (3.1a) iff there exists an open ball $\mathcal{B}_\varepsilon(\boldsymbol{x}) \subseteq \mathbb{R}^{n_x}$ with $\varepsilon > 0$ such that*

$$\forall \tilde{\boldsymbol{x}} \in \mathcal{B}_\varepsilon(\boldsymbol{x}) \cap \mathcal{F} : \; f(\boldsymbol{x}) \leqslant f(\tilde{\boldsymbol{x}}). \tag{3.3}$$

*The point $\boldsymbol{x}$ is a **strict local minimizer** iff the inequality is strictly fulfilled for all $\tilde{\boldsymbol{x}} \neq \boldsymbol{x}$. A local minimizer with the smallest objective function value is referred to as a **global minimizer** of (3.1a).*

3.1.2 Constraints

Definition 3.4. ***Active Constraint, Active Set***
*An inequality constraint is called **active** in a feasible point $\boldsymbol{x} \in \mathcal{F}$ if equality holds, otherwise it is called **inactive**. The **active set** $\mathcal{A}(\boldsymbol{x})$ is the set of all indices of constraints active in the point $\boldsymbol{x}$,*

$$\mathcal{A}(\boldsymbol{x}) := \left\{ i \mid h_i(\boldsymbol{x}) = 0 \right\} \subseteq \left\{ 1, \ldots, n_h \right\}, \quad \boldsymbol{x} \in \mathcal{F}. \tag{3.4}$$

Further, let $\tilde{\boldsymbol{h}}(\boldsymbol{x})$ be the restriction of the inequality constraint vector $\boldsymbol{h}$ to the $\tilde{n}_h(\boldsymbol{x})$ inequality constraints active in the point $\boldsymbol{x}$,

$$\tilde{\boldsymbol{h}}(\boldsymbol{x}) := \big(h_i(\boldsymbol{x}) \big), \quad i \in \mathcal{A}(\boldsymbol{x}). \tag{3.5}$$

When we are searching for a local minimizer, feasibility of the iterates can in general only be maintained by moving along a nonlinear path in $\mathbb{R}^{n_x}$. Consider such a path $\boldsymbol{\alpha}$, parametrized by a scalar θ, and starting in a feasible point $\boldsymbol{\alpha}(0) = \boldsymbol{x}$. Further, denote it's tangent direction in $\theta = 0$ by $\boldsymbol{p} \in \mathbb{R}^{n_x}$. We can observe that for the path to remain feasible for some small $\theta > 0$, the direction $\boldsymbol{p}$ must at least fall into the null-space of the equality- and active inequality constraint Jacobians $\nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x})$ and $\nabla_{\boldsymbol{x}} \hat{\boldsymbol{h}}(\boldsymbol{x})$ (cf. Leineweber [38]).

Definition 3.5. ***Constraint Qualification***
*First order **constraint qualification** holds in a feasible point $\boldsymbol{x} \in \mathcal{F}$ if for every non-zero vector $\boldsymbol{p} \in \mathbb{R}^{n_x}$ satisfying*

$$\nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x})\, \boldsymbol{p} \;=\; \boldsymbol{0}, \tag{3.6a}$$

$$\nabla_{\boldsymbol{x}} \hat{\boldsymbol{h}}(\boldsymbol{x})\, \boldsymbol{p} \;\geqslant\; \boldsymbol{0}, \tag{3.6b}$$

*there exists a **feasible** $\mathscr{C}^1$ **arc** $\boldsymbol{\alpha}(\theta)$ with $\boldsymbol{\alpha}(0) = \boldsymbol{x}$ and $\frac{d\boldsymbol{\alpha}}{d\theta}(0) = p$.*

For linear constraints $\boldsymbol{g}$, $\boldsymbol{h}$, constraint qualification always holds, i.e., the feasible arc always exists for the required directions $\boldsymbol{p}$ (3.6). For nonlinear constraints, the conditions (3.6) are necessary, though not sufficient. The following theorem provides a practical sufficient test for constraint qualification.

Theorem 3.6. ***Linear Independence Constraint Qualification Condition***
*Constraint qualification holds in a feasible point $\boldsymbol{x} \in \mathcal{F}$ if the following **linear independence constraint qualification condition** is satisfied,*

$$\operatorname{rank} \begin{bmatrix} \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}) \\ \nabla_{\boldsymbol{x}} \tilde{\boldsymbol{h}}(\boldsymbol{x}) \end{bmatrix} = n_g + \tilde{n}_h(\boldsymbol{x}). \tag{3.7}$$

Definition 3.7. ***Regular Point***
*A **regular point** is a feasible point $\boldsymbol{x} \in \mathcal{F}$ in which the linear independence constraint qualification condition (3.7) holds.*

Consequentially, there may exist non-regular points satisfying constraint qualification, but not the sufficient condition (3.7). In practical problems, however, condition (3.7) will frequently be fulfilled (cf. Leineweber [38]).

3.1.3 Necessary Conditions for Optimality

The *Lagrangian*[1] *function* of problem (3.1a) plays a central role in the characterization of optimal solutions; its definition is given below.

Definition 3.8. ***Lagrangian Function***
*The **Lagrangian function** L belonging to problem (3.1a) is defined as*

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) := f(\boldsymbol{x}) - \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\mu}^{\mathsf{T}} \boldsymbol{h}(\boldsymbol{x}). \tag{3.8}$$

*The vectors $\boldsymbol{\lambda} \in \mathbb{R}^{n_g}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_h}$ are called **Lagrange multipliers**.*

The derivative of L with respect to the unknown $\boldsymbol{x}$ is easily computed as

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) - \boldsymbol{\lambda}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\mu}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{h}(\boldsymbol{x}). \tag{3.9}$$

The well-known first-order Karush-Kuhn-Tucker conditions for the local optimality of a regular point have first been derived by Karush [32], and independently by Kuhn and Tucker [36], and contain a generalization of Lagrange's multiplier rule (see, e.g., Königsberger [34]).

Theorem 3.9. ***Karush-Kuhn-Tucker Necessary Condition (Stationarity Condition)***
In a local minimizer $\boldsymbol{x}^*$ of problem (3.1a) there exist unique Lagrange multipliers $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solving the stationarity conditions

$$\begin{aligned}
\nabla_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) &= \boldsymbol{0}, && && \text{(3.10a)}\\
\boldsymbol{\mu}^* &\geqslant \boldsymbol{0}, && && \text{(3.10b)}\\
\boldsymbol{\mu}^{*\mathsf{T}} \boldsymbol{h}(\boldsymbol{x}^*) &= 0, && \textit{(complementarity)} && \text{(3.10c)}\\
\boldsymbol{g}(\boldsymbol{x}^*) &= \boldsymbol{0}, && \textit{(feasibility)} && \text{(3.10d)}\\
\boldsymbol{h}(\boldsymbol{x}^*) &\geqslant \boldsymbol{0}. && \textit{(feasibility)} && \text{(3.10e)}
\end{aligned}$$

Proof. Proofs may be found in any standard textbook on numerical optimization (see, e.g., Nocedal and Wright [43]). □

In a local minimizer, by virtue of condition (3.10a), the objective gradient $\nabla_{\boldsymbol{x}} f$ is a linear combination of the constraint gradients $\nabla_{\boldsymbol{x}} \boldsymbol{g}$ and $\nabla_{\boldsymbol{x}} \boldsymbol{h}$. Lagrange multipliers of active inequality constraints must be non-negative (3.10b), while the complementarity condition (3.10c) assigns zero multipliers to inactive constraints.

Definition 3.10. ***Karush-Kuhn-Tucker Point***
*A triplet $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ satisfying the Karush-Kuhn-Tucker first-order necessary conditions of Theorem 3.9 is called a **Karush-Kuhn-Tucker point** (short **KKT point**).*

Definition 3.11. ***Stationary Point***
*A regular point $\boldsymbol{x} \in \mathcal{F}$ satisfying the Karush-Kuhn-Tucker first-order necessary conditions of Theorem 3.9 (i.e. a regular KKT point) is referred to as a **stationary point** $\boldsymbol{x}^*$ of problem (3.1a).*

[1] Joseph Louis Lagrange (1763–1813)

3.1.4 A Sufficient Condition for Optimality

The KKT point defined in the previous section is a possible candidate for a local minimum of the nonlinear problem (3.1a). Many NLP methods thus attempt to converge to a KKT point, that mandates additional validation after convergence. In order to verify that the resulting point actually is a minimizer, further conditions like the second-order sufficient condition presented below need to be satisfied.

Definition 3.12. ***Strictly Active Constraints***
An active constraint h_i is said to be ***strictly active*** *if the associated Lagrange multiplier μ_i is positive. The set of strictly active constraints is denoted by*

$$\tilde{\mathcal{A}}(\boldsymbol{x}) := \big\{ i \mid h_i(\boldsymbol{x}) = 0 \ \wedge \ \mu_i > 0 \big\} \subseteq \mathcal{A}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{F}. \tag{3.11}$$

Theorem 3.13. ***Second-Order Sufficient Condition***
A sufficient condition for the point $\boldsymbol{x}^*$ to be a local minimizer of problem (3.1a) is that there exists a KKT point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ such that for every non-zero direction $\boldsymbol{p} \in \mathbb{R}^{n_x}$ satisfying

$$\begin{aligned} \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}^*)\, \boldsymbol{p} &= \boldsymbol{0}, && \text{(3.12a)} \\ \nabla_{\boldsymbol{x}} h_i(\boldsymbol{x}^*)\, \boldsymbol{p} &= 0, \quad i \in \tilde{\mathcal{A}}(\boldsymbol{x}^*), && \text{(3.12b)} \\ \nabla_{\boldsymbol{x}} h_i(\boldsymbol{x}^*)\, \boldsymbol{p} &\geqslant 0, \quad i \in \mathcal{A}(\boldsymbol{x}^*) \setminus \tilde{\mathcal{A}}(\boldsymbol{x}^*), && \text{(3.12c)} \end{aligned}$$

it follows that

$$\boldsymbol{p}^{\mathsf{T}} \nabla^2_{\boldsymbol{x}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\, \boldsymbol{p} > 0. \tag{3.13}$$

We refer to Leineweber [38] for a detailed motivation of this condition, as well as a slightly stricter variant guaranteeing uniqueness of the Lagrange multipliers.

3.1.5 Stability of a Solution

Definition 3.14. ***Strict Complementarity***
We say that ***strict complementarity*** *holds in a Karush-Kuhn-Tucker point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ iff any active inequality constraint is strictly active,*

$$\mathcal{A}(\boldsymbol{x}^*) = \tilde{\mathcal{A}}(\boldsymbol{x}^*), \tag{3.14}$$

or equally

$$\forall i \in \{1, \ldots, n_h\} : \ \big(h_i(\boldsymbol{x}^*) = 0 \ \wedge \ \mu_i > 0\big) \quad \vee \quad \big(h_i(\boldsymbol{x}^*) > 0 \ \wedge \ \mu_i = 0\big). \tag{3.15}$$

If strict complementarity holds (3.14) in a minimum $\boldsymbol{x}^*$, it can be shown that the solution is stable against small deviations of the problem. This is vitally important for the validity of numerically computed minima, since small deviations due to precision and round-off errors acting as perturbations to the original nonlinear problem are inevitable. In this section we mention the main stability theorem, a proof of which can be found in Bock [6].

Theorem 3.15. ***Stability of the Solution***
Consider the disturbed problem

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathcal{D}} \quad & \tilde{f}(\boldsymbol{x};\, \varepsilon) && \text{(3.16a)} \\ \text{s.t.} \quad & \tilde{\boldsymbol{g}}(\boldsymbol{x};\, \varepsilon) = \boldsymbol{0}, && \text{(3.16b)} \\ & \tilde{\boldsymbol{h}}(\boldsymbol{x};\, \varepsilon) \geqslant \boldsymbol{0}, && \text{(3.16c)} \end{aligned}$$

the argument $\varepsilon > 0$ being a disturbation parameter. For $\varepsilon = 0$ the disturbed problem shall coincide with the original problem (3.1a).

If for a Karush-Kuhn-Tucker point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$

1. *the second-order sufficient condition (3.12) is satisfied,*
2. *constraint qualification holds, e.g., the linear independence constraint qualification condition (3.7) is satisfied,*
3. *strict complementarity (3.14) holds,*

then there exists an open ball $\mathcal{V} \subset \mathbb{R}$ of 0 and an open ball $\mathcal{W} \subset \mathbb{R}^{n_x} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h}$ of the KKT point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ as well as a $\mathscr{C}^1$ mapping φ from $\mathcal{V}$ to $\mathcal{W}$ such that

1. *$\varphi(\varepsilon) = \big(\boldsymbol{x}(\varepsilon), \boldsymbol{\lambda}(\varepsilon), \boldsymbol{\mu}(\varepsilon)\big)$ is the only KKT point of the disturbed system within $\mathcal{W}$,*
2. *the disturbed system has a strict local minimum in $\boldsymbol{x}(\varepsilon)$,*
3. *the set of active inequality constraints $\mathcal{A}\big(\boldsymbol{x}(\varepsilon)\big)$ is fixed on the whole of $\mathcal{V}$.*

Proof. See Bock [6]. □

3.2 Sequential Quadratic Programming

The basic principle of sequential methods for the solution of the nonlinear problem (3.1a) is that of replacing it by a sequence of sub-problems related to the original one. When locating extremal points, the simplest imaginable sub-problem would be a local quadratic representation of the original nonlinear problem. Linear representations are inappropriate since their boundedness cannot be guaranteed.
From the solution of the subproblems one obtains a step towards the minimizer of the local subproblem. Here it is important for the step to remain in a region of the search space where the local subproblem can be trusted to satisfactorily approximate the original nonlinear problem. Ensuring this is the domain of trust-region and line-search methods (cf. Nocedal and Wright [43]). If, in general, it is possible to ensure that significant progress can be made in every step, one may obtain a globally convergent method that eventually ends up in a local solution of the original problem regardless of the choice of the initializer $\boldsymbol{x}^{(0)}$.

3.2.1 Fundamental Idea

SQP methods for the solution of (3.1a) are based on the optimality criteria presented in the first section of this chapter. From Theorem 3.9 it becomes clear that the optimality of a point $\boldsymbol{x}$ in the search space $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ depends on the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ rather than on the NLP's objective function f itself. It is therefore straightforward to obtain a local sub-problem P from a quadratic approximation of the Lagrangian, for example by collecting the first three terms of the Taylor series of L in the point $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$.

$$P(\boldsymbol{x} + \boldsymbol{\Delta x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = L + \nabla_{\boldsymbol{x}} L \boldsymbol{\Delta x} + \tfrac{1}{2} \boldsymbol{\Delta x}^{\mathsf{T}} \nabla^2_{\boldsymbol{x}} L \boldsymbol{\Delta x}. \tag{3.17}$$

To obtain the step $\boldsymbol{\Delta x}$ we minimize the model P. Omitting the constant term L, and observing that $\nabla_{\boldsymbol{x}} L \boldsymbol{\Delta x} = \nabla_{\boldsymbol{x}} f \boldsymbol{\Delta x}$ for strict complementarity and directions $\boldsymbol{\Delta x}$ in the null-space of the equality and active inequality constraint Jacobian, we arrive at the simpler model

$$P(\boldsymbol{x} + \boldsymbol{\Delta x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla_{\boldsymbol{x}} f \boldsymbol{\Delta x} + \tfrac{1}{2} \boldsymbol{\Delta x}^{\mathsf{T}} \nabla^2_{\boldsymbol{x}} L \boldsymbol{\Delta x}. \tag{3.18}$$

In general, a stationary point $\boldsymbol{x}^*$ of the Lagrangian is only a minimizer on that null-space, as seen from the second-order sufficient condition of Theorem 3.13. This suggests to impose linear constraints on the quadratic sub-problem (cf. Leineweber [38]). Together with the derived objective, we obtain a linearly constrained quadratic program (LCQP) from the quadratic approximation of problem (3.1a) around the current iterate.

Definition 3.16. *Linearly Constrained Quadratic Sub-Problem*

$$\min_{\boldsymbol{\Delta x}} \quad \nabla_{\boldsymbol{x}} f(\boldsymbol{x})\boldsymbol{\Delta x} + \tfrac{1}{2}\boldsymbol{\Delta x}^{\mathsf{T}}\nabla^2_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu})\boldsymbol{\Delta x} \tag{3.19a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x})\boldsymbol{\Delta x} = \boldsymbol{0}, \tag{3.19b}$$

$$\boldsymbol{h}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\Delta x} \geqslant \boldsymbol{0}. \tag{3.19c}$$

Recall now the first-order necessary Karush-Kuhn-Tucker conditions from Theorem 3.9. Applying them to the LCQP yields

$$\begin{aligned}
\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) + \tfrac{1}{2}\nabla^2_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu})\boldsymbol{\Delta x} - \boldsymbol{\Delta\lambda}^{\mathsf{T}}\nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\Delta\mu}^{\mathsf{T}}\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{0}, && (3.20a)\\
\boldsymbol{\Delta\mu} &\geqslant \boldsymbol{0}, && (3.20b)\\
\boldsymbol{\Delta\mu}^{\mathsf{T}}\big(\boldsymbol{h}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\Delta x}\big) &= 0, && (3.20c)\\
\boldsymbol{g}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x})\boldsymbol{\Delta x} &= \boldsymbol{0}, && (3.20d)\\
\boldsymbol{h}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\Delta x} &\geqslant \boldsymbol{0}. && (3.20e)
\end{aligned}$$

Implying convergence of the sequential algorithm we look at the limiting case $\boldsymbol{\Delta x} \to \boldsymbol{0}$. If $\boldsymbol{\Delta x}^* = \boldsymbol{0}$ is a solution of the LCQP, then there exist Lagrange multipliers $\boldsymbol{\Delta\lambda}^*$ and $\boldsymbol{\Delta\mu}^*$ such that $(\boldsymbol{\Delta x}^*, \boldsymbol{\Delta\lambda}^*, \boldsymbol{\Delta\mu}^*)$ is a KKT point of the LCQP. It immediately follows that $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ must be a KKT point of the original problem, and that the Lagrange multipliers match. From Eq. (3.20a) it is clear that we would not see this benefit if we had not replaced the Lagrangian gradient by the objective gradient before.

3.2.2 A Basic SQP Method

The described idea gives rise to the following very basic SQP algorithm 3.1.

Algorithm 3.1 A basic sequential quadratic programming (SQP) method.

Input: Objective and constraint functions f, $\boldsymbol{g}$, and $\boldsymbol{h}$, iterate $\big(\boldsymbol{x}^{(0)}, \boldsymbol{\lambda}^{(0)}, \boldsymbol{\mu}^{(0)}\big)$, tolerance TOL.
Output: Approximation $\boldsymbol{x}^*$ to a local minimum of the NLP, satisfying the tolerance TOL.

Set $k := 0$.

repeat

Evaluate the functions

$$f(\boldsymbol{x}^{(k)}), \quad \boldsymbol{g}(\boldsymbol{x}^{(k)}), \quad \boldsymbol{h}(\boldsymbol{x}^{(k)}),$$

and their gradients

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)}), \quad \nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x}^{(k)}), \quad \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}^{(k)}).$$

Evaluate the Lagrangian's Hessian $\nabla^2_{\boldsymbol{x}} L$, or a suitable approximation (cf. Section 3.2.3).

Solve the linearly constrained quadratic program (3.19a) at the current iterate $\big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\mu}^{(k)}\big)$, to obtain a step $\boldsymbol{\Delta x}$ and Lagrangian multipliers $\boldsymbol{\Delta\lambda}$, $\boldsymbol{\Delta\mu}$.

Determine a step length $\alpha^{(k)} \in (0, 1] \subset \mathbb{R}$ using, e.g., a line search method (cf. Section 3.2.3).

Perform the step

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &:= \boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{\Delta x},\\
\boldsymbol{\lambda}^{(k+1)} &:= \boldsymbol{\lambda}^{(k)} + \alpha^{(k)}\big(\boldsymbol{\Delta\lambda} - \boldsymbol{\lambda}^{(k)}\big),\\
\boldsymbol{\mu}^{(k+1)} &:= \boldsymbol{\mu}^{(k)} + \alpha^{(k)}\big(\boldsymbol{\Delta\mu} - \boldsymbol{\mu}^{(k)}\big).
\end{aligned}$$

Set $k := k + 1$

Evaluate a given convergence criterion using the tolerance TOL (cf. Section 3.2.3).

until the convergence criterion is met.

Terminate with $\boldsymbol{x}^* := \boldsymbol{x}^{(k)}$ as the solution.

The proof of locally quadratic convergence is by analogy to Newton's method, and can be found in, e.g., Nocedal and Wright [43] or Leineweber [38].

3.2.3 Details and Improvements

Obviously the above algorithm leaves much room for details to be specified, and improvements to be made. We refer to Nocedal and Wright [43] for a thorough discussion that is beyond the scope of this thesis, and just briefly mention some crucial points here.

Solving the Quadratic Subproblem

We haven't yet made any comment about how to actually solve the linearly constrained quadratic subproblem (LCQP) to obtain the step and new Lagrangian multipliers. Various quadratic program (QP) solving methods for general inequality-constrained QPs may be found in the literature. Amongst them are primal active-set methods (Gill and Murray [23]), dual active-set methods (Goldfarb and Idnani [24]), and interior-point methods. Specialized methods benefiting from restrictions on the class of treated QPs, such as unconstrained, equality-constrained, or convex QPs, are available (Nocedal and Wright [43]). We refer to Leineweber [38] for details on the QP solvers available in the optimal control software package *MUSCOD-II*.

Determining a Step Length

Line search strategies for the decision on a suitable step length $\alpha^{(k)}$ are, e.g., based on the Armijo-Wolfe or Armijo-Goldstein (3.21) criteria,

$$f^{(k)} + (1-c)\alpha^{(k)}\nabla_{\boldsymbol{x}}^{\mathsf{T}} f^{(k)} \boldsymbol{\Delta x}^{(k)} \leqslant f(\boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{\Delta x}^{(k)}) \leqslant f^{(k)} + c\alpha^{(k)}\nabla_{\boldsymbol{x}}^{\mathsf{T}} f^{(k)} \boldsymbol{\Delta x}^{(k)}, \quad (3.21)$$

where $f^{(k)} := f(\boldsymbol{x}^{(k)})$ and $c \in (0, \frac{1}{2})$. Satisfying these conditions ensures that the resulting step $\alpha^{(k)}\boldsymbol{\Delta x}^{(k)}$ leads to sufficient decrease of the objective function, while preventing the step length from getting too small. Many related approaches may be found in the literature. The condition ensuring sufficient progress may be dropped in favor of a back-tracking line search method. Curvature information from second-order derivatives may be evaluated to avoid convergence to non-minimizing stationary points. Interpolation approaches may be used to reduce the computational cost of evaluating the objective f and its gradient. In addition, derivative-free line search methods exist. Nocedal and Wright [43] discuss these points to some detail, and present algorithms.
Line search methods may be employed to globalize the convergence of an otherwise locally convergent method. Han [28] proved global convergence of the variable metric approach (Han [27], Leineweber [38]) in conjunction with a special ℓ_1 penalty function.

Testing for Convergence

Ideally, an iterate $\boldsymbol{x}^{(k)}$ constituting a solution of problem (3.1a) will satisfy the necessary Karush-Kuhn-Tucker conditions (3.9). Namely, the iterate will be feasible, and the objective gradient will be zero.
A suitable termination criterion for the above algorithm pays respect to these properties. For example, one might test for a norm $\left|\left|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(k)})\right|\right|$ of the objective's gradient, and a norm of the infeasibility of the current iterate, e.g.,

$$\sum_{i=1}^{n_g} \left|\lambda_i^{(k)} g_i(\boldsymbol{x}^{(k)})\right| + \sum_{i=1}^{n_h} \left|\mu_i^{(k)} h_i(\boldsymbol{x}^{(k)})\right|,$$

to fall beyond a prescribed tolerance in order to consider an iterate $\boldsymbol{x}^{(k)}$ to be an acceptable approximate of the solution.

Shortcomings

In this original version developed by Wilson [60] the presented method suffers from several shortcomings (Leineweber [38]):

1. The exact Hessian of (3.1a) may become indefinite, which immensely complicates solving the then non-convex quadratic sub-problems.
2. The exact Hessian turns out to be largely irrelevant for the computation of a good search direction if the Lagrange multipliers are far off those of the real solution.

Most of these shortcomings may be overcome by the variable metric approach (see Leineweber [38] for details). A proof of locally q-superlinear convergence was established by Han [27]. A first successful implementation of an SQP method based on these ideas was presented by Powell [49].

3.3 The Constrained Gauß-Newton Method

The Gauß-Newton[2] method for the solution of unconstrained least-squares problems is classical, and dates back to the work of J.C.F. Gauß (*Theoria motus corporum coelestium*, 1809). First successful implementations date back to Levenberg [39], later rediscovered by Marquardt [41]. In this section we briefly present an efficient algorithm for the solution of the discretized parameter estimation problem of Chapter 5, namely a generalization of the Gauß-Newton method to constrained least-squares problems that is due to Bock [6] and Schlöder [52] .

3.3.1 Fundamental Idea

Given a vectorial objective $\boldsymbol{f}(\boldsymbol{x})$ mapping unknowns $\boldsymbol{x} \in \mathbb{R}^{n_x}$ to residuals $\boldsymbol{f} \in \mathbb{R}^{n_f}$ we consider the general constrained nonlinear problem of least squares (3.22a). Observe that the discretized parameter estimation problem derived in Chapter 5 fits into this framework.

Definition 3.17. ***Nonlinear Constrained Least-Squares Problem***

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathcal{D}} \quad & \tfrac{1}{2}\,||\boldsymbol{f}(\boldsymbol{x})||_2^2 && \text{(3.22a)}\\ \text{s.t.} \quad & \boldsymbol{g}(x) = 0, && \text{(3.22b)}\\ & \boldsymbol{h}(x) \geqslant 0. && \text{(3.22c)} \end{aligned}$$

We assume f, g, and h to be $\mathscr{C}^2$ functions over the domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$.

Applying the Karush-Kuhn-Tucker first-order necessary conditions of Theorem 3.9 to this problem we find that a stationary point $\boldsymbol{x}^* \in \mathcal{F}$ satisfies

$$\begin{aligned} \nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\lambda}^{\mathsf{T}}\nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\mu}^{\mathsf{T}}\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{0},\\ \boldsymbol{\mu}^{*\mathsf{T}}\boldsymbol{h}(\boldsymbol{x}^*) &= 0. \end{aligned}$$

Consider on the other hand the linearized constrained least-squares problem that results from the linearization of problem (3.22a) around the point $\boldsymbol{x} \in \mathcal{D}$.

Definition 3.18. ***Linearized Constrained Least-Squares Problem***

$$\begin{aligned} \min_{\boldsymbol{\Delta x}} \quad & ||\boldsymbol{f}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x})\boldsymbol{\Delta x}||_2^2 && \text{(3.23a)}\\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{g}(\boldsymbol{x})\boldsymbol{\Delta x} = \boldsymbol{0}, && \text{(3.23b)}\\ & \boldsymbol{h}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\Delta x} \geqslant \boldsymbol{0}, && \text{(3.23c)} \end{aligned}$$

[2] Johann Carl Friedrich Gauß (1777–1855), Isaac Newton (1643–1727)

Stationary points of this linearized problem satisfy

$$\begin{aligned} \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}} \big(\boldsymbol{f}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \boldsymbol{\Delta x} \big) - \boldsymbol{\lambda}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\mu}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{0}, \\ \boldsymbol{\mu}^{*\mathsf{T}} \big(\boldsymbol{h}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{h}(\boldsymbol{x}) \boldsymbol{\Delta x} \big) &= 0. \end{aligned}$$

and it is easily seen that a stationary point $(\boldsymbol{0}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ of the linearized system is a stationary point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ of the original system. This observation closely resembles the one made to derive the sequential quadratic programming (SQP) framework. Hence, with minor modifications in place, the basic SQP algorithm presented earlier in this chapter may also serve to solve nonlinear least-squares problems. From the specialized objective function the Hessian approximate $\nabla_{\boldsymbol{x}} \boldsymbol{f}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{f}$ is derived to replace the Lagrangian's Hessian $\nabla^2_{\boldsymbol{x}} L$, and the constrained linear least-squares problem takes the place of the quadratic sub-problem.

3.3.2 Solving the Linearized System

It remains to be discussed how to solve the linearized constrained problem (3.23a) in order to obtain the step $\boldsymbol{\Delta x}$. From Theorem 3.15 we find that the convergence behavior of the constrained Gauß-Newton (CGN) method actually depends on the equality constraints and the active inequality constraints only. The latter may thus be viewed as additional equality constraints when discussing the local convergence behavior of this method. The domain $\mathcal{D}$ needs to be restricted to $\mathcal{D} \cap \mathcal{W}$ (see the stability Theorem 3.15) to guarantee that no changes in the active set occur.

Definition 3.19. ***Linearized Equality-Constrained Least-Squares Problem***

$$\min_{\boldsymbol{x} \in \mathcal{D} \cap \mathcal{W}} \; ||\boldsymbol{f}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \boldsymbol{\Delta x}||_2^2 \tag{3.24a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}) \boldsymbol{\Delta x} = \boldsymbol{0}. \tag{3.24b}$$

Theorem 3.20. ***Existence of a Generalized Inverse*** *(Bock [6])*
Consider the linearized equality-constrained least-squares problem (3.24a). We assume that constraint qualification holds (e.g., in a regular point), and that the Hessian approximate

$$\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \tag{3.25}$$

is positive definite on the null-space of $\nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x})$. Then the solution of (3.24a) may be obtained by way of a generalized inverse:

1. *For any value $\begin{bmatrix} \boldsymbol{f}^{\mathsf{T}} & \boldsymbol{g}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{n_f + n_g}$ there exists exactly one KKT-point $(\boldsymbol{\Delta x}^*, \boldsymbol{\Delta \lambda}^*)$ of problem (3.24a). Further, $\boldsymbol{\Delta x}^*$ is a strict local minimum.*

2. *There exists a linear mapping*

$$\boldsymbol{J}^+ : \mathbb{R}^{n_f + n_g} \longrightarrow \mathbb{R}^{n_x} \tag{3.26}$$

   such that

$$\boldsymbol{\Delta x} := -\boldsymbol{J}^+ \boldsymbol{f} \tag{3.27}$$

   is the solution of (3.24a) for any value $\begin{bmatrix} \boldsymbol{f}^{\mathsf{T}} & \boldsymbol{g}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$.

3. *The solution operator J^+ is a generalized inverse, satisfying the defining relationship*

$$\boldsymbol{J}^+ \boldsymbol{J} \boldsymbol{J}^+ = \boldsymbol{J}^+. \tag{3.28}$$

The representability of the linearized problem's solutions by way of the operator $\boldsymbol{J}^+$ allows for a generalized treatment of nonlinear problems as well as constrained and unconstrained least-squares problems. The operator $\boldsymbol{J}^+$ also holds valuable statistical information, as described in Section 5.2.

A proof of locally linear convergence as well as an extensive description of effective globalization strategies can be found in Bock [6].

3.3.3 The Levenberg-Marquardt Method

The option to confine the Gauß-Newton step $\boldsymbol{\Delta x}$ to a spherical trust-region is given by the Levenberg-Marquardt modification to the CGN method (cf. Levenberg [39] and Marquardt [41]). It avoids the weak behavior of the Gauß-Newton method on singular Jacobians $\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})$ by adding a regularizing term $\lambda \boldsymbol{I}$ to the Hessian approximation.

This modification may be viewed as a trust-region approach by virtue of the following lemma.

Lemma 3.21. *The vector $\boldsymbol{\Delta x}$ is a solution for the trust-region sub-problem*

$$\min_{\boldsymbol{\Delta x}} \quad ||\boldsymbol{f}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) \boldsymbol{\Delta x}||_2^2 \tag{3.29a}$$

$$\text{s.t.} \quad ||\boldsymbol{\Delta x}||_2 \leqslant \Delta, \tag{3.29b}$$

for some $\Delta > 0$ iff there is a scalar $\lambda \geqslant 0$ such that

$$\left(\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}} \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) + \lambda \boldsymbol{I}\right) \boldsymbol{\Delta x} = -\nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}), \tag{3.30a}$$

$$\lambda\big(\Delta - ||\boldsymbol{\Delta x}||_2\big) = 0. \tag{3.30b}$$

$$\tag{3.30c}$$

Proof. See Nocedal and Wright [43]. □

Both Levenberg [39] and Marquardt [41] did not view their method as a trust-region approach but directly selected the multiplier λ according to some heuristic, similar to trust-region heuristics for the adjustment of the trust-region radius Δ. We refer to Nocedal and Wright [43] for a discussion on how to find the scalar λ if given a trust-region radius Δ determined by such a heuristic.

Ellipsoidal trust-regions, with principal axes lengths matching the scale of the corresponding variables reduce effects generated by poorly scaled least-squares problems. These can be generated by replacing the identity $\boldsymbol{I}$ with a diagonal matrix $\boldsymbol{D}$ with positive entries. Nocedal and Wright [43] suggest choosing the diagonal elements of the respective Hessian approximate.

Chapter 4

A Continuous Runge-Kutta Method Handling Implicit Switches

In this chapter we describe a continuous extension to the explicit Runge-Kutta method for the integration of non-stiff ODE systems. The method will support an event detection facility that allows for the treatment of implicitly defined switches of the model functions, i.e., jumps in the system's differential states or discontinuities of the right-hand side. An important advantage of this approach over the realization of explicit switches using *model stages* in *MUSCOD-II* [12, 38] is the fact that neither the number of actually occurring switches, nor their order in time need to be known in advance.
We saw in Chapter 1 that the ability to handle discontinuities exactly is crucial for the proper implementation of the powertrain model. In addition, it gives us considerable additional freedom in the choice of optimal control scenarios, as will be discussed in Chapter 6.
Named *RKFSWT* (**R**unge-**K**utta-**F**ehlberg with **swit**ches), an implementation of the described method is now available in the optimal control software package *MUSCOD-II*.

4.1 Runge-Kutta Methods

We start this chapter with a brief presentation of Runge-Kutta schemes as specialized one-step methods for the numerical solution of a non-stiff initial-value problem (IVP) of the form

Definition 4.1. ***Initial-Value Problem***

$$\frac{d\boldsymbol{y}}{dt}(t) = \boldsymbol{f}\big(t, \boldsymbol{y}(t)\big), \quad t \in \mathcal{T} \tag{4.1a}$$

$$\boldsymbol{y}(t_0) = \boldsymbol{y_0}, \tag{4.1b}$$

on the time horizon $\mathcal{T} := [t_0, t_\mathrm{f}] \subset \mathbb{R}$. We assume $\boldsymbol{f}$ to be Lipschitz continuous on $\mathcal{T} \times \mathbb{R}^{n_y}$, thus ensuring existence, uniqueness, and continuous differentiability of the IVP's solution $\boldsymbol{y}(t)$ on $\mathcal{T}$.

4.1.1 Definition of Runge-Kutta Methods

Definition 4.2. ***One-Step Method***
*A **one-step method** for the solution of the initial-value problem (4.1) is given by a function $\boldsymbol{\Phi}$ depending on $(t, h,\ \boldsymbol{y},\ \boldsymbol{f})$. Starting in $t^{(0)} := t_0$ with the given initial value $\boldsymbol{\eta}^{(0)} := \boldsymbol{y_0}$ and using the iteration scheme*

$$\boldsymbol{\eta}^{(k+1)} := \boldsymbol{\eta}^{(k)} + h^{(k)}\ \boldsymbol{\Phi}\left(t^{(k)}, h^{(k)},\ \boldsymbol{\eta}^{(k)},\ \boldsymbol{f}\right), \tag{4.2a}$$

$$t^{(k+1)} := t^{(k)} + h^{(k)}, \tag{4.2b}$$

the function $\boldsymbol{\Phi}$ creates a sequence of approximations $\boldsymbol{\eta}^{(k)}$ to the exact solutions $\boldsymbol{y}(t^{(k)})$ of (4.1) on the discretization grid defined by the step lengths $h^{(k)}$.

Within this framework, Runge-Kutta methods, first described by Runge [51] and Kutta [37], are one-step methods with a generating function $\boldsymbol{\Phi}$ of the following special structure.

Definition 4.3. ***Runge-Kutta Method***
*A **Runge-Kutta method** with $s \in \mathbb{N}$ stages is a one-step method with a generating function $\boldsymbol{\Phi}$ depending on $(t, h, \boldsymbol{y}, \boldsymbol{f})$ which is defined by coefficients $\boldsymbol{c} \in \mathbb{R}^s$ and $\boldsymbol{k}_i \in \mathbb{R}^s$, $i = 1, \ldots, s$ as follows.*

$$\boldsymbol{\Phi}(t, h, \boldsymbol{y}, \boldsymbol{f}) := \sum_{i=1}^{s} c_i \boldsymbol{k}_i, \tag{4.3}$$

where the vectors $\boldsymbol{k}_i$ are implicitly defined by

$$\boldsymbol{k}_i = \boldsymbol{f}\Big(t + \alpha_i h, \boldsymbol{y} + h \sum_{j=1}^{s} B_{ij} \boldsymbol{k}_j\Big), \tag{4.4}$$

using suitable coefficients $\boldsymbol{\alpha} \in \mathbb{R}^s$ and $\boldsymbol{B} \in \mathcal{M}(s \times s, \mathbb{R})$.

For general matrices $\boldsymbol{B}$, Eq. (4.4) yields an implicit definition of the vectors $\boldsymbol{k}_i$, whose computations thus require the solution of a system of nonlinear equations. A Runge-Kutta method is said to be of the *explicit* type if the rows of $\boldsymbol{B}$ can be reordered to form a lower triangular matrix of rank $s - 1$. This allows for iterative direct computation of the values $\boldsymbol{k}_i$. The method's coefficients are commonly given in the form of a so-called Butcher tableau (cf. Butcher [10]). Fig. 4.1 shows Butcher tableaux of two embedded Runge-Kutta-Fehlberg methods.

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{3}{8} & \frac{3}{8} & & & & \\
\frac{9}{16} & 0 & \frac{9}{16} & & & \\
\frac{25}{32} & -\frac{125}{672} & \frac{325}{336} & 0 & & \\
1 & \frac{371}{891} & -\frac{200}{297} & \frac{1120}{891} & 0 & \\
\hline
 & \frac{37}{225} & \frac{44}{117} & 0 & \frac{448}{975} & \\
 & \frac{25}{162} & \frac{32}{135} & \frac{256}{567} & 0 & \frac{11}{70}
\end{array}$$

(a) Nørsett's 4^{th}-order method RKN3(4).

$$\begin{array}{c|cccccc}
0 & & & & & & \\
\frac{1}{4} & \frac{1}{4} & & & & & \\
\frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\
\frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\
1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\
\frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\
\hline
 & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & \\
 & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55}
\end{array}$$

(b) Fehlberg's popular method of 5^{th} order, RKF4(5).

Fig. 4.1: Butcher tableaux of two embedded Runge-Kutta methods of Fehlberg type.

4.2 Convergence and Error Control

In this section we look at the approximation errors incurred by Runge-Kutta methods, and present an overview over approaches for controlling these approximation errors by varying the method's step size $h^{(k)}$.

4.2.1 Errors

Definition 4.4. ***Errors of a One-Step Method***
*Let $\boldsymbol{\Phi}$ be a one-step method, and let $\boldsymbol{y}(t)$ denote the exact solution of an IVP on $\mathcal{T} \subset \mathbb{R}$. Then the local discretization error or **consistency error $\boldsymbol{\tau}$** of step (k) is defined as*

$$\boldsymbol{\tau}^{(k)}\left(t^{(k)}, h^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{f}\right) := \frac{\boldsymbol{y}(t^{(k+1)}) - \boldsymbol{y}(t^{(k)})}{h^{(k)}} - \boldsymbol{\Phi}\left(t^{(k)}, h^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{f}\right) \tag{4.5}$$

*and the **local error $\boldsymbol{\sigma}$** is defined as*

$$\boldsymbol{\sigma}^{(k)}\left(t^{(k)}, h^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{f}\right) := h^{(k)}\, \boldsymbol{\tau}^{(k)}\left(t^{(k)}, h^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{f}\right). \tag{4.6}$$

*The **global error** $\boldsymbol{\varepsilon}$ is*

$$\boldsymbol{\varepsilon}^{(k)}\left(t^{(k)}, h^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{f}\right) := \boldsymbol{\eta}^{(k+1)} - \boldsymbol{y}(t^{(k+1)}). \tag{4.7}$$

The consistency error $\boldsymbol{\tau}^{(k)}$ measures the deviation of the approximated secant from the exact one, while the local error $\boldsymbol{\sigma}^{(k)}$ represents the accumulated secant error over the single step (k). The global error $\boldsymbol{\varepsilon}^{(k)}$ also accumulates local errors from all previous steps $(0), \ldots, (k-1)$.

4.2.2 Establishing Convergence

Definition 4.5. ***Consistency of a One-Step Method***
*A one-step method $\boldsymbol{\Phi}$ is called a **consistent** method iff*

$$\lim_{h\to 0} \overline{\tau}(h) = \mathbf{0}, \qquad \overline{\tau}(h) := \sup_{t\in\mathcal{T}} ||\boldsymbol{\tau}\left(t, h, \boldsymbol{y}, \boldsymbol{f}\right)||\,. \tag{4.8}$$

*A consistent one-step method is said to have a **consistency order** of $p \geqslant 1$ iff*

$$\overline{\tau}(h) = \mathcal{O}(h^p), \quad \textit{or equivalently} \quad \overline{\sigma}(h) := \sup_{t\in\mathcal{T}} ||\boldsymbol{\sigma}\left(t, h, \boldsymbol{y}, \boldsymbol{f}\right)|| = \mathcal{O}(h^{p+1}).$$

Definition 4.6. ***Convergence of a One-Step Method***
*A one-step method $\boldsymbol{\Phi}$ is **convergent** iff*

$$\lim_{h\to 0} \overline{\varepsilon}(h) = 0, \qquad \overline{\varepsilon}(h) := \sup_{t\in\mathcal{T}} ||\boldsymbol{\varepsilon}\left(t, h, \boldsymbol{y}, \boldsymbol{f}\right)|| \tag{4.9}$$

*and if so, it is said to have a **convergence order** of $p \geqslant 1$ iff*

$$\overline{\varepsilon}(h) = \mathcal{O}(h^p).$$

Definition 4.7. ***Stability of a One-Step Method***
*A one-step method $\boldsymbol{\Phi}$ is **stable** iff there exists $0 \leqslant \kappa < \infty$ such that*

$$\overline{\varepsilon}(h) \leqslant \kappa\, \overline{\tau}(h).$$

The convergence order of a one-step method often is simply referred to as "the order" of the method. Butcher [10] showed that there is no upper limit to the order of constructible Runge-Kutta methods. The number of stages required to obtain such methods, however, grows much faster than the order itself, cf. Tab. 4.1 on page 46.

Concerning the behavior of Runge-Kutta methods in this context, proofs of the following claims that establish the stability of Runge-Kutta methods under mild assumptions can be found in all standard textbooks on numerical treatment of ODEs, e.g., Stoer and Bulirsch [56], Press et al. [50].

Lemma 4.8. ***Lipschitz Continuity of a Runge-Kutta Method***
If the right-hand side $\boldsymbol{f}$ of the IVP (4.1) is Lipschitz continuous, then every Runge-Kutta method $\boldsymbol{\Phi}$ (4.3) applied to (4.1) is also Lipschitz continuous.

Lemma 4.9. ***Consistency of a Runge-Kutta Method***
A Runge-Kutta method is consistent iff $\sum_{i=1}^{s} c_i = 1$.

Proposition 4.10. ***Convergence of a One-Step Method***
A Lipschitz-continuous one-step method $\boldsymbol{\Phi}$ is stable. It is convergent if it is also consistent, and its convergence order equals its consistency order.

4.2.3 Variable-Step Methods

All modern methods vary the size of the integrator steps $h^{(k)}$ in order to satisfy a certain prescribed error tolerance TOL. Given an asymptotically correct estimate $\mathbf{est}$ of the local error $\boldsymbol{\sigma}^{(k)}$ (4.6)

$$\mathbf{est} := \boldsymbol{y}(t^{(k)} + h^{(k)}) - \boldsymbol{\eta}^{(k+1)} + \mathcal{O}\left((h^{(k)})^{p+1}\right), \tag{4.10}$$

p being the order of the method, we desire to choose $h^{(k)}$ such that the error estimate $||\mathbf{est}||$ stays beyond a prescribed tolerance TOL. It is important to chose a suitable norm here, e.g., an ℓ_∞ norm of the scaled state vector

$$||\mathbf{est}||_{S,\infty} := \max_{i=1,\ldots,n_y} \left|\frac{\mathrm{est}_i}{s_i}\right|.$$

Obviously the quality of the error estimate is of vital importance to the performance of the method. Careless overestimation of the local error will lead to unnecessarily small step sizes wasting computation time, while underestimating the local error hurts the precision and validity of the computed approximation to the solution of the IVP.
In the literature one may find several estimates of the local error, and we give a brief overview along the lines of Enright et al. [20].

1. Local error per unit step;
   (Fehlberg [21], *MUSCOD-II* [38])

   This is the classical concept devised by Fehlberg. Using two related Runge-Kutta methods $\boldsymbol{\Phi}^{[p]}$ and $\boldsymbol{\Phi}^{[p+1]}$ of orders p and $p+1$ respectively, we compute two steps to the new approximation

   $$\boldsymbol{\Delta}^{[p]} := h^{(k)} \cdot \boldsymbol{\Phi}^{[p]}(t^{(k)}, h^{(k)}, \boldsymbol{\eta}^{(k)}, \boldsymbol{f}), \tag{4.11a}$$
   $$\boldsymbol{\Delta}^{[p+1]} := h^{(k)} \cdot \boldsymbol{\Phi}^{[p+1]}(t^{(k)}, h^{(k)}, \boldsymbol{\eta}^{(k)}, \boldsymbol{f}), \tag{4.11b}$$

   and advance with the lower-order step. Using the difference to the higher order step as an estimate that is asymptotically correct for the lower-order method,

   $$\mathbf{est} := \boldsymbol{\Delta}^{[p+1]} - \boldsymbol{\Delta}^{[p]}, \tag{4.12}$$

   we choose $h^{(k)}$ to satisfy

   $$||\mathbf{est}|| \leqslant h^{(k)} \cdot \mathrm{TOL}. \tag{4.13}$$

2. Local error per unit step, combined with extrapolation;
   (Dormand and Prince [15, 16], Shampine [53])

   Most commonly used, this approach maintains two Runge-Kutta methods of orders p and $p-1$. Again using the difference to the higher order step as an estimate asymptotically correct for the lower-order method,

   $$\mathbf{est} := \boldsymbol{\Delta}^{[p]} - \boldsymbol{\Delta}^{[p-1]}, \tag{4.14}$$

   this time we advance with the higher-order step and choose $h^{(k)}$ to satisfy

   $$||\mathbf{est}|| \leqslant \mathrm{TOL}. \tag{4.15}$$

   Both of the above strategies are most efficiently implemented with embedded Runge-Kutta methods due to Fehlberg [21].

3. Defect control;

   This approach suits continuous Runge-Kutta methods such as those discussed in Section 4.5, which have a continuous representation $\boldsymbol{\varrho}(t)$ of the approximate solution available on the time slice $[t^{(k)}, t^{(k)} + h^{(k)}]$.

We effectively require the interpolant to obey a local approximation of order $p+1$ for a method of order p and obtain an asymptotically correct error estimate in the spirit of the first strategy.

With any of the above strategies, for strict tolerances TOL the controlled step size h may become very small. Large time intervals $[t_0, t_f]$ may thus require an excessively high number of integration steps to be computed. Implementations should therefore introduce an upper limit to the number of integration steps and trigger an error condition should this limit be exceeded.

4.2.4 Area of Stability

Stability analysis for one-step methods is carried out on the linear test system

$$\frac{d\boldsymbol{y}}{dt}(t) = \boldsymbol{A}\cdot\boldsymbol{y}(t), \quad t\in\mathcal{T}, \tag{4.16a}$$

$$\boldsymbol{y}(t_0) = \boldsymbol{y_0}. \tag{4.16b}$$

Assuming $\boldsymbol{A}\in\mathcal{M}(n,\mathbb{R})$ to be non-singular there exist non-singular matrices $\boldsymbol{S},\boldsymbol{T}$ such that

$$\boldsymbol{A} = \boldsymbol{S}\cdot\operatorname{diag}(\lambda_1,\ldots,\lambda_n)\cdot\boldsymbol{T}^{-1}, \tag{4.17}$$

where $\lambda_i\in\mathbb{C}$, $i=1,\ldots,n$ are the eigenvalues of $\boldsymbol{A}$. The test system has a stable fixed point in $\boldsymbol{y}=\boldsymbol{0}$ iff all eigenvalues are found in the negative real half of the complex plane $\mathbb{C}$, i.e., $\Re\lambda_i<0$. A stiff ODE system is characterized by eigenvalues with widely differing absolute values. Such systems frequently are impossible to solve with an explicit solver such as explicit Runge-Kutta methods, or require excessively many small steps to be computed.
The area $\mathcal{A}$ of stability for a one-step method $\boldsymbol{\Phi}$ is the set of step sizes h and eigenvalues λ for which the fixed point is stable under the approximation scheme $\boldsymbol{\Phi}$,

$$\boldsymbol{\eta}^{(k+1)} := \boldsymbol{\eta}^{(k)} + h\cdot\boldsymbol{\Phi}(t^{(k)},h,\boldsymbol{\eta}^{(k)},\boldsymbol{A}), \quad \text{and} \quad \boldsymbol{\eta}^{(k)}\longrightarrow\boldsymbol{0} \text{ for } k\longrightarrow\infty. \tag{4.18}$$

This is the case whenever the approximation scheme's eigenvalues are found within the unit disc. Restricting ourselves to the single eigenvalue λ with largest absolute value, we are thus interested in the one-dimensional Runge-Kutta map's derivative, evaluated in the fixed point,

$$\left.\frac{d\eta^{(k+1)}}{d\eta^{(k)}}\right|_{\eta^{(k)}=0} = 1 + h\cdot\frac{d\Phi}{d\eta^{(k)}}(h,0,\lambda), \tag{4.19}$$

and find

$$\mathcal{A}(\Phi) := \left\{(h,\lambda)\in\mathbb{R}\times\mathbb{C}\;\middle|\;\left|1+h\cdot\tfrac{d\Phi}{dy}(h,0,\lambda)\right|<1\right\}. \tag{4.20}$$

Figure 4.2 shows the stability area of the Fehlberg method RKN3(4) and RKF4(5) in $h\lambda$ space.

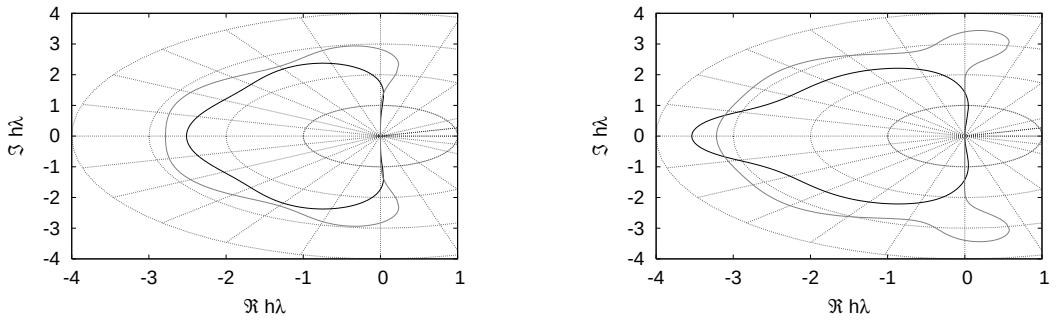


Fig. 4.2: Areas of stability for the Runge-Kutta methods RKN3(4) (left) and RKF4(5) (right). (——) and (——) bound the lower- and higher-order method's area respectively.

4.3 Implicitly Defined Discontinuities

For the following discussion of implicitly defined discontinuities we consider a non-stiff initial value problem on $\mathcal{T} = [t_0, t_\mathrm{f}] \subset \mathbb{R}$, depending on a time-constant parameter vector $\boldsymbol{p} \in \mathbb{R}^{n_p}$ and the switching function's sign structure $\operatorname{sgn} \boldsymbol{\sigma}$. The problem's definition is explained in the section below.

Definition 4.11. ***Parameter-dependent Initial Value Problem with Switches***

$$\frac{d\boldsymbol{y}}{dt}(t) = \boldsymbol{f}\big(t, \boldsymbol{y}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \quad t \in \mathcal{T}, \tag{4.21a}$$

$$\boldsymbol{y}(t_0) = \boldsymbol{y_0}, \tag{4.21b}$$

$$\boldsymbol{y}_+(t_\mathrm{s}) = \boldsymbol{\Delta}_j\big(t_\mathrm{s}, \boldsymbol{y}_-(t_\mathrm{s}), \boldsymbol{p}\big) \quad \forall t_\mathrm{s} \in \mathcal{T} : \ \exists j \in \mathbb{N} : \ \sigma_j(t_\mathrm{s}) = 0. \tag{4.21c}$$

4.3.1 Defining Switches

A discontinuity in the IVP's right-hand side $\boldsymbol{f}$ or in its state vector $\boldsymbol{y}$ is called a *switch*. The activation time $t_\mathrm{s} \in \mathcal{T}$ of switch $j \in \{1, \ldots, n_\mathrm{s}\}$ is implicitly defined by a zero-crossing of the corresponding component of the switching function $\boldsymbol{\sigma}$

$$\boldsymbol{\sigma} : \mathcal{T} \times \mathbb{R}^{n_y} \times \mathcal{P} \longrightarrow \mathbb{R}, \quad \big(t, \boldsymbol{y}(t), \boldsymbol{p}\big) \mapsto \boldsymbol{\sigma}\big(t, \boldsymbol{y}(t), \boldsymbol{p}\big). \tag{4.22}$$

At any time $t \in \mathcal{T}$, the sign structure of $\boldsymbol{\sigma}$ uniquely identifies the activation state of any switch. We assume σ_j to be a $\mathscr{C}^1$ function of the time t, the states $\boldsymbol{y}$, and the parameters $\boldsymbol{p}$. By virtue of the implicit function theorem, the switching point t_s may be considered a function of $(t_0, \boldsymbol{y_0}, \boldsymbol{p})$.

We now introduce some notations to be used when discussing implicitly defined discontinuities. Imagine for a moment that only a single switch is present in the problem. The left-hand side limit of the states at the switch time t_s is called $\boldsymbol{y}_-$, while $\boldsymbol{y}_+$ is the right-hand side limit, i.e., the new states after the switch.

$$\boldsymbol{y}_-(t_\mathrm{s};\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) := \lim_{\substack{\varepsilon \to 0 \\ \varepsilon < 0}} \boldsymbol{y}(t_\mathrm{s} + \varepsilon;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}), \tag{4.23}$$

$$\boldsymbol{y}_+(t_\mathrm{s};\ \boldsymbol{y}_-, \boldsymbol{\Delta}, \boldsymbol{p}) := \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \boldsymbol{y}(t_\mathrm{s} + \varepsilon;\ \boldsymbol{y}_-, \boldsymbol{\Delta}, \boldsymbol{p}). \tag{4.24}$$

We denote with $\boldsymbol{\Delta}$ the jump function of the differential states that becomes effective when the switch is activated,

$$\boldsymbol{\Delta} : \mathcal{T} \times \mathbb{R}^{n_y} \times \mathcal{P} \longrightarrow \mathbb{R}^{n_y}, \quad \big(t, \boldsymbol{y}_-(t), \boldsymbol{p}\big) \mapsto \boldsymbol{y}_+(t). \tag{4.25}$$

Equivalently, for the right-hand side of the ODE system we'll denote with $\boldsymbol{f}_-$ the right-hand side in $(t_\mathrm{s}, \boldsymbol{y}_-, \boldsymbol{p})$, and with $\boldsymbol{f}_+$ the one in $(t_\mathrm{s}, \boldsymbol{y}_+, \boldsymbol{p})$,

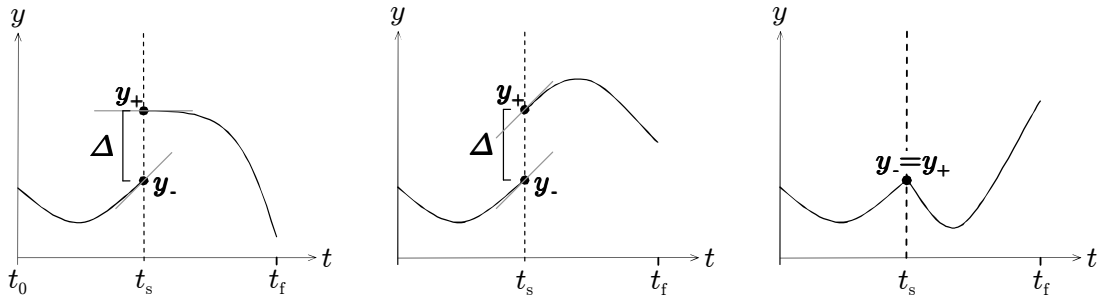
$$\boldsymbol{f}_-(t_\mathrm{s}, \boldsymbol{y}_-, \boldsymbol{p}) := \lim_{\substack{\varepsilon \to 0 \\ \varepsilon < 0}} \boldsymbol{f}(t_\mathrm{s} + \varepsilon, \boldsymbol{y}_-, \boldsymbol{p}), \tag{4.26}$$

$$\boldsymbol{f}_+(t_\mathrm{s}, \boldsymbol{y}_+, \boldsymbol{p}) := \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \boldsymbol{f}(t_\mathrm{s} + \varepsilon, \boldsymbol{y}_+, \boldsymbol{p}). \tag{4.27}$$

The right-hand side's jump vector is $\boldsymbol{\delta}$,

$$\boldsymbol{\delta}(t_\mathrm{s}, \boldsymbol{y}_-(t_\mathrm{s};\ \boldsymbol{y_0}, \boldsymbol{p}), \boldsymbol{p}) := \boldsymbol{f}_+(t_\mathrm{s}, \boldsymbol{y}_+, \boldsymbol{p}) - \boldsymbol{f}_-(t_\mathrm{s}, \boldsymbol{y}_-, \boldsymbol{p}). \tag{4.28}$$

Note that the discontinuity in the right-hand side need not be solely due to the jump $\boldsymbol{\Delta}$ in the states; the framework allows for $\boldsymbol{\delta} \neq \boldsymbol{0}$ even in the case of $\boldsymbol{\Delta} = \boldsymbol{0}$, e.g., exchanging the right-hand side function while maintaining continuity, but not differentiability, in the states. The three possible cases are visualized in Fig. 4.3.



(a) A full switch.

(b) Discontinuity in the states only.

(c) Discontinuity in the right-hand side only.

Fig. 4.3: Switches are discontinuities in the states and right-hand side.

4.3.2 Classification of Switches

Considering the one-sided derivatives of the switch function with respect to time

$$\frac{d\sigma_j^-}{dt_s}(t_s) := \frac{d\sigma_j}{dt_s}(t_s,\ \boldsymbol{y}_-(t_s;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}),\ \boldsymbol{p}), \tag{4.29}$$

$$\frac{d\sigma_j^+}{dt_s}(t_s) := \frac{d\sigma_j}{dt_s}(t_s,\ \boldsymbol{y}_+(t_s;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}),\ \boldsymbol{p}), \tag{4.30}$$

a switch event may fall into several different categories with varying difficulties in numerical treatment, see Bock [6] and Brandt-Pollmann [8]. Here we list the different conditions characterizing the respective switch types.

1. Consistent switching:

   (a) $\boldsymbol{\Delta} = \boldsymbol{0}$, no jump occurs in the differential states:

   $$\frac{d\sigma_j^-}{dt}(t_s) \cdot \frac{d\sigma_j^+}{dt_s}(t_s) > 0. \tag{4.31}$$

   (b) $\boldsymbol{\Delta} \neq \boldsymbol{0}$, a jump in the differential states is present:

   $$\frac{d\sigma_j^-}{dt_s}(t_s) \cdot \sigma_j^+(t_s,\ \boldsymbol{y}_+(t_s;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}),\ \boldsymbol{p}) > 0. \tag{4.32}$$

   The manifold defined by $\sigma_j = 0$ is crossed in either direction. This is the only type of switch we will encounter in the *DaimlerChrysler* powertrain model of Chapter 1.

2. Vanishing derivatives:

   $$\frac{d\sigma_j^-}{dt_s}(t_s) = 0 \quad \vee \quad \frac{d\sigma_j^+}{dt_s}(t_s) = 0. \tag{4.33}$$

   A consistent solution may exist, but its detection would require the evaluation of higher-order derivatives of σ_j. This case is not handled in the presented implementation; we stop with a suitable diagnosis message.

3. A bifurcation:

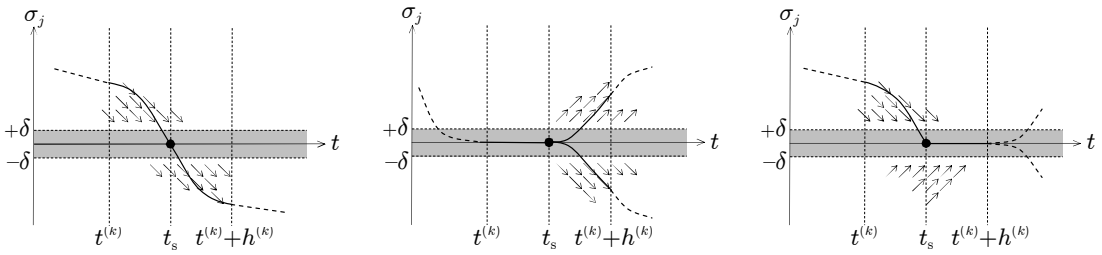
   $$\frac{d\sigma_j^-}{dt_s}(t_s) > 0 \quad \wedge \quad \frac{d\sigma_j^+}{dt_s}(t_s) < 0. \tag{4.34}$$

   A solution currently on the discontinuity manifold defined by $\sigma_j = 0$ may leave it in both directions. Additional reasoning is required to decide which direction to choose. We detect this type of switching event and stop the integration with a suitable diagnosis message.

4. An inconsistent switch:

$$\frac{d\sigma_j^-}{dt_\mathrm{s}}(t_\mathrm{s}) < 0 \quad \wedge \quad \frac{d\sigma_j^+}{dt_\mathrm{s}}(t_\mathrm{s}) > 0. \tag{4.35}$$

A solution approaching the discontinuity manifold from either direction must stay on it as the manifold cannot be left. Brandt-Pollmann [8] describes techniques due to Filippov [22] that may be applied to continue the integration process in this case. In view of the application we aimed at, these techniques have not yet been implemented. We detect inconsistent switching and stop the integration process.



(a) A consistent switch. (b) A bifurcation. (c) An inconsistent switch.

Fig. 4.4: Classification of switch events via derivatives of the switching function. (- - -) indicates past or future trajectories while (——) represents the trajectory during the step across t_s, marked by (•). Arrows (↗↗↗) visualize the vector field of the switching function σ_j.

4.3.3 Detecting Switches

The detection of implicit switches requires the model to allow for its evaluation in a sufficiently large surrounding of the state space around the switching point, cf. Fig. 4.5.

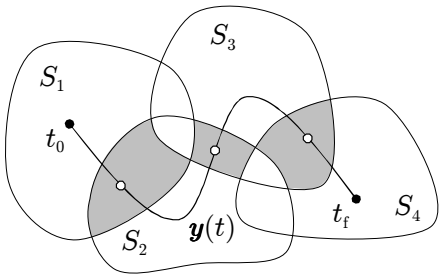


Fig. 4.5: Schematic of the switch state space. The solution $\boldsymbol{y}$ (——) on $[t_0, t_1]$ traverses validity regions S_i of different sub-models. Overlappings of these regions required around a switch point (○) are shaded (▬).

This makes it possible for the integrator to perform a full integration step across the switching point, which is detected by evaluation and comparison of the sign structure of σ before and after the step. A switch j is recognized on $[t^{(k)}, t^{(k+1)}] \subset \mathcal{T}$ if there exists $j \in \{1, \ldots, n_\mathrm{s}\}$ such that

$$\operatorname{sgn} \sigma_j \left(t^{(k)}, \boldsymbol{y}(t^{(k)}), \boldsymbol{p} \right) \neq \operatorname{sgn} \sigma_j \left(t^{(k+1)}, \boldsymbol{y}(t^{(k+1)}), \boldsymbol{p} \right).$$

In this case, the exact switch time t_s is found by locating the zero-crossing of the scalar switch function $\sigma_j(t, \boldsymbol{y}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}), \boldsymbol{p})$ within the interval $[t^{(k)}, t^{(k+1)}]$. Since for efficiency reasons we don't want to repeatedly call the integrator to compute the solution $\boldsymbol{y}(t)$ for a

certain time point t within this interval, locating $\sigma_j = 0$ requires a continuous representation of the solution $\boldsymbol{y}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p})$. Furthermore, since we are merely interested in sign changes, this representation needs to be exact only in a neighborhood of the actual switching point. Contrary to backward differentiation formula (BDF) methods discussed by Brandt-Pollmann [8], ordinary explicit Runge-Kutta methods do not naturally provide such a representation. This problem is resolved in Section 4.5.

Assuming a continuous representation $\boldsymbol{\varrho}(t)$ of the solution $\boldsymbol{y}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p})$ to be available on $[t^{(k)}, t^{(k+1)}]$, the derivative-free zero finding Algorithm 4.1 due to Brent [9], based on inverse quadratic interpolation safeguarded by regula falsi and a bisection strategy, is employed to find t_s with sufficient accuracy. Contrary to the secant method or to Newton's method, this algorithm guarantees convergence. The rate of convergence usually is the best of the three available methods.

Algorithm 4.1 Zero-Finding Algorithm due to Brent & Decker

Input: Function σ_j, search interval $[t_0, t_1]$, tolerance τ^*, machine precision ε.
Output: Approximation $t \in [t_0, t_1]$ to the root of σ_j, with $|\sigma_j(t)| < \tau^*$.

```
Set σ0 ← σj(t0), σ1 ← σj(t1), t2 ← t0, σ2 ← σ0.
repeat
  Set τ ← 2ε|σ1| + τ*/2, h ← t1 − t0, h' ← ½(t2 − t1).
  if |h'| ⩽ τ ∨ σ1 = 0 then stop, (t1, σ1) is the approximation to the root.
  if |h| > τ ∧ |σ0| > |σ1| then
    if t0 = t2 then
      Set p ← (σ1/σ0)(t2 − t1), q ← 1 − σ1/σ0.
    else
      Set p ← (σ1/σ0) ( (σ0/σ2)((σ0 − σ1)/σ2)(t2 − t1) − ((σ1/σ2) − 1)(t1 − t0) ).
      Set q ← ((σ0/σ2) − 1)((σ1/σ2) − 1)((σ1/σ0) − 1).
    end if
    If p/q ⩽ ¾(t2 − t1) − sgn(q) τ/2 ∧ |p/q| ⩽ |h|/2 then h' ← p/q.
    Set h' ← sgn(h') max{|h'|, τ}.
  end if
  Perform the step: (t0, σ0) ← (t1, σ1), t1 ← t1 + h', σ1 ← σj(t1).
  if |σ1| > |σ2| then (t1, σ1) ↔ (t2, σ2).
  if sgn(σ1) = sgn(σ2) then (t2, σ2) ← (t0, σ0).
until —
```

Numerically, we treat switch function values $\sigma_j(t)$ within a strip $[-\delta, +\delta] \subset \mathbb{R}$ as being zero. Scaling of the ODE system's states $\boldsymbol{y}$ allows for a fixed choice of δ, e.g., $\delta = 10^{-6}$. Equally, our implementation necessarily supports the treatment of multiple simultaneously occurring switches. Here we treat switch events occurring at points t_s^1 and t_s^2 in time as simultaneous if $\left|t_\mathrm{s}^1 - t_\mathrm{s}^2\right| \leqslant \varepsilon$. Again, scaling of the time horizon to $\mathcal{T} = [0, 1] \subset \mathbb{R}$ allows for the choice of a fixed value ε as above.

4.4 Sensitivity Generation

The numerical solution of parameter estimation problems (Chapter 5) and optimal control problems (Chapters 6 and 7) using Newton-Lagrange techniques as presented in Chapter 3 requires the availability of derivatives of the IVP's solution with respect to unknowns, such as initial values $\boldsymbol{y_0}$ and parameters $\boldsymbol{p}$.

4.4.1 Differentiability of the Solution

The existence of such derivatives is guaranteed by the following differentiability theorems under mild conditions.

Theorem 4.12. ***Differentiability with Respect to Initial Values and Parameters***
Let $\boldsymbol{f}$ be in $\mathscr{C}^0(\mathcal{T} \times \mathbb{R}^{n_y}, \mathbb{R}^{n_y})$ and assume the Jacobians $\nabla_{\boldsymbol{y}} \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p})$ and $\nabla_{\boldsymbol{p}} \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p})$ to exist and be continuous and bounded thereon.
Then the solution $\boldsymbol{y}(t)$ of the IVP (4.21) is continuously differentiable on $\mathcal{T}$ with respect to the initial value $\boldsymbol{y_0}$ and the parameters $\boldsymbol{p}$.

Proof. A proof may be found in Stoer and Bulirsch [56]. □

Definition 4.13. ***Sensitivity with Respect to Initial Values and Parameters***
The matrix $\boldsymbol{G}_{\mathrm{y}}(t_1;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) \in \mathcal{M}(n_y \times n_y, \mathbb{R})$ denotes the sensitivities of the solution $\boldsymbol{y}(t_1)$ with respect to the initial values in $\boldsymbol{y_0} \in \mathbb{R}^{n_y}$,

$$\boldsymbol{G}_{\mathrm{y}}(t_1;\ t_0) := \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{y_0}}(t_1;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}), \tag{4.36}$$

while the matrix $\boldsymbol{G}_{\mathrm{p}}(t_1;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) \in \mathcal{M}(n_y \times n_p, \mathbb{R})$ denotes the sensitivities of the solution $\boldsymbol{y}(t_1)$ with respect to the parameters $\boldsymbol{p}$,

$$\boldsymbol{G}_{\mathrm{p}}(t_1;\ t_0) := \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{p}}(t_1;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}). \tag{4.37}$$

When unambiguous, we will specify the time interval only, and omit the implicit dependencies of the sensitivity matrices on $\boldsymbol{y_0}$ and $\boldsymbol{p}$.

Theorem 4.14. ***Propagation of Sensitivities***
The sensitivity matrices with respect to $\boldsymbol{y_0}$ satisfy

1. $\boldsymbol{G}_{\mathrm{y}}(t_b;\ t_a) = \boldsymbol{G}_{\mathrm{y}}(t_a;\ t_b)^{-1}$ *for* $[t_a, t_b] \subseteq \mathcal{T}$,
2. $\boldsymbol{G}_{\mathrm{y}}(t_c;\ t_a) = \boldsymbol{G}_{\mathrm{y}}(t_c;\ t_b) \cdot \boldsymbol{G}_{\mathrm{y}}(t_b;\ t_a)$ *for* $t_b \in [t_a, t_c] \subseteq \mathcal{T}$.

The sensitivity matrices with respect to $\boldsymbol{p}$ satisfy

1. $\boldsymbol{G}_{\mathrm{p}}(t_b;\ t_a) = \boldsymbol{G}_{\mathrm{p}}(t_a;\ t_b)^{-1}$ *for* $[t_a, t_b] \subseteq \mathcal{T}$,
2. $\boldsymbol{G}_{\mathrm{p}}(t_c;\ t_a) = \boldsymbol{G}_{\mathrm{y}}(t_c;\ t_b) \cdot \boldsymbol{G}_{\mathrm{p}}(t_b;\ t_a) + \boldsymbol{G}_{\mathrm{p}}(t_c;\ t_b)$ *for* $t_b \in [t_a, t_c] \subseteq \mathcal{T}$.

Proof. Immediately results from standard calculus. □

4.4.2 Generating Sensitivities of the Solution

For the efficient and numerically stable generation of sufficiently precise first-order derivatives we make use of internal numerical differentiation (IND, Bock [4, 5]). Alternatively, the explicit solution of variational differential equations along with the nominal trajectory is possible. For systems with considerably more parameters than differential states, or for system with many local parameters, the computation of adjoint sensitivities is more efficient. Detailed presentations of both approaches can be found in Bock [6].

Internal Numerical Differentiation

The derivatives with respect to $\boldsymbol{y_0}$, and $\boldsymbol{p}$ analogously, are approximated by one-sided finite differences

$$\frac{\partial \boldsymbol{y}}{\partial y_{0,i}}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) \approx \frac{\boldsymbol{y}(t;\ t_0, \boldsymbol{y_0} + \varepsilon s_i \boldsymbol{e^i}, \boldsymbol{p}) - \boldsymbol{y}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p})}{\varepsilon}, \tag{4.38}$$

obtained from trajectories disturbed by $\varepsilon s_i \boldsymbol{e^i}$ integrated along with the solution. The fundamental principle of internal numerical differentiation (IND), as opposed to external numerical differentiation (END), is to generate derivatives of the approximative solution by differentiation of the integrator method itself. Adaptive parts of the method, such as error control and step size selection, are evaluated for the nominal solution only and get frozen during sensitivity generation (cf. Bock [4, 5, 6]). Nominal and varied trajectories are integrated simultaneously, using a common discretization scheme dictated by the error-controlled variable-step size Runge-Kutta method. For each directional derivative, the ODE system grows by n_x differential states, which yields $n_x(n_x + n_p)$ states for the full set of derivatives, if required.

Variational Differential Equations

The variational differential equations augment the ODE system by differential equations whose unique solutions are the sensitivity matrices $\boldsymbol{G}_{\mathrm{y}}(t; t_0)$ and $\boldsymbol{G}_{\mathrm{p}}(t; t_0)$. They can be found by differentiating the integral form of the IVP formulation with respect to $\boldsymbol{y_0}$ and $\boldsymbol{p}$ respectively. In general, the computational load is no less than that of internal numerical differentiation (IND).
The sensitivity matrix trajectory $\boldsymbol{G}_{\mathrm{y}}(t; t_0)$ is the solution of the following variational initial-value problem

$$\frac{\partial \boldsymbol{G}_y}{\partial t}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t, \boldsymbol{y}, \boldsymbol{p}) \cdot \boldsymbol{G}_y(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}), \tag{4.39a}$$

$$\boldsymbol{G}_y(t_0;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) = \boldsymbol{I}. \tag{4.39b}$$

Analogously, the sensitivity matrix trajectory $\boldsymbol{G}_{\mathrm{p}}(t; t_0)$ is the solution of the following variational initial-value problem

$$\frac{\partial \boldsymbol{G}_p}{\partial t}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t, \boldsymbol{y}, \boldsymbol{p}) \cdot \boldsymbol{G}_p(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}(t, \boldsymbol{y}, \boldsymbol{p}), \tag{4.39c}$$

$$\boldsymbol{G}_p(t_0;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) = \frac{\partial \boldsymbol{y_0}}{\partial \boldsymbol{p}}(\boldsymbol{p}). \tag{4.39d}$$

Here we obtain derivatives of the solution $\boldsymbol{y}(t)$ from derivatives of the model functions $\boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p})$ with respect to $\boldsymbol{y}$ and $\boldsymbol{p}$. Hence, this approach lends itself to automatic differentiation (cf. Griewank et al. [25]) where one provides exact symbolic derivatives of $\boldsymbol{f}$.

4.4.3 Sensitivity Updates in Implicit Discontinuities

If discontinuities are present in the integration process, Runge-Kutta methods can cope with this difficulty using techniques like the one described in Section 4.3. The sensitivity information generated by the variational differential equations (4.39a, 4.39c), however, will generally be incorrect or even totally wrong. Bock [6], Mombaur [42] show how to handle sensitivity updates in the presence of discontinuities in both the differential states and the right-hand side of the ODE system; Brandt-Pollmann [8] presents the extension to the DAE case.

The differentiability of the ODE system's solution across a switching point t_{s} can in general be shown whenever the switching structure of the trajectory does not change in a neighborhood of the switching point, as stated in the theorem below. This means that the order and number of actually occurring implicit switches remains unchanged therein. Switch events must not appear, disappear, coincide, or come to lie on the borders of the integration step interval.

Theorem 4.15. ***Differentiability across Implicit Discontinuities*** *(Bock [6])*
Let $\mathcal{I} := [t_0, t_1]$ be an interval containing without loss of generality exactly one switching point in $t_\mathrm{s} \in \mathcal{I}$, which bipartitions the interval into $\mathcal{I}_\text{-} := [t_0, t_\mathrm{s}]$ and $\mathcal{I}_\text{+} := (t_\mathrm{s}, t_1]$. Let $\boldsymbol{y}(t)$ be a solution of the initial-value problem (4.21). The right-hand side $\boldsymbol{f}$, the jump function $\boldsymbol{\Delta}$ (4.25), and the switching function σ_j (4.22) are assumed to be sufficiently often continuous differentiable. Further, let the regularity conditions (4.31, 4.32) for a consistent switch event be satisfied.
Then there exists a ball $\mathcal{B} \subset \mathcal{T} \times \mathbb{R}^{n_y} \times \mathcal{P}$ around $(t_\mathrm{s}, \boldsymbol{y}(t_\mathrm{s};\ \boldsymbol{y}(t^{(k)}), \boldsymbol{p}), \boldsymbol{p})$ with $\boldsymbol{y}(t_\mathrm{s};\ \boldsymbol{y}(t^{(k)}), \boldsymbol{p})$ sufficiently smooth. All solutions of the IVP (4.21) within $\mathcal{B}$ have exactly one switching point.

Proof. See Bock [6]. □

We now show the derivation of these first-order sensitivity updates. Consider an integrator step from time point t_0 to t_1, with a switch occurring in $t_s \in (t_0, t_1) \subset \mathcal{T}$. Let's suppose for a moment that only a single switch $j \in \{1, \ldots, n_\mathrm{s}\}$ is present. In the switching point, the solution is characterized by a jump $\boldsymbol{\Delta}_j$ (4.25) in the states, and a jump $\boldsymbol{\delta}_j$ (4.28) in the right-hand side. The full dependencies of the solution in the end-point t_1 on $(t_0, \boldsymbol{y}(t_0), \boldsymbol{p})$ are as follows:

$$\begin{array}{rclcrcl} \boldsymbol{y_1} & := & \boldsymbol{y}\big(t_1;\ t_\mathrm{s}, \boldsymbol{y}_\text{+},\ \boldsymbol{p}\big), & & \boldsymbol{y}_\text{+} & := & \boldsymbol{y}_\text{-} + \boldsymbol{\Delta}_j, \\ \boldsymbol{\Delta}_j & := & \boldsymbol{\Delta}_j(t_\mathrm{s}, \boldsymbol{y}_\text{-}, \boldsymbol{p}), & & \boldsymbol{y}_\text{-} & := & \boldsymbol{y}_\text{-}(t_\mathrm{s};\ t_0, \boldsymbol{y_0}, \boldsymbol{p}), \\ \boldsymbol{y_0} & := & \boldsymbol{y}(t_0), & & & & \\ \sigma_j & := & \sigma_j(t_\mathrm{s}, \boldsymbol{y}_\text{-}, \boldsymbol{p}), & & t_\mathrm{s} & := & t_\mathrm{s}(t_0, \boldsymbol{y_0}, \boldsymbol{p}). \end{array} \tag{4.40}$$

Hence, the sensitivity matrix $\boldsymbol{G}_\mathrm{y}$ on this interval according to Definition 4.13 is

$$\boldsymbol{G}_\mathrm{y}(t_1;\ t_0) \quad = \quad \frac{d\boldsymbol{y_1}}{d\boldsymbol{y_0}} = \frac{\partial \boldsymbol{y_1}}{\partial t_\mathrm{s}} \frac{\partial t_\mathrm{s}}{\partial \boldsymbol{y_0}} + \frac{\partial \boldsymbol{y_1}}{\partial \boldsymbol{y}_\text{+}} \frac{\partial \boldsymbol{y}_\text{+}}{\partial \boldsymbol{y_0}}. \tag{4.41}$$

We resolve the individual differentials separately. For the first differential in (4.41), standard calculus yields

$$\frac{\partial \boldsymbol{y_1}}{\partial t_\mathrm{s}} \quad = \quad -\frac{\partial \boldsymbol{y_1}}{\partial \boldsymbol{y}_\text{+}} \frac{d\boldsymbol{y}_\text{+}}{dt_\mathrm{s}}. \tag{4.42}$$

For the second differential, observe that by definition of t_s we have $\sigma_j = 0$, i.e., the switching function does not actually depend on the initial value $\boldsymbol{y_0}$. This gives

$$\begin{array}{rcl} 0 \ = \ \dfrac{d\sigma_j}{d\boldsymbol{y_0}} & = & \dfrac{\partial \sigma_j}{\partial t_\mathrm{s}} \dfrac{\partial t_\mathrm{s}}{\partial \boldsymbol{y_0}} + \dfrac{\partial \sigma_j}{\partial \boldsymbol{y}_\text{-}} \dfrac{\partial \boldsymbol{y}_\text{-}}{\partial t_\mathrm{s}} \dfrac{\partial t_\mathrm{s}}{\partial \boldsymbol{y_0}} + \dfrac{\partial \sigma_j}{\partial \boldsymbol{y}_\text{-}} \dfrac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{y_0}} \ = \ \dfrac{d\sigma_j}{dt_\mathrm{s}} \dfrac{\partial t_\mathrm{s}}{\partial \boldsymbol{y_0}} + \dfrac{\partial \sigma_j}{\partial \boldsymbol{y}_\text{-}} \dfrac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{y_0}} \\ \Longrightarrow \quad \dfrac{\partial t_\mathrm{s}}{\partial \boldsymbol{y_0}} & = & -\left(\dfrac{d\sigma_j}{dt_\mathrm{s}}\right)^{-1} \dfrac{\partial \sigma_j}{\partial \boldsymbol{y}_\text{-}} \dfrac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{y_0}}. \end{array} \tag{4.43}$$

Finally, from strict expansion of the last differential in question in (4.41) we find

$$\frac{\partial \boldsymbol{y}_\text{+}}{\partial \boldsymbol{y_0}} \quad = \quad \left(\boldsymbol{I} + \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_\text{-}}\right) \frac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{y_0}} + \left(\frac{\partial \boldsymbol{\Delta}_j}{\partial t_\mathrm{s}} + \left(\boldsymbol{I} + \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_\text{-}}\right) \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}}\right) \frac{dt_\mathrm{s}}{d\boldsymbol{y_0}}. \tag{4.44}$$

Substitution of (4.43) into (4.41, 4.44) and (4.42, 4.44) into (4.41) afterwards yields

$$\boldsymbol{G}_\mathrm{y}(t_1;\ t_0) = \frac{d\boldsymbol{y_1}}{d\boldsymbol{y}_\text{+}} \left[\boldsymbol{I} + \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_\text{-}} + \left(\frac{d\boldsymbol{y}_\text{+}}{dt_\mathrm{s}} - \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial t_\mathrm{s}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_\text{-}} \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}}\right) \frac{\frac{\partial \sigma_j}{\partial \boldsymbol{y}_\text{-}}}{\frac{d\sigma_j}{dt_\mathrm{s}}}\right] \frac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{y_0}}. \tag{4.45}$$

By using the very same approach to derive the sensitivity matrix $\boldsymbol{G}_\mathrm{p}$ we obtain

$$\begin{aligned} \boldsymbol{G}_\mathrm{p}(t_1;\ t_0) \quad = \quad & \frac{d\boldsymbol{y_1}}{d\boldsymbol{y}_\text{+}} \left[\left(\boldsymbol{I} + \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_\text{-}} + \left(\frac{d\boldsymbol{y}_\text{+}}{dt_\mathrm{s}} - \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial t_\mathrm{s}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_\text{-}} \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}}\right) \frac{\frac{\partial \sigma_j}{\partial \boldsymbol{y}_\text{-}}}{\frac{d\sigma_j}{dt_\mathrm{s}}}\right) \frac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{p}} \right. \\ & \left. + \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{p}} + \left(\frac{d\boldsymbol{y}_\text{+}}{dt_\mathrm{s}} - \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial t_\mathrm{s}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{p}} \frac{d\boldsymbol{y}_\text{-}}{dt_\mathrm{s}}\right) \frac{\frac{\partial \sigma_j}{\partial \boldsymbol{p}}}{\frac{d\sigma_j}{dt_\mathrm{s}}}\right] \frac{d\boldsymbol{y}_\text{-}}{d\boldsymbol{y_0}} + \boldsymbol{G}_\mathrm{p}(t_1;\ t_\mathrm{s}). \end{aligned} \tag{4.46}$$

By collecting the following update matrices from Eq. (4.45) and (4.46), and by substituting the right-hand side function $\boldsymbol{f}$ and its discontinuity jump $\boldsymbol{\delta}_j$ where appropriate,

$$\boldsymbol{U}_{\mathrm{y}} = \boldsymbol{I} + \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_{\text{-}}} + \left(\boldsymbol{\delta}_j - \frac{\partial \boldsymbol{\Delta}_j}{\partial t_{\mathrm{s}}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{y}_{\text{-}}} \boldsymbol{f}(t_{\mathrm{s}}, \boldsymbol{y}_{\text{-}}, \boldsymbol{p}) \right) \frac{\frac{\partial \sigma_j}{\partial \boldsymbol{y}_{\text{-}}}}{\frac{d\sigma_j}{dt_{\mathrm{s}}}}, \tag{4.47a}$$

$$\boldsymbol{U}_{\mathrm{p}} = \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{p}} + \left(\boldsymbol{\delta}_j - \frac{\partial \boldsymbol{\Delta}_j}{\partial t_{\mathrm{s}}} - \frac{\partial \boldsymbol{\Delta}_j}{\partial \boldsymbol{p}} \boldsymbol{f}(t_{\mathrm{s}}, \boldsymbol{y}_{\text{-}}, \boldsymbol{p}) \right) \frac{\frac{\partial \sigma_j}{\partial \boldsymbol{p}}}{\frac{d\sigma_j}{dt_{\mathrm{s}}}}, \tag{4.47b}$$

we may finally present the sensitivity updates to $\boldsymbol{G}_{\mathrm{y}}$ and $\boldsymbol{G}_{\mathrm{p}}$ in the following form:

$$\boldsymbol{G}_{\mathrm{y}}(t_1;\, t_0) = \boldsymbol{G}_{\mathrm{y}}(t_1;\, t_{\mathrm{s}})\, \boldsymbol{U}_{\mathrm{y}}\, \boldsymbol{G}_y(t_{\mathrm{s}};\, t_0), \tag{4.48a}$$

$$\boldsymbol{G}_{\mathrm{p}}(t_1;\, t_0) = \boldsymbol{G}_{\mathrm{y}}(t_1;\, t_{\mathrm{s}})\, (\boldsymbol{U}_{\mathrm{y}}\, \boldsymbol{G}_{\mathrm{p}}(t_{\mathrm{s}};\, t_0) + \boldsymbol{U}_{\mathrm{p}})\, \boldsymbol{G}_{\mathrm{y}}(t_{\mathrm{s}};\, t_0) + \boldsymbol{G}_{\mathrm{p}}(t_1;\, t_{\mathrm{s}}). \tag{4.48b}$$

When implementing these updates using finite differences, as opposed to automatic differentiation [25], it is important to remember that the detected switching time t_{s} is an approximation. In order to avoid instabilities, one should *not* assume $\sigma(t_{\mathrm{s}}) = 0$ but compute the derivatives of the switching function σ using two proper evaluations (in the case of one-sided finite differences), cf. Bock [6].

4.5 Continuous Extensions

In Section 4.3.3 we established the need for evaluation of the approximates at arbitrary points in time. This sections now focusses on the *dense output problem* connected to Runge-Kutta methods: Contrary to our needs, the iteration scheme (4.2) computes approximations $\boldsymbol{\eta}^{(k)}$ to the solution $\boldsymbol{y}(t)$ of the IVP only at discrete time points $t^{(k)}$ determined by the step sizes $h^{(k)}$.

A quick solution to this problem is to introduce for the step size h an upper bound $\overline{h}$ small enough to guarantee the required evaluation resolution. Obviously, for very fine resolutions this entirely defeats the purpose of error-controlled variable step size methods. Even for coarser resolutions the number of performed steps, and thus the number of potentially costly evaluations of the ODE system's right-hand side, will be much larger than necessary. Ultimately, for the purpose of switch detection this approach is not suitable at all, since the evaluation point is implicitly defined here. We cannot know in advance at which points on the integrator step interval we are going to evaluate the continuous solution, and thus cannot infer satisfactory upper bounds $\overline{h}$.

4.5.1 Literature Survey

In the literature one can find several more sophisticated approaches to the dense output problem. One of the first attempts, limited to Fehlberg's pair RKF4(5) and yielding a $\mathscr{C}^0$ interpolant, was described by Horn [30]. Subsequent approaches yield significantly improved interpolants that mainly differ in the following aspects:

1. The continuity order of the obtained interpolant across integrator steps;

   Shampine [53] argues that a $\mathscr{C}^1$ interpolant is highly desirable, and most of the approaches found agree with this assertion: Enright et al. [19, 20], Dormand and Prince [16], Owren and Zennaro [46]. Higham [29] also covers the construction of interpolants from $\mathscr{C}^n$, $n > 1$, at significantly increased computational cost.

2. The applicability of the method used to construct the interpolant;

   While Horn [30] initially constructed her interpolants through ad-hoc observations, Owren and Zennaro [44, 46] construct interpolants by solving a continuous variant

of the order conditions (cf. Butcher [10]) for the specific method in question. They consider amongst others the popular extrapolation pair by Dormand and Prince [15]. In Enright [18] a large collection of newly created higher-order continuous Runge-Kutta schemes is presented.

On the other hand, in Enright et al. [19], an iterative approach based on the solution of Hermite-Birkhoff interpolation problems is developed, that allows for generic construction of $\mathscr{C}^1$ interpolants for existing explicit and implicit Runge-Kutta methods. Very similar approaches are described by Dormand and Prince [16] and Shampine [53].

3. The truncation order, and computational cost incurred by evaluating the interpolant;

   The papers by Owren and Zennaro [44, 45] give some theoretical foundations and derive lower bounds for the number of right-hand side evaluations required to guarantee an interpolant with a given approximation order. The iterative scheme of Enright et al. [19] allows for trade-offs here.

We found that the paper by Enright et al. [20] serves as an excellent starting point for research on existing literature in this area.

4.5.2 Theoretical Limits

Concerning the stage count of continuous embedded Runge-Kutta methods with $\mathscr{C}^1$ interpolants across integrator step boundaries, Owren and Zennaro [45, 46] prove that for any Fehlberg-type Runge-Kutta method of order $(n, n+1)$ employing the minimal number of required stages, there is no $\mathscr{C}^1$ continuous extension of global approximation order n with the same stage count. In other words, at least one additional stage and evaluation of the right-hand side is required, and it will be seen that in the case of RKF4(5) even two evaluations are needed. Tab. 4.1 collects minimal required stage counts for various flavors of Runge-Kutta methods. It may be possible to employ a first-same-as-last (FSAL) strategy where the last stage of the method can be reused as the first if the next step. For some methods and orders, e.g., for CRK(3) to CRK(5), the number of required stages may be reduced by one (see, e.g., Enright et al. [20]).

Order p	1	2	3	4	5	6	7	8
RKp	1	2	3	4	6	7	9	11
CRKp	1	2	4	6	8	11	$\geqslant 12$	$\geqslant 14$
RKF$p(p+1)$	2	3	5	6	8		13	
CRKF$p(p+1)$	3	4	6	8	11			

Tab. 4.1: Minimal stage counts of explicit Runge-Kutta (RK) and Fehlberg (RKF) methods as well as their $\mathscr{C}^1$ continuous counterparts (CRK, CRKF).

4.5.3 Construction of Interpolants

For the discussion to follow, we focus on supplying the *existing* Runge-Kutta-Fehlberg methods of the optimal control software package *MUSCOD-II* [12] with $\mathscr{C}^1$ interpolants. Thus, the approach of Enright et al. [19] appears to best suit our needs, and we briefly present it in this section.

The Cubic Hermite Polynomial

An immediate and obvious idea is to construct such an interpolating polynomial $\boldsymbol{\varrho}(t)$ on $[t^{(k)}, t^{(k+1)}]$ from the values

$$\begin{aligned} \boldsymbol{\varrho}(t^{(k)}) &= \boldsymbol{\eta}(t^{(k)}), & \boldsymbol{\varrho}(t^{(k+1)}) &= \boldsymbol{\eta}(t^{(k+1)}), \\ \frac{d\boldsymbol{\varrho}}{dt}(t^{(k)}) &= \boldsymbol{k}_1^{(k)}, & \frac{d\boldsymbol{\varrho}}{dt}(t^{(k+1)}) &= \boldsymbol{k}_1^{(k+1)}. \end{aligned} \tag{4.49}$$

The resulting third-order interpolation polynomial ϱ on $[t^{(k)}, t^{(k+1)}]$ is uniquely determined,

$$\boldsymbol{\varrho}(t^{(k)} + \tau h^{(k)}) = \beta_0(\tau)\boldsymbol{\eta}^{(k)} + \beta_1(\tau)h^{(k)}\boldsymbol{k}_1^{(k)} + \beta_2(\tau)\boldsymbol{\eta}^{(k+1)} + \beta_3(\tau)h^{(k)}\boldsymbol{k}_1^{(k+1)} \tag{4.50}$$

where $\beta_j(\tau)$, $\tau \in [0,1] \subset \mathbb{R}$ are the cubic Hermite base polynomials

$$\begin{aligned} \beta_0(\tau) &:= (2\tau+1)(\tau-1)^2, & \beta_1(\tau) &:= \tau(\tau-1)^2, \\ \beta_2(\tau) &:= \tau^2(3-2\tau), & \beta_3(\tau) &:= \tau^2(\tau-1). \end{aligned} \tag{4.51}$$

We can find from basic polynomial interpolation theory that for a one-step method of order p it satisfies

$$\sup_{\tau\in[0,1]} \left|\left| \boldsymbol{\varrho}(t^{(k)} + \tau h^{(k)}) - \boldsymbol{y}(t^{(k)} + \tau h^{(k)};\ t_0, \boldsymbol{y_0}) \right|\right| = \mathcal{O}(h^{(k)^q}), \quad q := \min\{4, p+1\} \tag{4.52}$$

As a Hermite interpolation polynomial, it is inherently $\mathscr{C}^1$ continuous across integrator steps. Note that the implementation of this interpolant in Runge-Kutta codes requires $k_1^{(k+1)}$ in (4.50) to be pre-calculated in the preceding step.
It is straightforward to compare the interpolant's error to the local error $\overline{\sigma}$ (4.6) of the one-step method. A requirement commonly found (cf. Shampine [53]) is that the interpolant be at least as accurate as the underlying method, i.e., we consider an interpolant of local approximation order $q = p + 1$ sufficient for Runge-Kutta methods with a convergence order up to p. By that measure, this plain Hermite interpolant is sufficiently precise for third-order Runge-Kutta methods only.

Hermite-Birkhoff Polynomials

An iterative process ("*bootstrapping method*") for the construction of higher-order interpolants based on a series ϱ_j of Hermite-Birkhoff interpolation polynomials starting with the presented cubic Hermite polynomial $\boldsymbol{\varrho}_0$ is presented by Enright et al. [19].
The underlying Hermite-Birkhoff interpolation problem extends the problem (4.49) by imposing matching constraints on the derivative of the interpolant in additional time points $\tau_{j,k} \in (0,1)$.

$$\begin{aligned} \frac{d\boldsymbol{\varrho}_j}{dt}(t^{(k)} + \tau_{j,k}h^{(k)}) &= \boldsymbol{f}\big(t^{(k)} + \tau_{j,k}h^{(k)}, \boldsymbol{\varrho}_{j-1}(t^{(k)} + \tau_{j,k}h^{(k)})\big), \\ q = 4, \quad j = 1, \ldots, p-q+1, \quad k &= 1, \ldots, j. \end{aligned} \tag{4.53}$$

The existence of a – not necessarily unique – solution to these Hermite-Birkhoff interpolation problems depends on the choice of the intermediate time points $\tau_{j,k} \in (0,1)$. For maximum computational efficiency, it would be desirable to select them from the vector of coefficients $\boldsymbol{\alpha} \in \mathbb{R}^s$ (4.4) found in the method's Butcher tableau, as this would save us from invoking additional and potentially costly evaluations of the ODE system's right-hand side. It is obvious from the results presented in Section 4.5.2, however, that this is not always possible.
In case of solvability it can be shown that for the polynomials $\boldsymbol{\varrho}_j$ one obtains

$$\sup_{\tau\in[0,1]} \left|\left| \boldsymbol{\varrho}_j(t^{(k)} + \tau h^{(k)}) - \boldsymbol{y}(t^{(k)} + \tau h^{(k)};\ t_0, \boldsymbol{y_0}) \right|\right| = \mathcal{O}(h^{(k)^{q+j}}), \tag{4.54}$$

a process that naturally terminates with $q + j = p + 1$, i.e., when the interpolant $\boldsymbol{\varrho}_j$ has finally reached the approximation order of the underlying one-step method's local error $\overline{\sigma}$. Again, the resulting interpolant is inherently $\mathscr{C}^1$ continuous.

4.5.4 Selected Interpolants

Amongst the selection of integrators available in *MUSCOD-II* are the popular Runge-Kutta-Fehlberg methods RKF1(2), RKF2(3), and RKF4(5) (cf. Fehlberg [21]). This section shows the interpolants we selected, and summarizes the reasoning behind the decisions.
For methods of order four and higher, the interpolant cannot be obtained for free. In view of the potentially large number of differential states due to the generation of first- and second-order sensitivities (see Chapter 7), we put emphasis on keeping the computational impact incurred by the interpolants to be selected as low as possible. After analysis of the various possibilities, we decided to trade precision for speed where inevitable.
Using the selected interpolant enabled our implementation to realize the following new features:

1. The efficient detection of implicit switches;
2. The evaluation of continuous objectives of least-squares;
3. A more elegant approach to the generation of on-line graphics output.

The Runge-Kutta-Fehlberg Method RKF4(5)

Using no additional evaluations of the right-hand side, Enright et al. [19] construct a locally fifth-order interpolant for Fehlberg's method RKF4(5), based on an observation described by Horn [30] that for

$$\boldsymbol{\eta}_{\frac{3}{5}} := \boldsymbol{\eta}^{(k)} + h^{(k)} \left(\tfrac{1559}{12500}\boldsymbol{k}_1 + \tfrac{153856}{296875}\boldsymbol{k}_3 + \tfrac{68107}{2612500}\boldsymbol{k}_4 - \tfrac{243}{31250}\boldsymbol{k}_5 - \tfrac{2106}{34375}\boldsymbol{k}_6 \right) \tag{4.55}$$

it can be shown that

$$\left|\left| \boldsymbol{y}(t^{(k)} + \tfrac{3}{5}h^{(k)}) - \boldsymbol{\eta}_{\frac{3}{5}} \right|\right| = \mathcal{O}({h^{(k)}}^5). \tag{4.56}$$

Furthermore, Horn [30] also showed that the point $\tau_{1,1} = \frac{3}{5}$ is unique. The solution to the corresponding Hermite-Birkhoff interpolation problem is

$$\boldsymbol{\varrho}(t^{(k)} + \tau h^{(k)}) = \beta_0(\tau)\boldsymbol{\eta}^{(k)} + \beta_1(\tau)h^{(k)}\boldsymbol{k}_1^{(k)} + \beta_2(\tau)\boldsymbol{\eta}^{(k+1)} + \beta_3(\tau)h^{(k)}\boldsymbol{k}_1^{(k+1)} + \beta_4(\tau)\boldsymbol{\eta}_{\frac{3}{5}}, \tag{4.57}$$

with the base polynomials

$$\begin{aligned}
\beta_0(\tau) &:= (\tau-1)^2(1-\tfrac{5}{3}\tau)(\tfrac{11}{3}\tau+1), & \beta_1(\tau) &:= \tau(\tau-1)^2(1-\tfrac{5}{3}\tau),\\
\beta_2(\tau) &:= \tau^2(\tfrac{3}{4}-\tfrac{5}{4}\tau)(9\tau-11), & \beta_3(\tau) &:= \tau^2(\tau-1)(\tfrac{5}{2}\tau-\tfrac{3}{2}),\\
\beta_4(\tau) &:= \tfrac{625}{36}\tau^2(\tau-1)^2.
\end{aligned} \tag{4.58}$$

A locally sixth-order interpolant is presented by Enright et al. [19] as well. After considering the solvability of the Birkhoff interpolation problem and analyzing the interpolant's error bounds, they chose $\tau_{2,1} := 0.86$ and $\tau_{2,2} := 0.93$ to obtain the quintic Hermite-Birkhoff interpolant

$$\begin{aligned}
\boldsymbol{\varrho}(t^{(k)} + \tau h^{(k)}) &= \beta_0(\tau)\boldsymbol{\eta}^{(k)} + \beta_1(\tau)h^{(k)}\boldsymbol{k}_1^{(k)} + \beta_2(\tau)\boldsymbol{\eta}^{(k+1)}\\
&+ \beta_3(\tau)h^{(k)}\boldsymbol{k}_1^{(k+1)} + \beta_4(\tau)h^{(k)}\boldsymbol{k}_7 + \beta_5(\tau)h^{(k)}\boldsymbol{k}_8,
\end{aligned} \tag{4.59}$$

using the base polynomials

$$\begin{aligned}
\beta_0(\tau) &:= (\tau-1)^2\left(\tfrac{375}{64}\tau^3 - \tfrac{8925}{1024}\tau^2 + 2\tau + 1\right),\\
\beta_1(\tau) &:= \tau(\tau-1)^2\left(\tfrac{5375}{3968}\tau^2 - \tfrac{19062325}{8189952}\tau + 1\right),\\
\beta_2(\tau) &:= -\tau^2\left(\tfrac{375}{64}\tau^3 - \tfrac{20925}{1024}\tau^2 + \tfrac{12949}{512}\tau - \tfrac{11997}{1024}\right),\\
\beta_3(\tau) &:= (\tau-1)^2\left(\tfrac{199625}{6272}\tau^2 - \tfrac{5385075}{100352}\tau + \tfrac{2291427}{100352}\right),\\
\beta_4(\tau) &:= \tau^2(\tau-1)^2\left(\tfrac{78125}{1568}\tau - \tfrac{47953125}{1078784}\right),\\
\beta_5(\tau) &:= -\tau^2(\tau-1)^2\left(\tfrac{234375}{3038}\tau - \tfrac{8734375}{145824}\right)
\end{aligned} \tag{4.60}$$

and requiring the two additional evaluations of the system's right-hand side per step,

$$\begin{aligned}
\boldsymbol{k}_7 &:= \boldsymbol{f}(t^{(k)} + \tau_{2,1} h^{(k)}, \boldsymbol{\eta}_{0.86}), && (4.61)\\
\boldsymbol{\eta}_{0.86} &:= -\tfrac{396851}{1250000}\boldsymbol{\eta}^{(k)} + \tfrac{3918031}{5000000}\boldsymbol{\eta}^{(k+1)} + \tfrac{90601}{360000}\boldsymbol{\eta}_{\frac{3}{5}} - h^{(k)}\left(\tfrac{27391}{3750000}\boldsymbol{k}_1^{(k)} + \tfrac{168259}{2500000}\boldsymbol{k}_1^{(k+1)}\right),\\
\boldsymbol{k}_8 &:= \boldsymbol{f}(t^{(k)} + \tau_{2,2} h^{(k)}, \boldsymbol{\eta}_{0.93}), && (4.62)\\
\boldsymbol{\eta}_{0.93} &:= -\tfrac{237699}{20000000}\boldsymbol{\eta}^{(k)} + \tfrac{75064671}{70000000}\boldsymbol{\eta}^{(k+1)} + \tfrac{47089}{640000}\boldsymbol{\eta}_{\frac{3}{5}} - h^{(k)}\left(\tfrac{50127}{20000000}\boldsymbol{k}_1^{(k)} + \tfrac{1997919}{40000000}\boldsymbol{k}_1^{(k+1)}\right).
\end{aligned}$$

The Runge-Kutta methods implemented in *MUSCOD-II* integrate a large number of varied trajectories or variational differential equations along with the nominal solution. They generally advance with the lower order method's step only, and also control the nominal solution's error only. They may thus save some computation time since it is possible to skip the computation of the higher-order method's coefficient $\boldsymbol{k}_6$ (4.4) for the sensitivity solutions.
Since the 5$^{\text{th}}$-order interpolant (4.57) relies on $\boldsymbol{k}_6$, this shortcut is no longer possible when using the presented interpolant. Once we accept the additional computations required to obtain $\boldsymbol{k}_6$, we may as well advance with the higher-order method's step and get the possibly improved precision for free. Then again, a fifth-order step would be best accompanied by locally sixth-order interpolant, requiring the evaluation of two additional stages as can be seen from Tab. 4.1 and Eq. (4.61).
For our implementation *RKFSWT* we favored the locally fifth-order interpolant (4.57) together with the higher-order step. This choice leads to moderately increased computational cost, while exploiting the highest possible increase in precision.

Orders			Required	RHS	
ε	η	ϱ	Stages	Evaluations	
5	4	–	6	$6+5n$	Existing RKF4(5) code.
5	4	3	6	$6+5n$	The Hermite interpolant is free.
5	4	4	6	$6+6n$	The Birkhoff interpolant is not.
5	5	4	6	$6+6n$	Advance with the higher-order step.
5	5	5	8	$8+8n$	A 5$^{\text{th}}$-order interpolant is costly.

Tab. 4.2: Cost of RKF realizations. The columns ε, η, and ϱ denote the global order of the error, the approximate, and the interpolant respectively. The number of computed trajectories is denoted by n.

The Runge-Kutta-Nørsett Method RKN4(3)

From Tab. 4.1 it is clear that there exists a continuous fourth-order method with a locally fifth-order interpolant and the computational cost of the non-continuous fifth-order method RKF4(5). Instead of accepting additional computations to obtain an interpolant while maintaining the method's order, we reduce the order to maintain the method's computational cost.
The fourth-order cubic Hermite interpolant $\boldsymbol{\varrho}_0$ (4.50) can be obtained without additional evaluations of the right-hand side. After one bootstrapping iteration with $\tau_{1,1} := \frac{1}{10}$,

$$\begin{aligned}
\boldsymbol{k}_6 &:= \boldsymbol{f}\left(t^{(k)} + \tfrac{1}{10}h^{(k)}, \boldsymbol{\varrho}_0(t^{(k)} + \tfrac{1}{10}h^{(k)})\right), && (4.63)\\
\boldsymbol{\varrho}_0(t^{(k)} + \tfrac{1}{10}h^{(k)}) &= \left(\tfrac{243}{250}\boldsymbol{\eta}^{(k)} + \tfrac{7}{250}\boldsymbol{\eta}^{(k+1)}\right) + h^{(k)}\left(\tfrac{81}{1000}\boldsymbol{k}_1^{(k)} - \tfrac{9}{1000}\boldsymbol{k}_1^{(k+1)}\right), && (4.64)
\end{aligned}$$

Enright et al. [19] derived the following fifth-order interpolant $\boldsymbol{\varrho}_1$ (4.65) for Nørsett's method RKN3(4), cf. Fig. 4.1(a):

$$\begin{aligned}
\boldsymbol{\varrho}_1(t^{(k)} + \tau h^{(k)}) &= \beta_0(\tau)\boldsymbol{\eta}^{(k)} \beta_1(\tau) h^{(k)} \boldsymbol{k}_1^{(k)}\\
&+ \beta_2(\tau)\boldsymbol{\eta}^{(k+1)} + \beta_3(\tau) h^{(k)} \boldsymbol{k}_1^{(k+1)} + \beta_4(\tau) h^{(k)} \boldsymbol{k}_6, && (4.65)
\end{aligned}$$

with the base polynomials

$$\begin{aligned}
&\beta_0(\tau) := (\tau-1)^2(\tfrac{15}{4}\tau^2+2\tau+1), \quad &&\beta_1(\tau) := \tau(\tau-1)^2(1-\tfrac{35}{8}\tau),\\
&\beta_2(\tau) := 1-\beta_0(\tau), &&\beta_3(\tau) := \tfrac{1}{72}\tau^2(\tau-1)(85\tau-13),\\
&\beta_4(\tau) := \tfrac{125}{18}\tau^2(\tau-1)^2.
\end{aligned} \tag{4.66}$$

4.6 The *RKFSWT* Integrator Algorithm

Algorithm 4.2 The switch-detecting Runge-Kutta-Fehlberg method *RKFSWT*.

Input: t_0, t_f, h, $\boldsymbol{y_0}$, $\boldsymbol{f}(\cdot)$, $\boldsymbol{\sigma}(\cdot)$, Butcher tableau $\left(s^{[p-1]}, s^{[p]}, \boldsymbol{\alpha}, \boldsymbol{B}, \boldsymbol{c}^{[p-1]}, \boldsymbol{c}^{[p]}\right)$, dense output grid $\theta \in [t_0, t_\mathrm{f}]^{n_g}$ with $\theta_i < \theta_j$ for $i < j$, tolerance TOL.
Set $t^{(0)} \leftarrow t_0$, $\boldsymbol{\eta}^{(0)} \leftarrow \boldsymbol{y_0}$, $i \leftarrow 0$, $g \leftarrow 0$.
Precompute $\boldsymbol{k}_1^{(0)} \leftarrow \boldsymbol{f}\left(t^{(0)}, \boldsymbol{\eta}^{(0)}\right)$ (4.4) for solution and sensitivities (4.39a, 4.39c).
while $t^{(k)} < t_\mathrm{f}$ **do**
  repeat
    Compute $\boldsymbol{k}_j^{(k)}$, $j = 2, \ldots, s^{[p]}$ from Eq. (4.4).
    Compute step delta: $\boldsymbol{\Delta\eta}^{[p]} - \boldsymbol{\Delta\eta}^{[p-1]} \leftarrow \sum_{j=1}^{s^{[p]}} (c_j^{[p]} - c_j^{[p-1]})\boldsymbol{k}_j$.
    Compute local error estimate: $\mathrm{est} \leftarrow ||\boldsymbol{\Delta\eta}^{[p]} - \boldsymbol{\Delta\eta}^{[p-1]}||_S$.
    if $\mathrm{est} > \mathrm{TOL}$ **then**
      Reduce step size: $h^{(k)} \leftarrow 0.9\, h^{(k)} \left(\frac{\mathrm{TOL}}{\mathrm{est}}\right)^{\frac{1}{p}}$.
      If $h^{(k)} < \underline{h}$ **then** Stop with error "Local error exceeds TOL for $h^{(k)} < \underline{h}$."
    end if
  until $\mathrm{est} \leqslant \mathrm{TOL}$.
  Perform higher-order step: $\boldsymbol{\eta}^{(k+1)} \leftarrow \boldsymbol{\eta}^{(k)} + \boldsymbol{\Delta\eta}^{[p]}$.
  Precompute $\boldsymbol{k}_1^{(k+1)} \leftarrow \boldsymbol{f}\left(t^{(k)} + h^{(k)}, \boldsymbol{\eta}^{(k+1)}\right)$ (4.4).
  Evaluate switch states $\boldsymbol{\sigma}(t^{(k)} + h^{(k)}, \boldsymbol{\eta}^{(k+1)})$ (4.22).
  for all switches j that changed their sign **do**
    Find $t_\mathrm{s} \in [t^{(k)}, t^{(k)} + h^{(k)}]$ satisfying $\sigma_j(t_s, \boldsymbol{\varrho}(t_\mathrm{s})) = 0$, c.f Algorithm 4.1.
    if $t^{(k)} + h^{(k)} > t_\mathrm{s}$ **then**
      Reduce step length: $h^{(k)} \leftarrow t_\mathrm{s} - t^{(k)}$.
    end if
  end for
  If necessary, compute and perform step $\boldsymbol{\Delta\eta}^{[p]}$ for new step length.
  Save switch states $\boldsymbol{\sigma}(t^{(k)} + h^{(k)}, \boldsymbol{\eta}^{(k+1)})$.
  Compute $\boldsymbol{k}_j^{(k)}$, $j = 2, \ldots, s^{[p]}$ and $\boldsymbol{k}_1^{(k+1)}$ (4.4) for sensitivities (4.39a, 4.39c).
  Compute and perform higher-order step in the sensitivities.
  while $\theta_g \leqslant t^{(k)} + h^{(k)} \wedge g \leqslant n_g$ **do**
    Evaluate a suitable interpolant $\boldsymbol{\varrho}(\theta_g)$, cf. Section 4.5.
    Provide dense output of solution and sensitivities.
    $g \leftarrow g + 1$.
  end while
  if a switch just occurred **then**
    Compute the state jump $\boldsymbol{\Delta}$ (4.25) and the right-hand side jump $\boldsymbol{\delta}$ (4.28).
    Perform the first-order sensitivity updates at discontinuities (4.48, 4.48).
  end if
  Adjust step size: $h^{(k+1)} \leftarrow 0.9\, h^{(k)} \left(\frac{\mathrm{TOL}}{\mathrm{est}}\right)^{\frac{1}{p}}$. Ensure $h^{(k+1)} \in [\underline{h},\ t_\mathrm{f} - t^{(k)}]$.
  $i \leftarrow i + 1$.
end while

Chapter 5

Parameter Estimation

In this chapter we discuss the topic of parameter estimation for models of ordinary differential equations (ODEs). Solving this type of problems necessarily requires a look at the statistical background, as the solution is valid only in the context of the model in use, and the data observed. We motivate least-squares fits against observed data as maximum-likelihood estimators associated with observation errors that satisfy certain statistical assumptions. After having established the connection to nonlinear problems of least squares, we give a brief overview over the constrained Gauß-Newton (CGN) method used to solve problems of this structure. We conclude this chapter with a discussion of confidence estimates for the obtained solutions.
In Appendix A the implementation and usage of the new software tool *QuickFit* for easy parameter estimation based on the optimal control software package *MUSCOD-II* is described.

5.1 Parameter Estimation Problems

The collection of observed data used as input to the parameter estimation task inevitably is subject to random experimental measurement errors, and the observed data may thus be considered as a series of random values. The probability for the experimenter to obtain a specific series of observed data is referred to as the *likelihood* of this data series. In the case of parameter estimation, the probability depends on a set of parameters, and we are interested in determining such a set maximizing the likelihood.

5.1.1 Observation Errors

Consider a series of observed data points

$$\xi_i \in \mathbb{R}, \quad i = 1, \ldots, n_r, \quad n_r \in \mathbb{N}, \tag{5.1}$$

whose observation is subject to measurement errors ε_i

$$\varepsilon_i := \hat{\xi}_i - \xi_i, \quad i = 1, \ldots, n_r, \tag{5.2}$$

that represent the deviation from the true but inaccessible data point $\hat{\xi}_i$. This means that the ξ_i are realizations of random variables Ξ_i, each ξ_i being one sample out of all possible outcomes of the observation process. As a consequence, we may also consider the true observation errors ε_i to be outcomes of random variables E_i. They are distributed according to some a priori unknown probability distribution.
The theory to follow assumes the observation errors E_i to satisfy the following characteristics (Nocedal and Wright [43], von Schwerin [57]):

1. They are free of systematic errors;

   This essentially means that we assume the model $\boldsymbol{y}$ to be valid, and attribute diverging observed data to measurement errors. Most often the correct model is not known in advance, so this usually, and also in case of the powertrain model presented in Chapter 1,

constitutes an iterative process during which the model is continually refined to match the observed data, until eventually no systematic errors remain.

2. They are random variables independent from each other;

   This assumption allows to compute the probability (5.5) as the simple product of the probabilities of the individual data samples.

3. They are attributed with constant variances;

   Which allows for a description of the observed data series in terms of a mean series plus disturbations of constant strength.

4. They are distributed around a mean of zero;

   This is a simplifying criterion only. By introducing a constant offset, distributions around other mean values may be accounted for.

5. They are distributed according to a common probability density function;

   This allows to set up a common objective function for all observed data points. In this chapter, we will mainly discuss normally (Gaussian) distributed measurement errors, resulting in least-squares objectives.

5.1.2 Maximum Likelihood Estimators

Parameter estimation attempts to compare the observations $\boldsymbol{\xi}$ against a model $\boldsymbol{y}$ dependent on a set of parameters $\boldsymbol{x} \in \mathcal{X}$,

$$\boldsymbol{y} : \mathbb{R}^{n_x} \longrightarrow \mathbb{R}^{n_r}, \quad \boldsymbol{x} \mapsto \boldsymbol{y}(\boldsymbol{x}). \tag{5.3}$$

We define the *residual* $r_i(\boldsymbol{x})$ as the discrepancy between model and observation,

$$r_i(\boldsymbol{x}) := y_i(\boldsymbol{x}) - \xi_i, \quad i = 1, \ldots, n_r. \tag{5.4}$$

For a correct model evaluated using the true parameters $\hat{\boldsymbol{x}}$, the measurements errors ε_i equal the residuals r_i. Given some parameter vector $\boldsymbol{x} \in \mathcal{X}$ the probability of observing a particular series $\boldsymbol{\xi}$ of data samples is

$$\mathbb{P}(\boldsymbol{\varepsilon} = \boldsymbol{r} \mid \boldsymbol{x}) = \prod_{i=1}^{n_r} \mathbb{P}(\varepsilon_i = r_i(\boldsymbol{x})) = \prod_{i=1}^{n_r} f(r_i(\boldsymbol{x})), \tag{5.5}$$

where f denotes the probability density function of the observation errors' distribution. Since we actually know the particular series $\boldsymbol{\xi}$ we are working with, we now identify the probability of observing the series $\boldsymbol{\xi}$ with the likelihood of $\boldsymbol{x}$ (cf. Press et al. [50]). A most likely, but not necessarily unique value $\boldsymbol{x}^*$ is referred to as a *maximum likelihood estimate* of the unknowns $\boldsymbol{x}$ given the observed data $\boldsymbol{\xi}$ and the model $\boldsymbol{y}$.

5.1.3 Normally Distributed Observation Errors

For several reasons the most important distribution of observation errors is the normal or Gaussian distribution.

1. When observing typical quantities like mass, acceleration, velocity, or length, one commonly assumes the measurement errors to be distributed according to a normal, or Gaussian, distribution (Bard [1], von Schwerin [57]).

2. The preceding point is stressed by the existence of central-limit theorems that state that under mild assumptions the resulting distribution of many additive, independent random effects approaches the normal distribution.

3. Employing Shannon's measure of information (Shannon [54]), one may show that the normal distribution conveys the least possible amount of a priori information concerning the values that the random variable may assume (Bard [1]).

Definition 5.1. ***Normal Distribution*** $\mathcal{N}_\nu(\mu, \Sigma)$
Let $\boldsymbol{x} \in \mathbb{R}^\nu$ be a random variable of ν degrees of freedom, with a mean $\mathbb{E}(\boldsymbol{x}) =: \boldsymbol{\mu} \in \mathbb{R}^\nu$ and variance-covariance matrix $\operatorname{cov}(\boldsymbol{x}) =: \boldsymbol{\Sigma} \in \mathcal{M}(\nu, \mathbb{R})$. The probability density of the ***multivariate normal distribution*** *is given by the function*

$$f_{\mathcal{N}}(\boldsymbol{x}) := \frac{1}{\sqrt{(2\pi)^\nu \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{5.6}$$

In the one-dimensional case $\nu = 1$ we commonly find the notation $\boldsymbol{\Sigma} := \left[\sigma^2\right]$ leading to $\sqrt{\det \boldsymbol{\Sigma}} = \sigma$, and thus

$$f_{\mathcal{N}}(x) := \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \tag{5.7}$$

Assuming a one-dimensional normal distribution of the observation errors, with mean $\mu = 0$ and a variance σ^2, we may specialize equation (5.5) to

$$\mathbb{P}(\boldsymbol{\varepsilon} = \boldsymbol{r} \mid \boldsymbol{x}) \;=\; \frac{1}{\sqrt{2\pi}^{n_r}} \prod_{i=1}^{n_r} \frac{1}{\sigma_i} \exp\left(-\frac{r_i^2(\boldsymbol{x})}{2\sigma_i^2}\right). \tag{5.8}$$

Maximizing this probability over all $\boldsymbol{x} \in \mathcal{X}$ amounts to finding an $\boldsymbol{x}$ with maximum likelihood. Conveniently, function (5.8) is also referred to as the *likelihood function* L of the estimate $\boldsymbol{x}$. Due to the monotony of the logarithm, it is sufficient to minimize

$$-\log \mathbb{P}(\boldsymbol{\varepsilon} = \boldsymbol{r} \mid \boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{n_r} \frac{r_i^2(\boldsymbol{x})}{\sigma_i^2} + \sum_{i=1}^{n_r} \log \sqrt{2\pi}\sigma_i, \tag{5.9}$$

where the latter part is constant. After introducing residuals weighted by their respective standard deviations σ_i

$$\tilde{r}_i(\boldsymbol{x}) \;:=\; \frac{1}{\sigma_i} r_i(\boldsymbol{x}) \tag{5.10}$$

this problem takes the form of a general nonlinear problem of least squares (5.11) with n_r residuals and n_x unknowns to be determined,

$$\min_{\boldsymbol{x} \in \mathcal{X}} \quad \frac{1}{2} \sum_{i=1}^{n_r} \tilde{r}_i^2(\boldsymbol{x}). \tag{5.11}$$

Assuming a normal distribution of the observation errors therefore relates to determining an estimator $\boldsymbol{x}^*$ minimizing their ℓ_2-norm. A brief overview of the Gauß-Newton method for the efficient solution of this kind of problem is presented in Chapter 3.

5.1.4 Other Error Distributions

It is in general possible to minimize with respect to different ℓ_p-norms, $1 \leqslant p \leqslant \infty$, (see von Schwerin [57]) to obtain maximum likelihood estimators for observation errors distributed according to the respective probability density functions

$$f_p(r_i(\boldsymbol{x})) := \frac{p}{2\alpha_i(\sigma_i)\Gamma\left(\frac{1}{p}\right)} \exp\left(-\left|\frac{r_i(\boldsymbol{x})}{\alpha_i(\sigma_i)}\right|^p\right), \quad i = 1, \ldots, n_r, \tag{5.12}$$

where $\alpha_i(\sigma_i)$ needs to be chosen such that the probability density functions satisfy

$$\int_{\mathbb{R}} f_p(r_i(\boldsymbol{x}))\, dr_i = 1.$$

For $p = 2$ it is easily seen that by setting $\alpha_i(\sigma_i) := \sqrt{2}\sigma_i$ one again obtains the ℓ_2 maximum likelihood estimator. From f_p one derives the ℓ_p objective function just as in the ℓ_2 case demonstrated in the previous section,

$$\min_{\boldsymbol{x}\in\mathcal{X}} \quad \frac{1}{p}\sum_{i=1}^{n_r} |r_i(\boldsymbol{x})|^p \,. \tag{5.13}$$

The case $p = 1$ is of special interest. One minimizes the sum of the absolute deviations of the $\boldsymbol{r}_i$ from the model prediction $\boldsymbol{y}_i$, while putting most emphasis on residuals with *small* observation errors. Under certain assumptions exactly n_x residuals vanish in the solution, which thus interpolates n_x of n_r observations (cf. Bock [6], Körkel et al. [35]).

5.1.5 A Measurement for Goodness-Of-Fit

Aside from the feeling that the model's output matches the observed data quite well when we visually compare them to each other, we have not yet introduced any means of judging the quality of a parameter set. An estimate for this quality can be found in Press et al. [50] and is presented here.

Definition 5.2. ***Upper Incomplete Gamma Function***
The ***upper incomplete Gamma function***[1] $\Gamma(a, z)$ *is defined as*

$$\Gamma(a, z) := \int_z^\infty \exp(-t)\; t^{a-1}\; dt \quad (\textit{here } a \in \mathbb{R}^+, z \in \mathbb{R}). \tag{5.14}$$

It extends the (ordinary) Gamma function such that $\Gamma(a) = \Gamma(a, 0)$.

Theorem 5.3. ***Chi-Square Distribution*** $\mathcal{X}^2_\nu$
Let $\boldsymbol{x} \in \mathbb{R}^\nu$ *be a random variable of* ν *degrees of freedom, whose individual components are normally distributed with unit variance around a mean of zero. Then the sum of squares* $||\boldsymbol{x}||_2^2$ *is a* ***chi-square distributed random variable****, and the probability for it to exceed a threshold value of* χ^2 *is*

$$\overline{\mathbb{P}}_{\mathcal{X}}(\chi^2 \mid \nu) := \mathbb{P}(||\boldsymbol{x}||_2^2 \geqslant \chi^2) = \Gamma\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right)\Gamma^{-1}\left(\frac{\nu}{2}\right). \tag{5.15}$$

Proof. Compute by transformation to ν-dimensional polar coordinates (cf. Königsberger [34]) the integral

$$\mathbb{P}(||\boldsymbol{x}||_2^2 \geqslant \chi^2) = \int_{||\boldsymbol{x}||_2^2 \geqslant \chi^2} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\,||\boldsymbol{x}||_2^2\right)\; d\boldsymbol{x}$$

and apply Eq. (5.14). □

It is obvious that the least-squares objective is, by way of its dependency on the residuals r_i, a random variable of n_r degrees of freedom. In addition, the residuals $\tilde{r}_i$ (5.10) divided by their standard deviations have unit variance. Having been adjusted to maximize the likelihood, the components of $\boldsymbol{x}$, however, are no longer statistically independent of each other but satisfy a common constraint. Thus the presented definition cannot be directly applied.
For linear models $\boldsymbol{y}$ it is nonetheless possible to derive the statistical distribution of $\boldsymbol{x}^*$ analytically, and it turns out to be the chi-square distribution of $n_r - n_x$ degrees of freedom. Furthermore, it is quite common to assume this relation to hold even for models not strictly linear in $\boldsymbol{x}$ (cf. Press et al. [50]). Given a correct model and a maximum likelihood estimator $\boldsymbol{x}^*$ found from the solution of problem (5.11), the probability for the least-squares objective to exceed *by chance* a value of

$$\chi^2 := \sum_{i=1}^{n_r} \tilde{r}_i^2(\boldsymbol{x}^*)$$

[1] `GAMMA(a,z)` in *Maple V*, `gammainc(a,z)` in *MATLAB*. Press et al. [50] present a C implementation.

equals, by this assumption,

$$Q := \overline{\mathbb{P}}_{\mathcal{X}}(\chi^2 \mid n_r - n_x). \tag{5.16}$$

The larger that probability is, the more significance bears the solution $\boldsymbol{x}^*$ we found. In contrast, with $\overline{\mathbb{P}}_{\mathcal{X}}$ approaching zero, virtually *any* choice of the estimator $\boldsymbol{x}$ would be as good as the $\boldsymbol{x}^*$ we identified from problem (5.11). This may be regarded as an indicator of a wrong model $\boldsymbol{y}$ being in use.
Finally, note that the applicability of this measure for goodness-of-fit requires, besides the prerequisites stated in the introductory section, that the variances σ_i^2 of the residuals are *known in advance*. If we used estimates of the σ_i, an excellent goodness-of-fit might only mean that we severely overestimated the variances.

5.1.6 Regularization

Often it is the case that some a priori information about the values of the parameters $\hat{\boldsymbol{x}}$ to be estimated is available. In that case it is obviously desirable to be able to exploit that information in the parameter estimation method used.

1. The vector $\boldsymbol{\sigma}$ could be a vector of standard deviations from the mean parameter values $\boldsymbol{x}$. This would lead us to an assumed $\mathcal{N}_{n_x}(\overline{\boldsymbol{x}}, \mathrm{diag}(\boldsymbol{\sigma})^2)$ distribution of the parameter.
2. It is also very convenient to specify upper and lower bounds for the parameters, in which case $\boldsymbol{\sigma}$ is the maximum deviation allowed from $\overline{\boldsymbol{x}}$. We obtain a uniform distribution of $\boldsymbol{x}$ over the hypercube $[\overline{\boldsymbol{x}} - \boldsymbol{\sigma}, \overline{\boldsymbol{x}} + \boldsymbol{\sigma}] \subset \mathbb{R}^{n_x}$.

We refer to the respective probability density function as the *prior density function* $f_0(\boldsymbol{x})$ of the unknown parameters. In both cases, the assumed probability distributions convey the least possible information according to Shannon [54].
The presence of a priori information may be considered a case of conditional probability. Along the lines of Bard [1] we use Bayes' theorem to derive the posterior probability distribution given prior information about the unknowns by way of f_0 (5.21), and about the data by way of the likelihood function L (5.8).

Theorem 5.4. ***Bayes' Theorem*** *(T. Bayes, 1763)*
Let A and B be two events whose probabilities $\mathbb{P}(A)$ and $\mathbb{P}(B) \neq 0$. The conditional probability $\mathbb{P}(A|B)$ that A occurs given that B has occurred equals

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A) \cdot \mathbb{P}(A)}{\mathbb{P}(B)}. \tag{5.17}$$

Proof. Straightforward from the definition of conditional probability, see, e.g., Bard [1]. □

In our case we consider the two events *A:* That the true value $\hat{\boldsymbol{x}}$ of the unknowns falls within the hypercube of size $d\boldsymbol{x}$ centered at $\boldsymbol{x}$; and *B:* That the true value $\hat{\boldsymbol{\xi}}$ of the measurements falls within a hypercube of size $d\Xi$ centered at $\boldsymbol{\xi}$. By definition we have

$$\mathbb{P}(A) = f_0(\boldsymbol{x})\, d\boldsymbol{x}, \qquad \mathbb{P}(B|A) = L(\boldsymbol{x})\, d\Xi, \tag{5.18}$$

and obtain the value of $\mathbb{P}(B)$ by summing $\mathbb{P}(B|A) \cdot \mathbb{P}(A)$ over all possible A,

$$\mathbb{P}(B) = \left(\int_{\mathbb{R}^{n_x}} L(\boldsymbol{x}) f_0(\boldsymbol{x})\, d\boldsymbol{x} \right)\, d\boldsymbol{\Xi}. \tag{5.19}$$

Applying Bayes' theorem now yields the likelihood

$$P(A|B) = L(\boldsymbol{x})\, d\Xi \cdot f_0(\boldsymbol{x})\, d\boldsymbol{x} \cdot \left(\int_{\mathbb{R}^{n_x}} L(\boldsymbol{x}) f_0(\boldsymbol{x})\, d\boldsymbol{x} \right)^{-1}. \tag{5.20}$$

To incorporate both of the above types of a priori information, we assume a non-truncated normal distribution $\mathcal{N}_{n_x}(\overline{\boldsymbol{x}}, \boldsymbol{T})$ of the parameters to be estimated, and impose explicit inequality constraints on the unknowns $\boldsymbol{x}$. The treatment of constrained nonlinear least-squares problems is discussed in Chapter 3.

With the covariance matrix being $\boldsymbol{T} = \operatorname{diag}(\boldsymbol{\tau})^2$, the prior density function f_0 is

$$f_0(x_i) := \frac{1}{\sqrt{2\pi}\tau_i} \exp\left(-\frac{1}{2}\left(\frac{x_i - \overline{x}_i}{\tau_i}\right)^2\right) \tag{5.21}$$

Plugging (5.21) into (5.20) and ignoring the integral factor that is independent of $\boldsymbol{x}$ we obtain

$$-\log \mathbb{P}(A|B) = \frac{1}{2}\sum_{i=1}^{n_r} \frac{r_i^2(\boldsymbol{x})}{\sigma_i^2} + \frac{1}{2}\sum_{i=1}^{n_x} \frac{(x_i - \overline{x}_i)^2}{\tau_i^2} + \text{const.} \tag{5.22}$$

up to a common factor. Comparing this to Eq. (5.9) we are led to include the additional residuals

$$r_{n_r+i}(\boldsymbol{x}) := \frac{1}{2}\left(\frac{x_i - \overline{x}_i}{\tau_i}\right)^2, \quad i = 1, \ldots, n_x. \tag{5.23}$$

in the least-squares parameter estimation problem. We may extend this approach even further to include a priori covariance information in the problem. If we extend the assumed a priori distribution to $\boldsymbol{T} = [\tau_{ij}]_{ij}$ being the full variance-covariance matrix associated with $\overline{\boldsymbol{x}}$, we obtain the following additional least-squares residuals instead of those given in (5.23):

$$r_{n_r+i}(\boldsymbol{x}) := \frac{1}{2}\sum_{j=1}^{n_x} \frac{(x_i - \overline{x}_j)^2}{\tau_{ij}}, \quad i = 1, \ldots, n_x. \tag{5.24}$$

5.2 Uncertainty Estimates and Confidence Areas

A practical method for the estimation of parameters not only requires the solution of the underlying least-squares problem, but also asks for an analysis of the quality of the obtained maximum likelihood estimate. We are especially interested in the sensitivity of this estimate with respect to observation errors.

5.2.1 Covariance of the Solution

Let $\boldsymbol{f}$ denote the least-squares objective

$$\boldsymbol{f}(\boldsymbol{x}) := \sum_{i=1}^{n_r} \tilde{r}_i^2(\boldsymbol{x}). \tag{5.25}$$

Theorem 5.5. ***Covariance Matrix of the Solution*** *(Bard [1], Bock [6], von Schwerin [57])*
Under the assumptions of Section 5.1, the generalized Gauß-Newton method's solution $\boldsymbol{x}^* = -\boldsymbol{J}^+\boldsymbol{f}$ is a normally distributed random variable. The true solution is its expectation value, and its ***covariance matrix*** *is computed as*

$$\operatorname{cov} \boldsymbol{x}^* = \boldsymbol{J}^+(\boldsymbol{x}^*) \begin{bmatrix} \boldsymbol{I}_{n_r} & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{J}^+(\boldsymbol{x}^*)^{\mathsf{T}}. \tag{5.26}$$

If no restrictions are active, we find from an explicit representation of the generalized inverse J^+ that the covariance matrix may simply be obtained as the inverse of the Hessian approximation

$$\operatorname{cov} \boldsymbol{x}^* = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*)^{\mathsf{T}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*)\right)^{-1}. \tag{5.27}$$

5.2.2 An Estimate for the Common Variance Factor

If estimates $\tilde{\boldsymbol{\Sigma}}$ of the residuals' variances $\boldsymbol{\Sigma}$ have been used, or if the variances are unknown at all (and we used, e.g., $\tilde{\boldsymbol{\Sigma}} = \tau \boldsymbol{I}$), the common variance factor β^2 of the observed data's variances is unknown as well. An independent estimate of the common variance factor is available as $\tilde{\beta}^2$ (see Bard [1]),

$$\tilde{\beta}^2 := \frac{||\boldsymbol{f}(\boldsymbol{x}^*)||_2^2}{n_r - (n_x - n_c)}, \tag{5.28}$$

where n_c is the number of constraints active in the solution of problem (5.13), corresponding to equality constraints in problem (3.24a),

$$n_c := n_g + \hat{n_h}. \tag{5.29}$$

Estimates σ_i^2 close to the true variances yield a common factor of $\tilde{\beta}^2 \approx 1$. In any case, the unbiased covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ of the estimate $\boldsymbol{x}^*$ is then obtained from the biased one (5.27) as

$$\boldsymbol{\Sigma}_{\boldsymbol{x}} := \beta^2 \boldsymbol{J}^+(\boldsymbol{x}^*) \begin{bmatrix} \boldsymbol{I}_{n_r} & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{J}^+(\boldsymbol{x}^*)^\mathsf{T}. \tag{5.30}$$

5.2.3 Confidence Area of the Solution

Having obtained an optimal set of parameters given a model and observed data, we are also interested in its statistical significance. A suitable tool is the specification of confidence areas, centered around the estimated parameter value, and indicating with what probability the real-world parameter falls into these areas.
The area of confidence of the solution $\boldsymbol{x}^*$ of (5.13) may be represented by the indifference region $\mathcal{G}_\mathrm{N}$ of the least-squares objective (see Bock [6])

$$\mathcal{G}_\mathrm{N}(\boldsymbol{x}^*, \alpha) := \Big\{ \boldsymbol{x} \in \mathcal{F} \;\Big|\; \boldsymbol{g}(\boldsymbol{x}) = 0 \;\wedge\; ||\boldsymbol{f}(\boldsymbol{x})||_2^2 - ||\boldsymbol{f}(\boldsymbol{x}^*)||_2^2 \leqslant \gamma(\alpha) \Big\}. \tag{5.31}$$

We are looking for an appropriate indifference level $\gamma \geqslant 0$ determined by the desired level of uncertainty $\alpha \in [0, 1]$, e.g., $\alpha = 0.05$ for the area of 95% confidence. Since the estimate $\boldsymbol{x}^*$ is truly not a random variable, this statement is to be interpreted as follows: The region $\mathcal{G}_\mathrm{N}$ is defined in such a way that only a fraction α of the regions $\mathcal{G}_\mathrm{N}$ resulting from multiple independent observations of the same scenario, which are subject to *random* observation errors, will *not* contain the true value $\hat{\boldsymbol{x}}$,

$$\mathbb{P}\big(\hat{\boldsymbol{x}} \in \mathcal{G}_\mathrm{N}(\boldsymbol{x}^*, \alpha)\big) = 1 - \alpha. \tag{5.32}$$

It can be shown (see, e.g., Bard [1]) that by Taylor expansion up to second-order terms the objective function's indifference region takes the approximate form

$$||\boldsymbol{x} - \boldsymbol{x}^*||_{\Sigma_x^{-1},2}^2 = (\boldsymbol{x} - \boldsymbol{x}^*)^\mathsf{T} \boldsymbol{\Sigma}_{\boldsymbol{x}}{}^{-1} (\boldsymbol{x} - \boldsymbol{x}^*) \leqslant \gamma(\alpha). \tag{5.33}$$

For errors normally distributed around zero, the resulting estimates are normally distributed as well, with a mean of $\boldsymbol{x}^*$. We may thus consider the above scaled norm to be distributed as $\mathcal{X}^2$ with n_x degrees of freedom. Taking the equality constraints $\boldsymbol{g}(\boldsymbol{x})$ into account, the number of degrees of freedom reduces to $n_x - n_c$.

Definition 5.6. ***Quantile of a Distribution***
Let $\mathbb{P}_\mathcal{S}(z)$ be the cumulative probability density function of a statistical distribution $\mathcal{S}$ with ν degrees of freedom. The ***quantile*** *$\mathcal{S}_\nu(\alpha)$ is defined for $0 \leqslant \alpha \leqslant 1$ as*

$$\mathcal{S}_\nu(\alpha) := \inf\big\{ z \mid \mathbb{P}_\mathcal{S}(z) \geqslant \alpha \big\}. \tag{5.34}$$

It specifies the scalar value below which one finds a fraction α of all cases.

Assuming the estimate's covariance $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ is known, we obtain from this definition

$$\gamma(\alpha) := \mathcal{X}^2_{n_x - n_c}(1-\alpha). \tag{5.35}$$

Since the nonlinear confidence area $\mathcal{G}_{\mathrm{N}}$ (5.31) is generally hard to compute, an approximation may be obtained by linearization around $\boldsymbol{x}^*$ and yields the following confidence ellipsoid restricted to the feasible set

$$\begin{aligned}
\mathcal{G}_{\mathrm{L}}(\boldsymbol{x}^*, \alpha) &:= \Big\{ \boldsymbol{x} \in \mathcal{F} \;\Big|\; \boldsymbol{g}(\boldsymbol{x}^*) + \nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*) = 0 && \text{(5.36a)} \\
&\qquad \wedge\ ||\boldsymbol{f}(\boldsymbol{x}^*) + \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*)||_2^2 - ||\boldsymbol{f}(\boldsymbol{x}^*)||_2^2 \leqslant \gamma(\alpha) \Big\} \\
&= \Big\{ \boldsymbol{x}^* + \boldsymbol{\Delta x} \;\Big|\; \boldsymbol{\Delta x} = -\boldsymbol{J}^+ \begin{bmatrix} \boldsymbol{\Delta y} \\ \boldsymbol{0} \end{bmatrix},\ ||\boldsymbol{\Delta y}||_2^2 \leqslant \gamma(\alpha) \Big\}. && \text{(5.36b)}
\end{aligned}$$

For a proof of this equality, we refer to Bock [6].

If the observation errors' variances Σ are unknown, i.e., if they have been estimated, the common variance factor estimate $\tilde{\beta}^2$ needs to be taken into account. The least-squares residual objective $||\boldsymbol{f}(\boldsymbol{x}^*)||_2^2$ of (5.28) depends on the estimate $\boldsymbol{x}^*$, but is known to be distributed independently of that estimate. The distribution of β^2 is thus $\mathcal{X}^2$ with $n_r - (n_x - n_c)$ degrees of freedom (see Bard [1]). The resulting objective indifference region radius

$$\begin{aligned}
||\boldsymbol{x} - \boldsymbol{x}^*||^2_{\left(\tilde{\beta}^2 \Sigma_x\right)^{-1}, 2} &= (\boldsymbol{x} - \boldsymbol{x}^*)^{\mathsf{T}} \left(\tilde{\beta}^2 \boldsymbol{\Sigma}_{\boldsymbol{x}}\right)^{-1} (\boldsymbol{x} - \boldsymbol{x}^*) && \text{(5.37a)} \\
&= \big(n_r - (n_x - n_c)\big) \frac{||\boldsymbol{x} - \boldsymbol{x}^*||^2_{\Sigma_x^{-1}, 2}}{||\boldsymbol{f}(\boldsymbol{x})||_2^2} && \text{(5.37b)}
\end{aligned}$$

is thus F-distributed with $n_x - n_c$ and $n_r - (n_x - n_c)$ degrees of freedom if divided by $n_x - n_c$, as can be seen from the following theorem.

Theorem 5.7. ***Fisher's F-Distribution*** $\mathcal{F}_{\nu_1, \nu_2}$
Let $\boldsymbol{x} \in \mathbb{R}^{\nu_1}, \boldsymbol{y} \in \mathbb{R}^{\nu_2}$ be independent random and normally distributed variables, with zero mean and unit variance. Then the quotient

$$z := \left(\frac{1}{\nu_1} \sum_{i=1}^{\nu_1} x_i^2 \right) \left(\frac{1}{\nu_2} \sum_{i=1}^{\nu_2} y_i^2 \right)^{-1} \tag{5.38}$$

is an ***F-distributed scalar random variable*** *with ν_1 and ν_2 degrees of freedom.*

We obtain

$$\gamma(\alpha) := (n_x - n_c) \mathcal{F}_{n_x - n_c, n_r - (n_x - n_c)}(1-\alpha). \tag{5.39}$$

However, it should be noted that when $\nu_1 \gg \nu_2$, the $\mathcal{F}_{\nu_1, \nu_2}$-distribution approaches the $\mathcal{X}^2_{\nu_1}$-distribution again as a result of central limit theorems. Here $\nu_1 = n_r$ and $\nu_2 = n_x$, so this will, for example, very frequently be the case when estimating parameters by fitting the discretized solution of a parametrized initial-value problem (IVP).

5.3 Initial Value Problems for Parameter Estimation

While in general the model $\boldsymbol{y}(\boldsymbol{x})$ may create data points $\boldsymbol{y}_i$ from the parameter set $\boldsymbol{x}$ by way of many different mechanisms (one may think of systems of linear or nonlinear equations, neural networks, etc.) we are especially interested in models composed from a system of ordinary differential equations (ODEs).

5.3.1 Problem Formulation

We specify the model $\boldsymbol{y}(\boldsymbol{x})$ in terms of an initial value problem (IVP) (5.40) on the time horizon $\mathcal{T} := [t_0, t_\mathrm{f}] \subset \mathbb{R}$,

$$\frac{d\boldsymbol{y}}{dt}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}) = \boldsymbol{f}(t, \boldsymbol{y}(t), \boldsymbol{p}), \quad t \in \mathcal{T}, \tag{5.40a}$$

$$\boldsymbol{y}(t_0) = \boldsymbol{y_0}, \tag{5.40b}$$

and compose the vector $\boldsymbol{x}$ of model parameters from the IVP's global parameters $\boldsymbol{p} \in \mathcal{P}$ and initial values $\boldsymbol{y_0} \in \mathcal{Y}$,

$$\boldsymbol{x} := \begin{bmatrix} \boldsymbol{y_0} \\ \boldsymbol{p} \end{bmatrix}. \tag{5.41}$$

In addition, it is favorable to put constraints on the individual parameters p_i and initial values $y_{0,i}$,

$$\boldsymbol{y_0} \in \left[\underline{\boldsymbol{y}}_{\boldsymbol{0}}, \overline{\boldsymbol{y}}_{\boldsymbol{0}}\right] =: \mathcal{Y} \subset \mathbb{R}^{n_y}, \tag{5.42a}$$

$$\boldsymbol{p} \in \left[\underline{\boldsymbol{p}}, \overline{\boldsymbol{p}}\right] =: \mathcal{P} \subset \mathbb{R}^{n_p}. \tag{5.42b}$$

Badly conditioned parameters that cannot be properly estimated from the available data set will quickly hit one of these bounds, and may subsequently be eliminated from the parameter estimation problem by setting them to a reasonable fixed value (cf. Bock [6]). This formulation easily includes the case of fixed parameters or known initial values by setting $\underline{p}_i = \overline{p}_i$ or similarly $\underline{y}_{0,i} = \overline{y}_{0,i}$ for the appropriate components of the bounds vectors.

From the ODE model's behavior dependent on $(\boldsymbol{y_0}, \boldsymbol{p})$ we compute output trajectories

$$z_i : \mathcal{T} \times \mathbb{R}^{n_y} \longrightarrow \mathbb{R}, \quad t \mapsto z_i\left(t, \boldsymbol{y}(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p})\right), \quad i = 1, \ldots, n_z, \tag{5.43}$$

to be fitted against trajectories of observed data

$$\xi_i : \mathcal{T} \longrightarrow \mathbb{R}, \quad t \mapsto \xi_i(t). \tag{5.44}$$

For a given parameter set $[\boldsymbol{y_0}, \boldsymbol{p}]$ the computation of the residual vector $\boldsymbol{r}$ (5.4) involves the solution of the IVP (5.40) on the time horizon $\mathcal{T}$. Due to imprecise initial values, inherently unstable ODE systems, as well as due to truncation and round-off errors incurred by the ODE solver method, it is likely that the simple integration of the ODE system over the time horizon $\mathcal{T}$ might lead to inexact solutions, or even that a numerical solution can not be obtained at all (Bock [6]). We therefore solve the IVP using the multiple shooting method due to Bock and Plitt [7], as described in Chapter 6 in the context of optimal control problems.

5.3.2 Discretization

As of now we would attempt to fit trajectories $z_i(t)$ against $\xi_i(t)$, aiming at minimizing the $\mathscr{L}_2$ semi-norm

$$||z_i - \xi_i||^2_{\mathscr{L}_2} = \int_{\mathcal{T}} \Big(z_i\left(\boldsymbol{y}\left(t;\ t_0, \boldsymbol{y_0}, \boldsymbol{p}\right)\right) - \xi_i(t)\Big)^2\, dt. \tag{5.45}$$

The formulation of such semi-infinite parameter estimation problems with continuous observed data is delicate, and the convergence theory for the least-squares method presented in Chapter 3 is valid for finite sets of unknowns and observations only. Thus, the trajectories z_i get discretized using a suitably chosen discretization grid (5.46) fine enough to represent the observation's features we are interested in.

$$t_0 = t_1 < \ldots < t_{n_t} = t_\mathrm{f}, \quad n_t \in \mathbb{N}. \tag{5.46}$$

Depending on the measurement device, the trajectories probably will be sampled and discretized during recording (polling, or event-driven recording of data samples), and a continuous representation will not be available anyway. From the discretization we obtain series of sampled data points

$$\boldsymbol{\xi}_i \in \mathbb{R}^{n_t}, \quad i = 1, \ldots, n_z, \tag{5.47}$$

the dimension $n_t \in \mathbb{N}$ being the (possibly very large) number of sampled points. In this discretized form the $n_r = n_z \cdot n_t$ residuals are computed as

$$r_{i,j}(\boldsymbol{y_0}, \boldsymbol{p}) := z_i\left(\boldsymbol{y}\left(t_j;\, t_0, \boldsymbol{y_0}, \boldsymbol{p}\right)\right) - \xi_{i,j}, \tag{5.48}$$

and we end up with the following discrete least-squares minimization problem

$$\min_{\boldsymbol{y_0}, \boldsymbol{p}} \quad \sum_{i=1}^{n_z} \sum_{j=1}^{n_r} r_{i,j}^2(\boldsymbol{y_0}, \boldsymbol{p}) \tag{5.49a}$$

$$\text{s.t.} \quad \frac{d\boldsymbol{y}}{dt}(t;\, t_0, \boldsymbol{y_0}, \boldsymbol{p}) = \boldsymbol{f}(t, \boldsymbol{y}(t), \boldsymbol{p}), \quad t \in \mathcal{T}, \tag{5.49b}$$

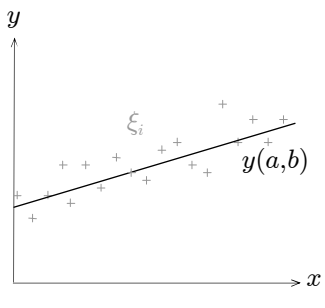
$$\boldsymbol{y}(t_0) = \boldsymbol{y_0}. \tag{5.49c}$$

5.3.3 Scaling and Weighting

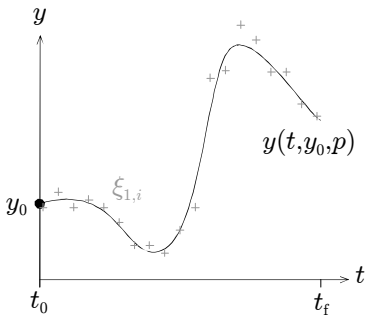
To put emphasis on certain time-local features of the trajectories $z_i(t)$ it may be desirable to put weights $\omega_i(t_j)$ on the samples $\xi_{i,j}$ resulting from the discretization.

$$\min_{\boldsymbol{y_0}, \boldsymbol{p}} \quad \sum_{i=1}^{n_z} \sum_{j=1}^{n_r} \omega_i(t_j) r_{i,j}^2(\boldsymbol{y_0}, \boldsymbol{p}). \tag{5.50}$$

In addition, the weighting functions ω_i should also be used to compensate for different scales of the trajectories. For example, consider trajectories z_1 and z_2, where z_1 might represent engine revolutions per second and samples roughly range from 10^3 [min^{-1}] to 10^4 [min^{-1}]. Trajectory z_2 might represent acceleration samples well below 10^1 [$\mathsf{m/s^2}$]. It would be misleading to set up an unscaled parameter estimation problem on this type of observed data, as even large discrepancies of the fit against the second trajectory would yield only small residuals compared to those of the first trajectory.



(a) Fitting a linear model $y(x) = ax + b$ to observed data points ξ_i.



(b) Fitting an ODE model $y(t, y_0, \boldsymbol{p})$ to a discretized trajectory $\xi_{1,i}$.

5.4 Powertrain Parameter Estimation

In this section we turn to the guiding example for the first time, and present parameter estimation results obtained by application of the methods presented in Chapter 5 to data observed in a *Mercedes C-Class* on the test tracks of *Stuttgart-Untertürkheim*. It will be seen that the powertrain model described in the introductory Chapter 1 allows for a very accurate and satisfying representation of the powertrain oscillation phenomena.
A look at the eigenvalues of a suitably linearized system enables us to identify one harmonic oscillator whose contributions to the oscillations under investigation are negligible. As a consequence we construct a smaller powertrain model that equally well represents the behavior seen from the observed data series, without even requiring a re-run of the parameter estimation procedure. This smaller model is non-stiff and may be integrated using an explicit integration method such as one of the explicit Runge-Kutta schemes presented in Chapter 4.

5.4.1 Parameter Estimation Scenario and Setup

Fig. 5.1 shows the observed input motor torque that was used to drive the ODE model and defines the parameter estimation scenario. On the time horizon of 37.0 seconds we can clearly distinguish four separate acceleration phases, each one taking place at increasing velocity and engine speed.

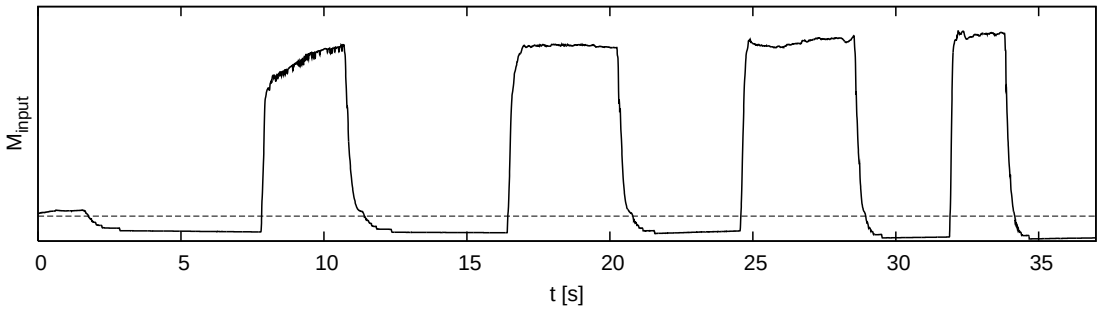


Fig. 5.1: Input motor torque M_{mot} defining the parameter estimation scenario.

For non-disclosure reasons, all graphs of observed and simulated data remain unlabeled on the ordinate when presenting results for the *DaimlerChrysler* powertrain models.

Initial Values

Initial values for the ODE system's angular velocities are computed from observed data at $t = 0$ s. The powertrain's angular torsions $\Delta\varphi_*$ cannot easily be initialized from observed data and are set to zero. This assumption is not too far off, since recording of observed data used for the parameter estimation scenario starts with a neutral motor torque that does not actively accelerate the powertrain.

State	Value	Unit
$\omega_{\mathrm{dmf},1}$	$2.205 \cdot 10^2$	rad/s
$\omega_{\mathrm{dmf},2}$	$1.279 \cdot 10^2$	rad/s
$\Delta\varphi_{\mathrm{dmf}}$	$0.000 \cdot 10^0$	rad
ω_{ad}	$4.516 \cdot 10^1$	rad/s

State	Value	Unit
$\Delta\varphi_{\mathrm{ad}}$	$0.000 \cdot 10^0$	rad
ω_{wh}	$4.594 \cdot 10^1$	rad/s
$\Delta\varphi_{\mathrm{wh}}$	$0.000 \cdot 10^0$	rad
v_{car}	$1.434 \cdot 10^1$	m/s

Tab. 5.1: Initialization of the powertrain model IVP for parameter estimation.

Model Parameters

Parameters are initialized to coarse estimates along with relatively lax bounds, allowing for parameter variations over two orders of magnitude. As long as physically meaningful, a bound gets shifted and adapted when hit during the process of parameter estimation.

Model Outputs

Output residuals are divided by scaling factors to be found in Tab. 5.2 in order to yield approximately equal ℓ_2 residuals on the first iteration. Residuals of selected outputs are weighted by the observed trajectory's curvature in the neighborhood of the residual. This puts emphasis on the oscillation phenomena found in the observed data, whose accurate representation is the prior aim of the parameter estimation process. The side shaft torque signal M_{ss} is excluded from this setup since the level of this torque signal shows a noticeable offset when compared to observed data. This is blamed on an inaccurately tuned measurement device. As clearly visible from the graphs on page 66, the acceleration signal is far too noisy to serve as a good reference for the acceleration computed from the ODE model. Thus the model output a_{car} is also excluded from this setup.

Model Output Signal Delays

Careful comparison of the ODE model's behavior against the measurement data reveals that the velocity signals got recorded with noticeable delays during observation on the test track. These delays may be blamed on low data transmission rates combined with long signal paths, and the usage of a prioritized data transfer scheme in the car. We account for them by introduction of additional signal delay parameters into the parameter estimation problem, that allow the time shift of the observed data to be estimated. This enabled us to significantly improve the obtained results. Signal delays are initialized to zero, with an allowed range of one second.

Model Output	n_{mot}	M_{cs}	n_{cs}	M_{ss}	n_{ss}	v_{wh}	v_{car}	a_{car}
Used in Objective	×	×	×	–	×	×	×	–
Curvature Weighting	×	×	×	–	×	–	–	–
Delays Expected	–	–	–	–	–	×	×	–
Scale	3.500	11.000	0.030	3.500	1.900	0.550	0.005	0.005

Tab. 5.2: Usage and curvature weighting of the powertrain model outputs.

5.4.2 Parameter Estimation Results

This section contains the signal delays, model parameters, and model output residuals we found using the new software tool *QuickFit* (appendix A).

Output	Delay	Standard Deviation		Unit
v_{wh}	$-3.124415 \cdot 10^{-2}$	$-4.100000 \cdot 10^{-3}$	13.12%	s
v_{car}	$-2.235588 \cdot 10^{-3}$	$-2.820410 \cdot 10^{-3}$	126.16%	s

Tab. 5.3: Estimated signal delays in observed data for the powertrain model.

Output v_{car} shows no traceable delay, the estimated delay can be assumed as zero within the estimated standard deviation. The estimated delay of the output signal v_{wh} falls within the expected bounds of 20 to 40 milliseconds.

Parameter	Std. Dev.	Unit
d_{dmf}	27.06%	Nm s/°
$d_{\mathrm{dmf},2}$	8.72%	Nm s/°
$J_{\mathrm{dmf},1}$	2.46%	kg m^2
$J_{\mathrm{dmf},2}$	0.39%	kg m^2
c_{cs}	15.52%	Nm/°
d_{cs}	52.11%	Nm/°
d_{ad}	24.39%	Nm s/°
J_{ad}	7.60%	kg m^2
c_{ss}	4.23%	Nm/°

Parameter	Std. Dev.	Unit
d_{ss}	53.00%	Nm/°
p_{ss}	5.52%	°
d_{wh}	8.09%	Nm s/°
J_{wh}	6.03%	kg m^2
μ_{roll}	0.17%	–
r_{tyre}	0.06%	m
A_{car}	1.05%	m^2
m_{car}	0.05%	kg

Tab. 5.4: Estimated uncertainties for the optimal parameter set of the powertrain model, given in percent of the estimated parameter values. For non-disclosure reasons, the actual values are omitted.

We were able to identify most parameter values with good confidence, e.g., with an estimated standard deviation of around 5% of the nominal parameter value. We found, however, that the uncertainty in the damping coefficients such as d_{dmf}, d_{cs}, d_{ad}, and d_{ss}, is noticeably higher. We could not obtain an estimate for the parameter $d_{\mathrm{dmf},1}$, which hit the lower bound of zero. Parameters that can be directly compared to real-world measurements, such as m_{car} or r_{tyre}, stay close to the expected values. The deviation of A_{car} from the expected value is attributed to actual deviations of the parameters ϱ_{air}, c_{w}, and β (see Eq. (2.11c)), which were encoded as fixed values and not exposed as variable parameters in the present model.

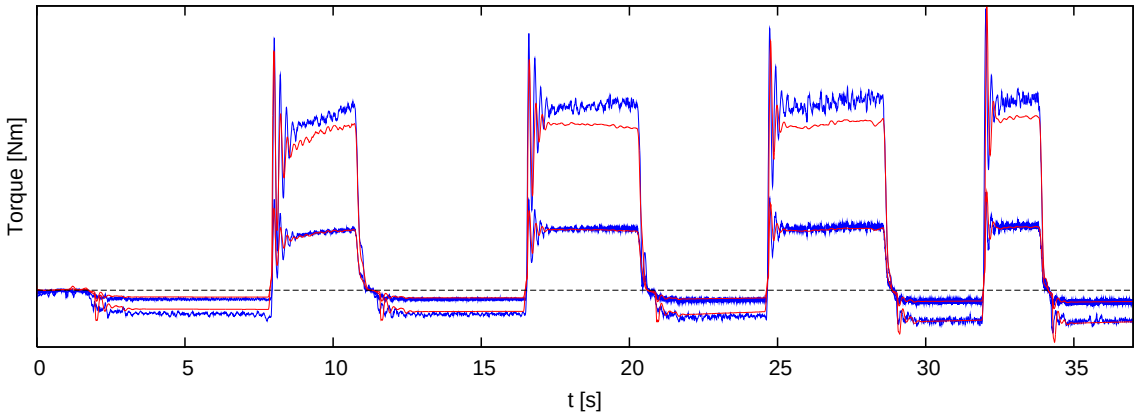
	Samples	Weighted ℓ_2		Unweighted ℓ_2	
	[Hz]	Overall	per Sample	Overall	per Sample
n_{mot}	100	$8.6168 \cdot 10^{1}$	$7.0741 \cdot 10^{-1}$	$1.6691 \cdot 10^{2}$	$1.3703 \cdot 10^{0}$
M_{cs}	1000	$1.1857 \cdot 10^{2}$	$9.7342 \cdot 10^{-1}$	$4.0551 \cdot 10^{2}$	$3.3291 \cdot 10^{0}$
n_{cs}	1000	$1.1035 \cdot 10^{2}$	$9.0590 \cdot 10^{-1}$	$3.8731 \cdot 10^{2}$	$3.1797 \cdot 10^{0}$
M_{ss}	1000	$1.0913 \cdot 10^{2}$	$8.9593 \cdot 10^{-1}$	$1.2032 \cdot 10^{3}$	$9.8776 \cdot 10^{0}$
n_{ss}	1000	$1.0526 \cdot 10^{2}$	$8.6422 \cdot 10^{-1}$	$5.7897 \cdot 10^{1}$	$4.7532 \cdot 10^{-1}$
v_{wh}	50	$1.3315 \cdot 10^{2}$	$1.0931 \cdot 10^{0}$	$6.1253 \cdot 10^{-1}$	$5.0286 \cdot 10^{-3}$
v_{car}	50	$9.5147 \cdot 10^{1}$	$8.0968 \cdot 10^{-1}$	$4.9476 \cdot 10^{-1}$	$3.7245 \cdot 10^{-3}$
a_{car}	1000	$1.0677 \cdot 10^{2}$	$8.7660 \cdot 10^{-1}$	$3.2033 \cdot 10^{0}$	$2.6298 \cdot 10^{-2}$

Tab. 5.5: Residuals of the optimal powertrain model parametrization.

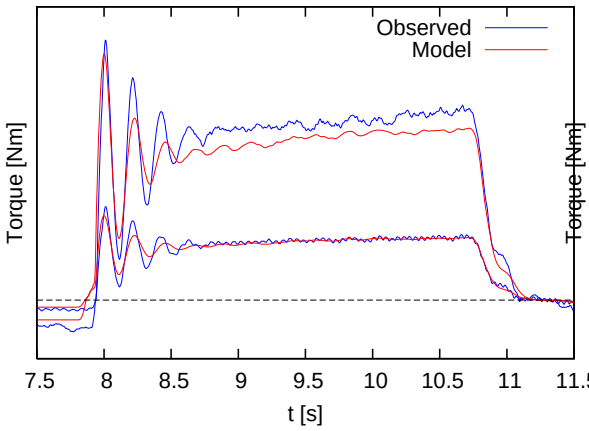
Pages 64 to 67 compare the model's outputs using the obtained parameter set against the series of observed data from the test track. The model obviously contains no systematic errors; the presented average residuals per sample (given in units of the output signal) are negligible. All graphs show an extremely satisfying match of the simulated trajectories against the observed data, the already mentioned exception being the side shaft torque signal M_{ss}. The results constitute a considerable improvement over those presented by Stelzer [55].

Powertrain Torques

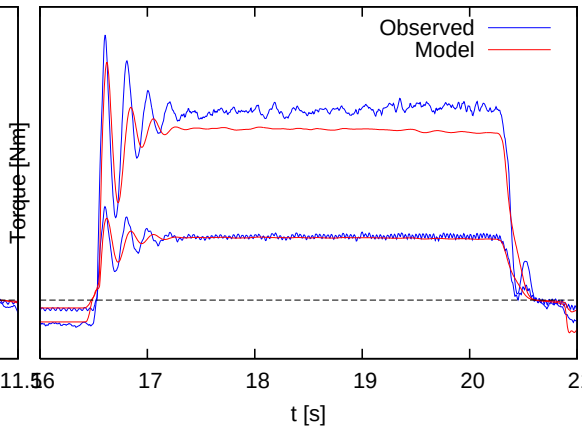
On the top of Fig. 5.2, an overview over the whole time horizon of 37.0 seconds is shown, allowing to judge on the general validity of the model. Detailed graphs of the four acceleration phases follow, each one staring with a *tip in*, terminated by a *tip out* after two to four seconds.



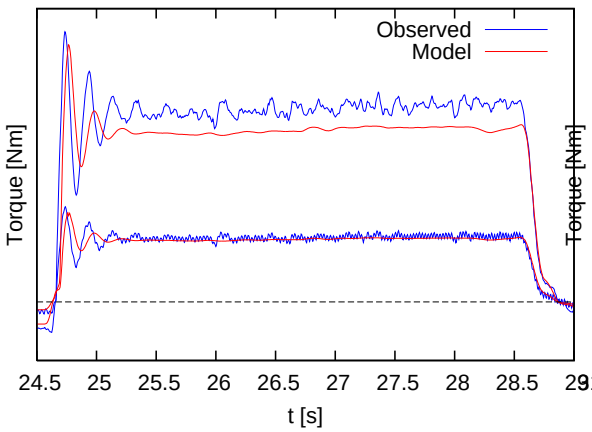
(a) Cardan shaft (M_{cs}, bottom) and side shaft (M_{ss}, top) torque over the whole time horizon.



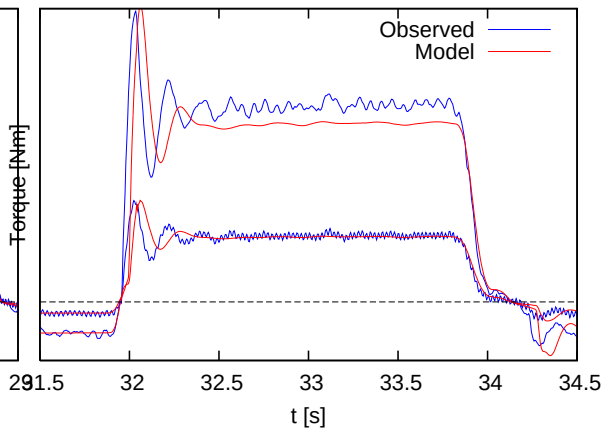
(b) Details of the first acceleration phase.



(c) Details of the second acceleration phase.



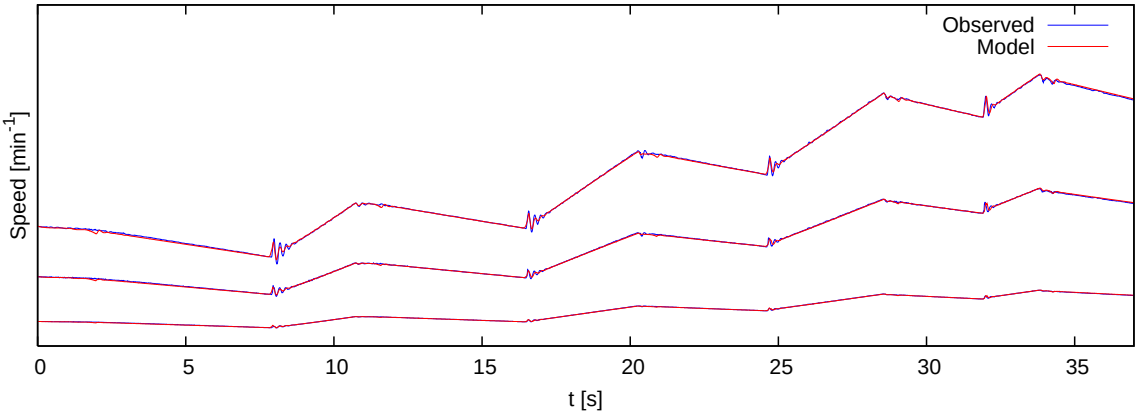
(d) Details of the third acceleration phase.



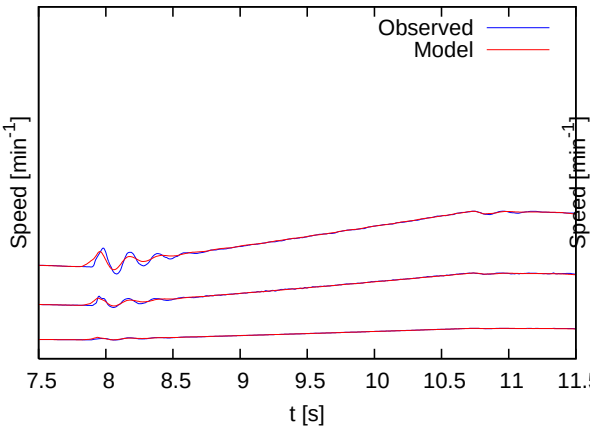
(e) Details of the fourth acceleration phase.

Fig. 5.2: Observed (—) and simulated (—) cardan and side shaft torques after parameter estimation.

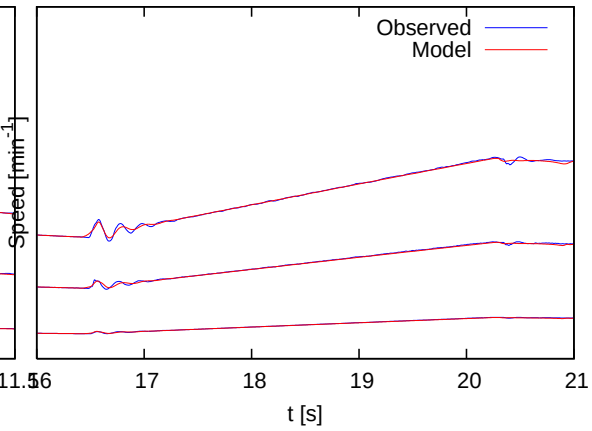
Powertrain Rotation Speeds



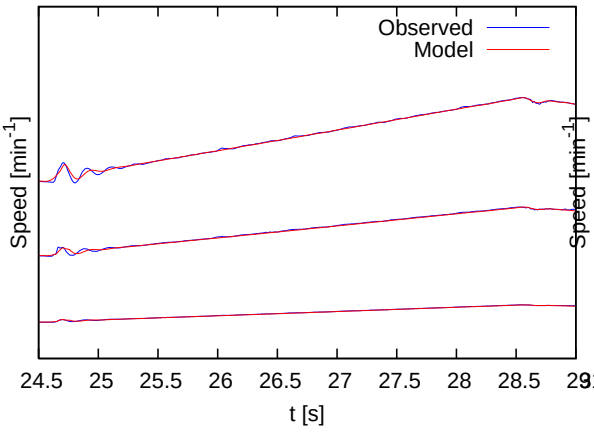
(a) Motor (n_{mot}, top), cardan shaft (n_{cs}, middle), and side shaft (n_{ss}, bottom) rotation speeds over the whole time horizon.



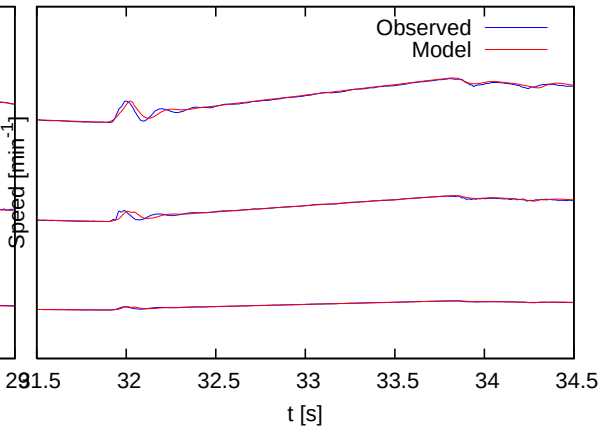
(b) Details of the first acceleration phase.



(c) Details of the second acceleration phase.



(d) Details of the third acceleration phase.

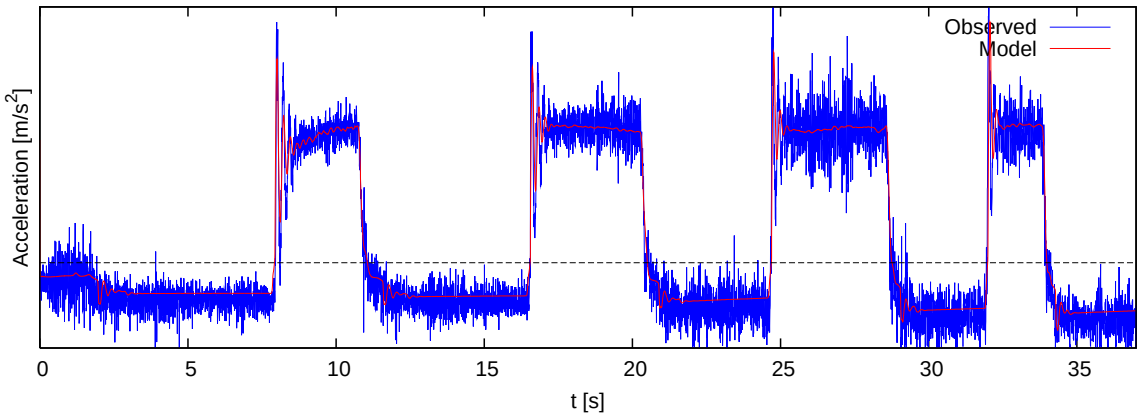


(e) Details of the fourth acceleration phase.

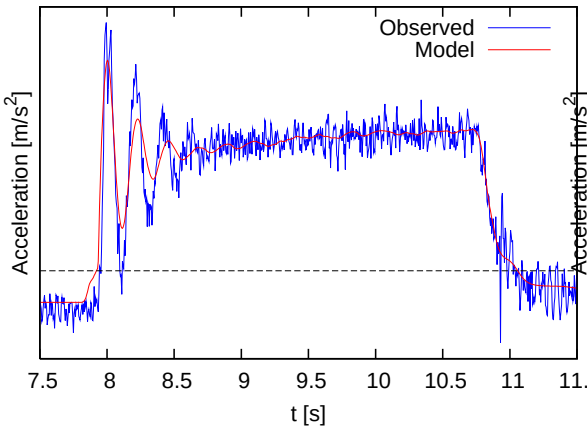
Fig. 5.3: Observed (—) and simulated (—) rotation speeds after parameter estimation: Engine (top), cardan shaft (middle), and side shaft (bottom).

Car Acceleration

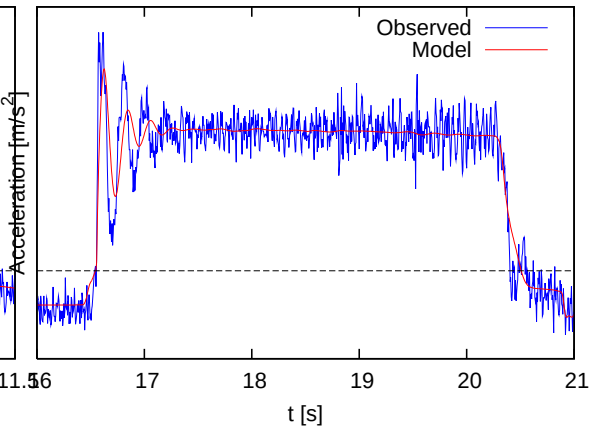
The observed acceleration signal is too noisy to serve as a good reference, and is omitted from the objective. Nonetheless, especially during the first three acceleration phases we obtain an acceptable match of the oscillations in both amplitude and frequency.



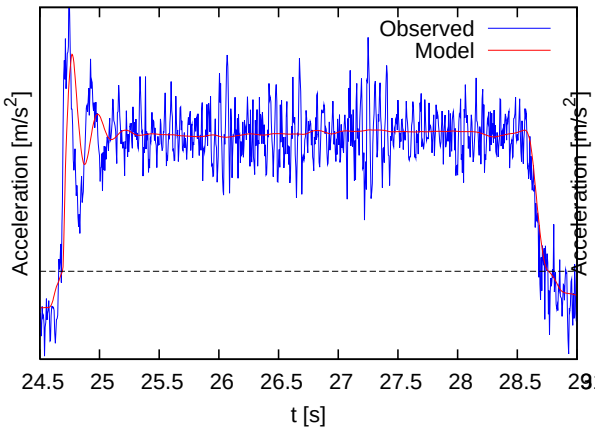
(a) Car acceleration (a_{car}) over the whole time horizon.



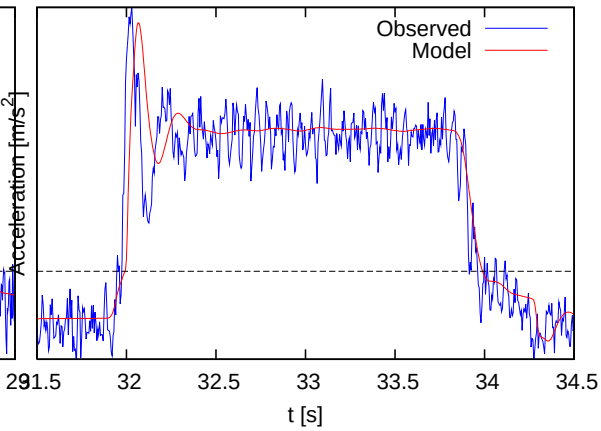
(b) Details of the first acceleration phase.



(c) Details of the second acceleration phase.



(d) Details of the third acceleration phase.

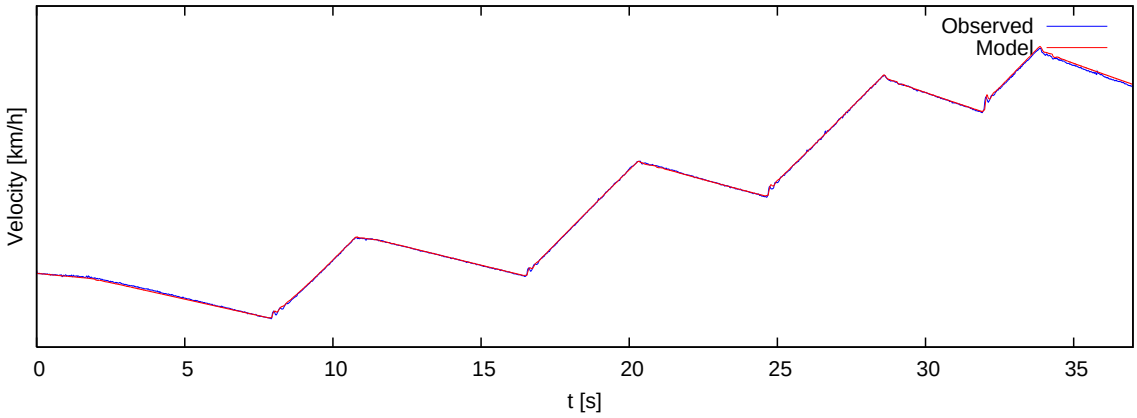


(e) Details of the fourth acceleration phase.

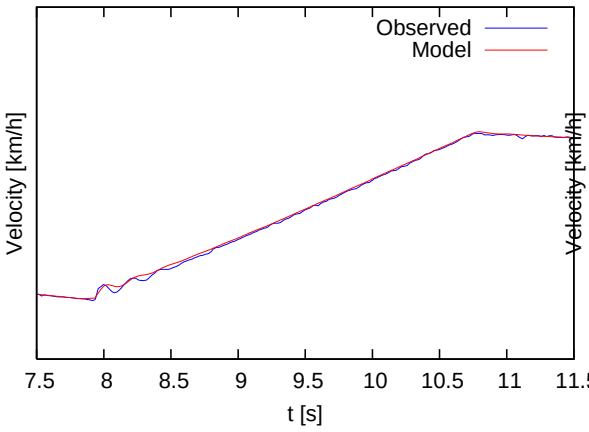
Fig. 5.4: Observed (—) and simulated (—) car acceleration after parameter estimation.

Rear Wheel Velocity

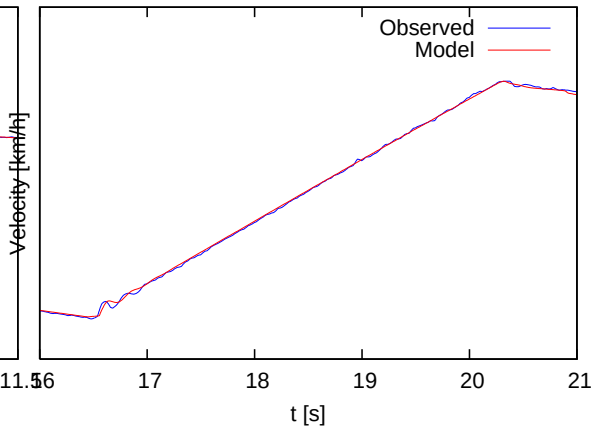
Up to the delay of approximately 31 milliseconds listed in Tab. 5.3, the second velocity signal v_{car} shows identical features and quality of fit. In favor of clarity we omitted it here.



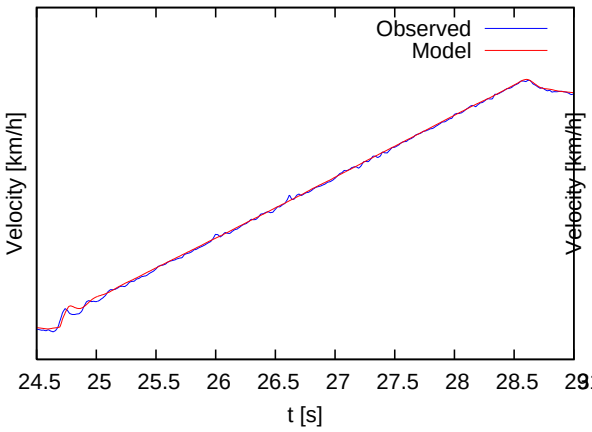
(a) Rear wheel velocity (v_{wh}) over the whole time horizon.



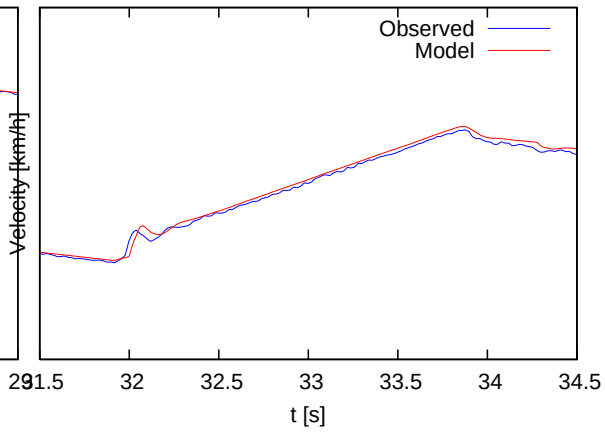
(b) Details of the first acceleration maneuver.



(c) Details of the second acceleration maneuver.



(d) Details of the third acceleration maneuver.



(e) Details of the fourth acceleration maneuver.

Fig. 5.5: Observed (—) and simulated (—) rear wheel velocity after parameter estimation.

5.5 Powertrain Model Identification

This section motivates and describes the construction of a reduced powertrain model. This model is non-stiff, and can be solved by the explicit Runge-Kutta method presented in Chapter 4.

5.5.1 Eigenvalue Analysis

By linearization of the nonlinear ODE system around the point $(t, \hat{\boldsymbol{y}}, \boldsymbol{p})$,

$$\frac{d\boldsymbol{y}}{dt}(t) = \frac{d\boldsymbol{f}}{d\boldsymbol{y}}(t, \boldsymbol{y}, \boldsymbol{p})\Big|_{\boldsymbol{y}=\hat{\boldsymbol{y}}} (\boldsymbol{y} - \hat{\boldsymbol{y}}) + \boldsymbol{f}(t, \hat{\boldsymbol{y}}, \boldsymbol{p}) \tag{5.51}$$

we obtain the nonlinear system's time-dependent fundamental matrix $\boldsymbol{F}(t) := \frac{d\boldsymbol{f}}{d\boldsymbol{y}}(t, \boldsymbol{y}, \boldsymbol{p})$. If analytic derivatives of the right-hand side (cf. Griewank et al. [25]) are not available, the columns of the fundamental matrix can be obtained by finite-difference approximation,

$$\boldsymbol{F}_{\bullet j}(t) := \frac{d\boldsymbol{f}}{dy_j}(t, \boldsymbol{y}, \boldsymbol{p}) \approx \frac{\boldsymbol{f}(t, \boldsymbol{y} + \varepsilon s_j \boldsymbol{e^i}, \boldsymbol{p}) - \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{p})}{\varepsilon s_j}, \tag{5.52}$$

where the s_j are scalers of the states y_j. The disturbance ε should be chosen as $\varepsilon = \sqrt{\mathrm{mach}}$ where mach denotes the machine precision (cf. Bock [6]).

Eigenvalues of the matrix $\boldsymbol{F}$ characterize the fundamental solutions of the ODE system as follows. We may associate an eigenfrequency f_e and a damping factor D with a pair of complex conjugated eigenvalues with non-zero imaginary part, while a decay constant t may be computed for a single non-zero real eigenvalue (cf. Königsberger [33], Stelzer [55]),

$$f_\mathrm{e} := \frac{1}{2\pi} \left|\Im\lambda\right|, \quad D := -\frac{1}{|\lambda|}\Re\lambda, \qquad t := -\frac{1}{\lambda} \quad (\lambda \in \mathbb{R}). \tag{5.53}$$

A stiff ODE system is characterized by eigenvalues with widely differing absolute values. Such systems frequently are impossible to solve with an explicit solver such as explicit Runge-Kutta methods, or require excessively many small steps to be computed.

5.5.2 Eigenvalues of the Powertrain Model

Using the presented scenario and set of parameters, the eigenvalues of the powertrain ODE model were computed over the time horizon of 37.0 seconds. The results are visualized in Fig. 5.6; eigenfrequencies and damping factors can be found in Tab. 5.6.

In Stelzer [55], an earlier revision of the powertrain model was analytically linearized. The eigenvalues of such a system obviously stay fixed over the whole time horizon. Thus, our results contain those of [55] as a subset. A notable difference concerns the cardan shaft's eigenvalues (clusters (3) and (4) in Fig. 5.6), which indicate overdamping if using our parameter set. Stelzer [55] found a complex conjugated pair associated with an eigenfrequency of about 400 Hz here, being highly dependent on the cardan shaft's damping coefficient d_cs. From Tab. 5.4 we find that exactly this parameter is most ill-determined. We can reproduce the reported results by replacing d_cs with an artificially lowered value still within the uncertainty set.

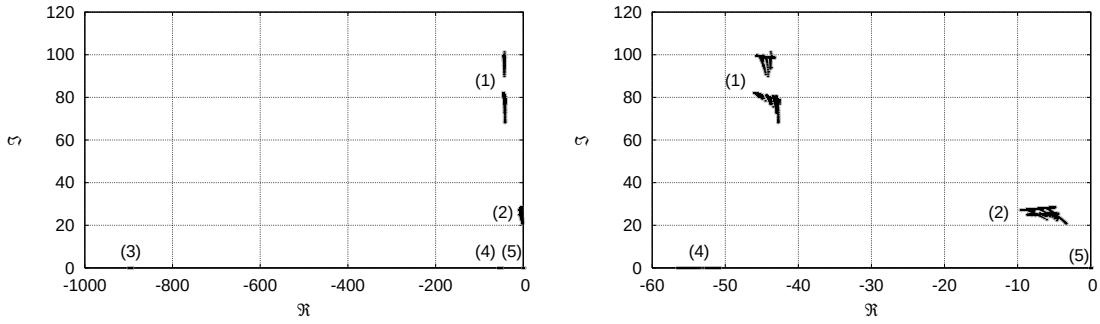


Fig. 5.6: Eigenvalues of the full powertrain model.

	Lowest		Highest		D [−]	f_e [Hz]
(1)	-46.07	$\pm 68.23\imath$	-42.60	$\pm 101.3\imath$	$0.385 - 0.674$	$10.86 - 16.12$
(2)	-9.589	$\pm 20.80\imath$	-3.410	$\pm 28.47\imath$	$0.119 - 0.419$	$3.311 - 4.532$
	Lowest		Highest		t [s]	
(3)	-900.3		-893.6		0.001	
(4)	-56.54		-50.83		$0.018 - 0.02$	
(5)	-0.0086		-0.0143		$69.93 - 116.3$	

Tab. 5.6: Eigenvalues of the full powertrain model during the scenario of Fig. 5.1. Eigenvalues and corresponding eigenfrequencies, damping factors, and decay constants are given to four significant digits.

Concerning the cardan shaft we conclude

1. That the cardan shaft constitutes the main source of stiffness in the powertrain model;

2. That information about a critical parameter determining its behavior cannot be obtained from the observed data available;

3. That it does not contribute to the modeling of the oscillation phenomena we are interested in, taking place at around 4 Hz to 16 Hz (cf. Stelzer [55]).

5.5.3 Curing Stiffness by Removing the Cardan Shaft

As clearly seen from the eigenvalue analysis carried out in the previous section, the optimal parametrization introduces a degree of stiffness into the powertrain model. This makes it inherently hard to treat with explicit integration methods such as Runge-Kutta methods discussed in Chapter 4. The results allow to identify the harmonic oscillator modeling the cardan shaft as the single source of stiffness.

By omitting the cardan shaft (c_cs, d_cs) and the associated axle drive's moment of inertia (J_ad), we're led to the following modifications to the powertrain model. The gearbox input torque M_cs is supplied by the transmission, thus Eq. (2.5) is replaced by

$$M_\mathrm{gb} := \begin{cases} \frac{1}{i_\mathrm{gb}\eta_\mathrm{gb}}\, M_\mathrm{tr} & \text{if } \Delta\varphi_{ss}(t) < 0, \\ \frac{\eta_\mathrm{gb}}{i_\mathrm{gb}}\, M_\mathrm{tr} & \text{if } \Delta\varphi_{ss}(t) \geqslant 0, \end{cases} \tag{5.54}$$

whereas the transmission's input angular velocity ω_{ad} is supplied by the gearbox and Eq. (2.8) is replaced by

$$M_{\mathrm{tr}} := \begin{cases} \frac{1}{i_{\mathrm{tr}}\eta_{\mathrm{tr}}} M_{\mathrm{ss}} + M_{\mathrm{loss,tr}}\left(\omega_{\mathrm{gb}}(t)\right) & \text{if } \Delta\varphi_{\mathrm{ss}}(t) < 0, \\ \frac{\eta_{\mathrm{tr}}}{i_{\mathrm{tr}}} M_{\mathrm{ss}} - M_{\mathrm{loss,tr}}\left(\omega_{\mathrm{gb}}(t)\right) & \text{if } \Delta\varphi_{\mathrm{ss}}(t) \geqslant 0, \end{cases} \tag{5.55}$$

$$\omega_{\mathrm{tr}}(t) := \frac{1}{i_{\mathrm{tr}}}\,\omega_{\mathrm{gb}}(t). \tag{5.56}$$

In this reduced model the cardan shaft and axle drive equations listed in Section 2.1 are no longer required. The resulting model is visualized in Fig. 5.7

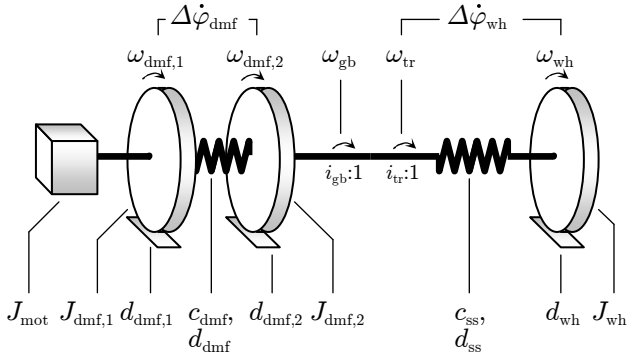


Fig. 5.7: Schematic of the reduced powertrain model.

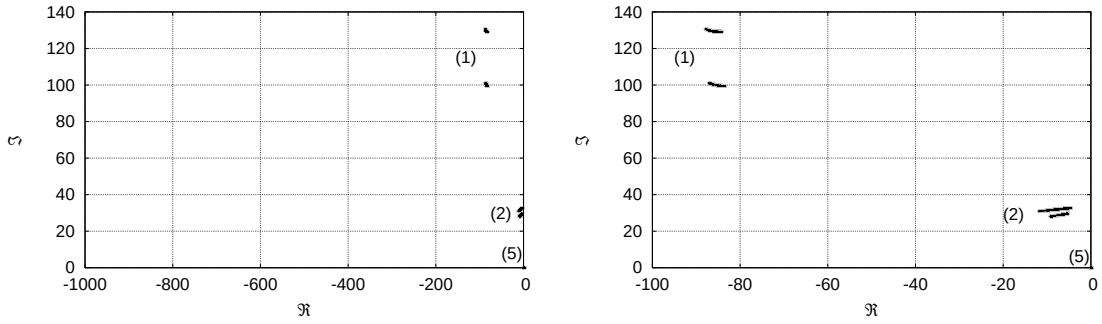


Fig. 5.8: Eigenvalues of the reduced powertrain model.

	Lowest		Highest		D [−]	f_{e} [Hz]
(1)	-87.92	$\pm 99.36\imath$	-83.71	$\pm 130.6\imath$	$0.54 - 0.66$	$15.81 - 20.78$
(2)	-11.90	$\pm 28.06\imath$	-4.706	$\pm\ 32.7\imath$	$0.143 - 0.390$	$4.466 - 5.196$
	Lowest		Highest		t [s]	
(5)	-0.0084		-0.0143		$69.93 - 119.0$	

Tab. 5.7: Eigenvalues of the reduced powertrain model during the scenario of Fig. 5.1. Eigenvalues and corresponding eigenfrequencies, damping factors, and decay constants are given to four significant digits.

Chapter 6

Nonlinear Optimal Control Problems

In this chapter we present a class of optimal control problems, and their treatment using the *direct multiple shooting method* due to Bock and Plitt [7]. We describe a discretization approach to obtain a finite-dimensional nonlinear problem (NLP) that may be solved using, e.g., the SQP method presented in Chapter 3. The introduction of implicit switches into the optimal control problem class is discussed, and an overview over the advantages and sensitive points of the chosen approach is given.

The guiding example of powertrain oscillations is used to to present and formulate a real-world optimal control problem. The oscillation phenomena occurring when a car is being accelerated are discussed in detail, and an objective function for the measurement of such oscillations is presented. We formulate an optimal control problem fitting into the presented problem class. From its solution using the discussed techniques, we obtain engine control schemes that allow for virtually oscillation-free acceleration of the car in a multitude of operating conditions.

6.1 The Continuous Optimal Control Problem

We present here a fairly general optimal control problem formulation closely related to the one treated by the optimal control software package *MUSCOD-II* [38]. From the simulation of the state trajectory $\boldsymbol{x}(t)$ of a dynamic process on the time horizon $\mathcal{T} := [t_0, t_\mathrm{f}] \subset \mathbb{R}$

$$\boldsymbol{x} : \mathcal{T} \longrightarrow \mathcal{X} \subset \mathbb{R}^{n_x}, \quad t \mapsto \boldsymbol{x}(t),$$

we compute and strive to minimize a performance index ϕ

$$\phi\left(t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f}), \boldsymbol{p}\right) \tag{6.1}$$

evaluated at the end of $\mathcal{T}$, while requiring certain constraints to be satisfied. This dynamic process is fully described by the solution of a system of ordinary differential equations (ODEs) with initial values $\boldsymbol{x_0} \in \mathcal{X} \subset \mathbb{R}^{n_x}$ subject to optimization

$$\text{s.t.} \quad \frac{d\boldsymbol{x}}{dt}\left(t;\ \boldsymbol{x_0}, \boldsymbol{u}, \boldsymbol{p}\right) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \quad t \in \mathcal{T}, \tag{6.2a}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x_0}. \tag{6.2b}$$

The process is controlled by an external control function $\boldsymbol{u}(t)$ on the whole of $\mathcal{T}$, also subject to optimization,

$$\boldsymbol{u} : \mathcal{T} \longrightarrow \mathcal{U} \subset \mathbb{R}^{n_u}, \quad t \mapsto \boldsymbol{u}(t).$$

We may require the solution to fulfill implicitly defined inequality constraints on $\mathcal{T}$,

$$\text{s.t.} \quad \boldsymbol{c}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}\big) \geqslant \boldsymbol{0}, \quad t \in \mathcal{T}, \tag{6.3a}$$

as well as equality and inequality point constraints depending on the states and controls,

$$\text{s.t.} \quad \boldsymbol{g}\big(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \ldots, \boldsymbol{x}(t_\mathrm{f}), \boldsymbol{u}(t_\mathrm{f}), \boldsymbol{p}\big) = \boldsymbol{0}, \tag{6.4a}$$

$$\boldsymbol{h}\big(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \ldots, \boldsymbol{x}(t_\mathrm{f}), \boldsymbol{u}(t_\mathrm{f}), \boldsymbol{p}\big) \geqslant \boldsymbol{0}. \tag{6.4b}$$

These point constraints are evaluated on a predefined constraint grid only:

$$t_0 < t_1 < \ldots < t_{m-1} < t_m = t_\mathrm{f}, \quad m \in \mathbb{N}. \tag{6.5}$$

In addition, differential states $\boldsymbol{x}(t)$ and controls $\boldsymbol{u}(t)$ at any point $t \in \mathcal{T}$ are commonly restricted to appropriate subsets of $\mathbb{R}^{n_*}$,

$$\text{s.t.} \quad \underline{\boldsymbol{x}}(t) \leqslant \boldsymbol{x}(t) \leqslant \overline{\boldsymbol{x}}(t), \tag{6.6a}$$
$$\underline{\boldsymbol{u}}(t) \leqslant \boldsymbol{u}(t) \leqslant \overline{\boldsymbol{u}}(t). \tag{6.6b}$$

The same is true for the global model parameters $\boldsymbol{p}$,

$$\text{s.t.} \quad \underline{\boldsymbol{p}} \leqslant \boldsymbol{p} \leqslant \overline{\boldsymbol{p}}. \tag{6.7}$$

These constraints, as well as the initial value condition (6.2b), however, are already covered by the more general inequality constraint function $\boldsymbol{c}$.
The treatment of implicit discontinuities requires a switch function $\boldsymbol{\sigma}$ (4.22) whose sign structure vector

$$\operatorname{sgn} \boldsymbol{\sigma}(t) := \operatorname{sgn} \boldsymbol{\sigma}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}\big) \tag{6.8}$$

appears as a dependency in the ODE system's right-hand side (6.2a) to realize discontinuities of the right-hand side. Discontinuities of the states $\boldsymbol{x}$ are realized using jump functions $\boldsymbol{\Delta}$,

$$\boldsymbol{x}(t_\mathrm{s}^+) = \boldsymbol{\Delta}_j\big(t_\mathrm{s}^-, \boldsymbol{x}(t_\mathrm{s}^-), \boldsymbol{u}(t_\mathrm{s}^-), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t_\mathrm{s}^-)\big) \quad \forall t_\mathrm{s} \in \mathcal{T} : \ \exists j \in \mathbb{N} : \ \sigma_j(t_\mathrm{s}^-) = 0. \tag{6.9}$$

Here, the superscript signs stand for the left- vs. right-hand side limits with respect to time. The theory and algorithms behind the detection, location, and treatment of such implicit switches are presented in Chapter 4.

We assume all functions to be $\mathscr{C}^2$ functions of their arguments. In addition we assume the ODE system's right-hand side $\boldsymbol{f}$ (6.2a) to fulfill for given initial values $\boldsymbol{x_0}$, controls $\boldsymbol{u}(t)$, and parameters $\boldsymbol{p}$ the usual assumptions ensuring local existence and uniqueness of the solution $\boldsymbol{x}(t)$ describing the state trajectory of the dynamic process on $\mathcal{T}$ (cf. Stoer and Bulirsch [56]).

6.1.1 Generality of the Problem Formulation

Some of the restrictions found in the above optimal control problem class can be lifted by procedures described in this section. We mention scenarios with variable end-time t_f, and objective functions other that the Mayer-term objective ϕ evaluated at the end of the dynamic process.

Variable Time Horizons

It is without loss of generality to use a fixed time horizon $\mathcal{T}$ in the above formulation. For instance, observe that for problems with free end time $\tilde{t}_\mathrm{f}$, we may introduce a new differential state x_{n_x+1} whose initial value $x_{n_x+1}(t_0) := \tilde{t}_\mathrm{f}$ represents the free end time and is subject to optimization. By appropriate extension and linear time transformation of the original ODE system's right hand side,

$$\theta(t) := t_0 + t(x_{n_x+1} - t_0), \quad t \in [0,1] \subset \mathbb{R}, \tag{6.10a}$$
$$\frac{d\boldsymbol{x}}{d\theta}(\theta;\ \boldsymbol{x_0}, \boldsymbol{u}) = \tilde{\boldsymbol{f}}\big(\theta, \boldsymbol{x}(\theta), \boldsymbol{u}(\theta)\big) := (x_{n_x+1} - t_0) \begin{bmatrix} \boldsymbol{f}\big(\theta, \boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ \boldsymbol{0} \end{bmatrix}, \tag{6.10b}$$

we easily obtain a new problem description on the normalized time horizon $[0,1] \subset \mathbb{R}$ whose solution coincides with the one of the original problem on $[t_0, \tilde{t}_\mathrm{f}] \subset \mathbb{R}$. For numerical reasons, a reasonable implementation will actually want to automatically apply these transformations so as to always work on the normalized time horizon internally.

Objective Functions

In the literature, several different performance index types are distinguished. Amongst them one finds the objective function of generalized Bolza[1] type (6.11), comprising the Mayer[2]-type objective M evaluated at the end of the controlled process (as included in the above problem class), and the Lagrange-type integral objective Λ evaluated on the whole of $\mathcal{T}$,

$$\phi\left(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\right) := M\left(t_{\mathrm{f}}, \boldsymbol{x}(t_{\mathrm{f}}), \boldsymbol{p}\right) + \int_{\mathcal{T}} \Lambda\left(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\right)\, dt. \tag{6.11}$$

In addition, least-squares objectives

$$\phi\left(t, \boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\right) := \int_{\mathcal{T}} L^2\left(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\right)\, dt. \tag{6.12}$$

as discussed in Chapter 5 are commonly supported. From a theoretical point of view it is obviously sufficient to discuss Mayer-type objectives. All other types of objectives may be regarded as additional differential states with terminal values contributing to the objective of Mayer type.

6.2 Discretization of the Continuous Problem

In order to solve the presented optimal control task, we discretize the continuous problem formulation by replacing the control function $\boldsymbol{u}$ with approximations of finite dimension. The resulting discrete problem can then be solved by standard Newton-Lagrange algorithms for nonlinear programs (NLP), such as sequential quadratic programming (SQP), an overview of which is given in Section 3.2.

6.2.1 Control Discretization

We replace the control function $\boldsymbol{u}$ by an approximation defined by way of a finite set of control parameters $\boldsymbol{q}_j \in \mathbb{R}^{n_q}$, $j = 1, \ldots, k$. From an algorithmic point of view, separable representations of the approximation are to be preferred over global representations for several reasons (cf. Leineweber [38]):

1. Discontinuities, frequently encountered in exact solutions to bang-bang optimal control problems, are difficult to deal with.
2. The influence of a specific parameter $\boldsymbol{q}_j$ on a global representation may be difficult to grasp (think of, e.g., power or Fourier series here).
3. For numerical efficiency, it is of great benefit to have a Lagrangian function that is separable in the sense that it may be composed from sums of functions depending on a local subset of the control discretization parameters only; see Equation (6.23a).

As a consequence, a piecewise approximation to the control function $\boldsymbol{u}$ is sought for. In a general framework, we introduce a control discretization grid on $\mathcal{T} = [t_0, t_{\mathrm{f}}]$,

$$t_0 < t_1 < \ldots < t_{m-1} < t_m = t_{\mathrm{f}}, \quad m \in \mathbb{N}. \tag{6.13}$$

For simplicity, we will assume that this grid coincides with the decoupled point constraint grid (6.13). We compose approximations to the control function $\boldsymbol{u}$ on each of the subintervals $[t_i, t_{i+1}] \subset \mathcal{T}$ from selected basis functions $\boldsymbol{b}_i$ parametrized by control parameters $\boldsymbol{q}_i \in \mathbb{R}^{k_i \times n_u}$,

$$\boldsymbol{u}(t)\,|_{[t_i, t_{i+1}]} := \boldsymbol{b}_i(t, \boldsymbol{q}_i), \quad i = 0, \ldots, m-1. \tag{6.14}$$

Simple choices as depicted in Fig. 6.1 could be

[1] Oskar Bolza (1857–1942)
[2] Adolph Mayer (1839–1908)

1. piecewise constant controls

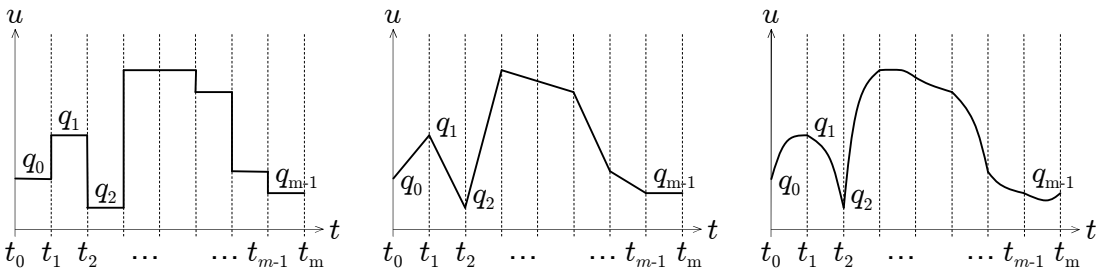
$$\boldsymbol{b}_i(t, \boldsymbol{q}_i) := \boldsymbol{q}_{i,1}, \quad k_i = 1, \tag{6.15}$$

2. piecewise linear controls,

$$\boldsymbol{b}_i(t, \boldsymbol{q}_i) := \boldsymbol{q}_{i,1} + (\boldsymbol{q}_{i,2} - \boldsymbol{q}_{i,1}) \frac{t - t_i}{t_{i+1} - t_i}, \quad k_i = 2, \tag{6.16}$$

3. or cubic spline controls, the functions $\beta_j(\tau)$ being the cubic spline base polynomials on $[0, 1] \subset \mathbb{R}$,

$$\boldsymbol{b}_i(t, \boldsymbol{q}_i) := \sum_{j=1}^{k_i} \boldsymbol{q}_{i,j} \beta_j \left(\frac{t - t_i}{t_{i+1} - t_i} \right), \quad k_i = 4. \tag{6.17}$$



(a) Piecewise constant controls.

(b) Piecewise linear controls with continuity.

(c) Cubic spline controls with continuity, but not differentiability.

Fig. 6.1: Examples of control discretizations.

It is obvious that the choice of the discretization grid (6.5) strongly influences the quality of the obtained solution. If a continuous piecewise discretization of $\boldsymbol{u}$ is desired, it is favourable to enforce continuity through additional constraints on the control discretization parameters,

$$\boldsymbol{b}_{i+1}\left(t_{i+1}, \boldsymbol{q}_{i+1}\right) - \boldsymbol{b}_i\left(t_{i+1}, \boldsymbol{q}_i\right) = 0. \quad i = 0, \ldots, m-1. \tag{6.18}$$

6.2.2 Constraint Discretization

The inequality constraints (6.3) have to be discretized, and are enforced on the discretization grid nodes only.

$$\boldsymbol{c}\big(t_i, \boldsymbol{s}_i, \boldsymbol{b}_i\left(t, \boldsymbol{q}_i\right), \boldsymbol{p}\big) \quad \geqslant \quad \boldsymbol{0}. \tag{6.19a}$$

While this generally leads to an approximation of the original constraints only, one finds that in many practical cases this approximation is sufficiently precise, or even exact due to the nature of the underlying problem [38]. A more advanced approach from the field of semi-infinite programming is presented in Potschka [48] to reduce the approximative solution's violation of the continuous constraints.

6.2.3 State Parametrization

After the presented discretization steps, the new problem might now be solved by combining single shooting integration of the IVP with some NLP solving technique.

As shown by Bock [6] in the case of boundary-value problems (BVPs) for parameter estimation, however, this approach will frequently fail even if excellent estimates of the initial values $\boldsymbol{x}(t_0)$ are available. The nonlinear problem solving method may fail to converge, or a solution to the initial-value problem may not even exist numerically, so that the constraints $\boldsymbol{c}$, $\boldsymbol{g}$, and

$\boldsymbol{h}$ cannot even be evaluated on the whole of $\mathcal{T}$. These problems arise due to the propagation of discretization and truncation errors while integrating unstable ODE systems.

The *direct multiple shooting method*, cf. Bock and Plitt [7], considerably improves the situation. Instead of using a single initial value $\boldsymbol{x}(t_0) = \boldsymbol{x_0}$ only, we compute the values

$$\boldsymbol{s}_i = \boldsymbol{x}(t_i), \quad i = 0, \ldots, m, \tag{6.20}$$

on the nodes of the discretization grid (6.5), or more generally on a suitable sub- or superset of that grid. The IVP (6.2) is thus separated into m IVPs on the sub-intervals $[t_i, t_{i+1}] \subset \mathcal{T}$,

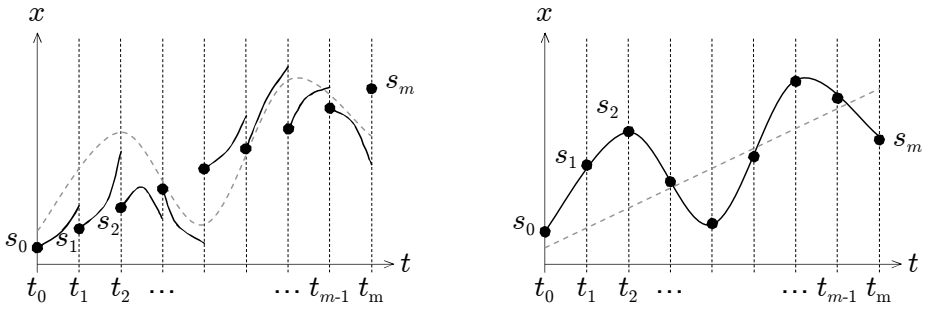
$$\frac{d\boldsymbol{x}}{dt}(t;\ \boldsymbol{s}_i, \boldsymbol{q}_i) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{b}_i(t, \boldsymbol{q}_i), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \quad t \in [t_i, t_{i+1}], \tag{6.21a}$$

$$\boldsymbol{x}(t_i) = \boldsymbol{s}_i. \tag{6.21b}$$

The problem is further augmented by the matching conditions

$$\boldsymbol{x}(t_{i+1}, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}) = \boldsymbol{s}_{i+1}, \quad i = 0, \ldots, m-1, \tag{6.22}$$

that ensure continuity of the spliced trajectory $\boldsymbol{x}(t)$ hence representing a solution of the original IVP (6.2).



(a) Initial states and spliced trajectory.

(b) Matching conditions fulfilled by the final trajectory.

Fig. 6.2: State discretization for the direct multiple-shooting method.

6.2.4 The Discrete Nonlinear Problem

We summarize here the nonlinear problem formulation resulting from the discretization of the optimal control problem class we presented.

$$\min_{\boldsymbol{p}, \boldsymbol{q}_i \boldsymbol{s}_i, \boldsymbol{s}_m} \ \phi\left(t_\mathrm{f}, \boldsymbol{s}_m, \boldsymbol{p}\right) \tag{6.23a}$$

$$\text{s.t.} \quad \frac{d\boldsymbol{x}_i}{dt}(t;\ \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}) = \boldsymbol{f}\big(t, \boldsymbol{x}_i(t), \boldsymbol{b}_i\left(t, \boldsymbol{q}_i\right), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \quad t \in [t_i, t_{i+1}], \tag{6.23b}$$

$$\boldsymbol{x}_i(t_i) = \boldsymbol{s}_i, \tag{6.23c}$$

$$\boldsymbol{0} = \boldsymbol{s}_{i+1} - \boldsymbol{x}_i\left(t_{i+1}, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}\right), \tag{6.23d}$$

$$\boldsymbol{0} = \boldsymbol{b}_{i+1}\left(t_{i+1}, \boldsymbol{q}_{i+1}\right) - \boldsymbol{b}_i\big(t_{i+1}, \boldsymbol{q}_i\big), \tag{6.23e}$$

$$\boldsymbol{0} = \boldsymbol{g}\big(t_i, \boldsymbol{s}_i, \boldsymbol{b}_i\left(t_i, \boldsymbol{q}_i\right), \boldsymbol{p}\big), \tag{6.23f}$$

$$\boldsymbol{0} \geqslant \boldsymbol{h}\big(t_i, \boldsymbol{s}_i, \boldsymbol{b}_i\left(t_i, \boldsymbol{q}_i\right), \boldsymbol{p}\big), \tag{6.23g}$$

$$\boldsymbol{0} \geqslant \boldsymbol{c}\big(t_i, \boldsymbol{s}_i, \boldsymbol{b}_i\left(t_i, \boldsymbol{q}_i\right), \boldsymbol{p}\big), \tag{6.23h}$$

where $i = 0, \ldots, m-1$. This problem may be solved with the sequential quadratic programming (SQP) method presented in Chapter 3.

6.3 Treatment of Implicit Switches

In this section we look at the treatment of implicit switches on the multiple-shooting and SQP level of the algorithm. We discuss both restrictions and features that arise from the inclusion of implicit discontinuities in the optimal control problem.

6.3.1 Implicit Switches and the Multiple Shooting Method

During the iterative process of finding a solution to the optimal control problem, guided by the sequential quadratic programming (SQP) algorithm, special care must be taken of implicit switch events.
The integrator method described in Chapter 4 properly accounts for updates of the first-order sensitivities. Thus, the discontinuity is effectively hidden from the SQP level as long as the switch time point t_s stays within a single multiple-shooting interval $[t_i, t_{i+1}]$.
Things are different when looking at second-order sensitivity information contained in the Hessian of the Lagrangian (3.18). Finite-difference computation of the Hessian, in addition to being extremely costly, suffers from imprecise approximations of the second derivatives. Within the optimal-control software *MUSCOD-II*, rank-2 BFGS updates are employed (cf. Leineweber [38], Nocedal and Wright [43]) that accumulate curvature information from first-order sensitivities over the course of several iterations.
Here the caveat is, that, once an implicit switch crosses a multiple-shooting discretization grid boundary, the solution as well as it's derivatives of first order change in a discontinuous manner. The curvature information accumulated in the Hessian approximate may essentially be invalid for subsequent iterations. For this reason, we currently artificially restrict switch events to the shooting interval they occurred in during the first SQP iteration. If optimality cannot be reached under this restriction, different initial controls are chosen to move the switch event to the proper adjacent shooting interval.
Experience shows that discontinuities in the right-hand side only ($\mathbf{\Delta} = 0$ using the notation of Chapter 4) can be overcome by the SQP algorithm, and the Hessian approximation appears to get adjusted by the BFGS updates of subsequent iterations. As of now, switches showing jumps in the differential states, however, need to be restricted as described above. A well-founded theory that allows for adapting the SQP algorithm to handle such discontinuities that cannot be hidden on the integrator level seems highly desirable and mandates further research efforts.

6.3.2 Implicit Objective and Constraints

At the very beginning of this chapter we presented an optimal control problem class with specialized objective function as well as equality- and inequality constraints. While the Mayer objective is evaluated at the end of the time horizon $\mathcal{T}$ only, the constraint functions may be evaluated on pre-selected (or possibly all) multiple shooting grid nodes. It is, however, impossible to evaluate an objective or constraint in an a-priori unknown point in time being defined by an implicit condition.
In *MUSCOD-II*, this lack of flexibility may partially be overcome by the use of a *multi-stage* problem setup. Essentially, we concatenate several sub-problems with variable duration, each of which is terminated upon satisfaction of the respective implicit condition.
We're still limited in several ways, though. We need to know in advance the order and number of required stages, and need to make sure that both stay fixed during the whole optimization scenario. We must also ensure that all implicit conditions are met on every iteration, and that the first time they're met is actually the proper time.
The inclusion of implicit switches relieves us from this burden. We introduce additional differential states representing the value of the objective or the equality- or inequality constraint in question. Upon fulfillment of the implicit condition, we trigger a jump in the respective

differential state. Conventional Mayer-term objective or end-point constraints are then evaluated from the values of these additional states at the end of the time horizon. This effectively removes all of the mentioned restrictions of multi-stage problem setups. In addition, the effort to be put in coding and proper problem setup is much reduced.

6.4 Optimal Control of Powertrain Oscillations

In the final section of this chapter we focus again on the guiding example of powertrain oscillations. Using the model presented in Chapter 1 and the set of model parameters identified in Section 5.4, we describe the optimal control problem formulation. Convincing results are obtained that allow for acceleration of the powertrain while preventing nearly all oscillations.

6.4.1 Overview

In this section we present the optimal control problem setup. We give some motivation and explanations from engineering for the formulation of objective and constraints and the presentation of the obtained results.

Controls

The optimal control problem formulation respects two different engine controls. First, the *plain engine torque*, denoted by $u_1(t)$, is controllable within given upper and lower bounds for the torque level (6.24d) and derivative (6.24e). It can be changed every ten milliseconds, thus defining the minimal multiple-shooting grid resolution. In the actual implementation we control the piecewise constant derivative $\dot{u}_1(t)$ and introduce an additional differential state for $u_1(t)$ itself, which in turn is piecewise linear continuous.
The *ignition timing angle* $u_2(t)$ ranges between zero and one (6.24f). It defines a *negative* offset to the plain engine torque, depending on the powertrain's current revolutionary speed, and the plain engine torque. This is accomplished by interpolating from a table of given torque offsets for certain revolutionary speeds and plain engine torques. When presenting results, the graphs show the *effective engine torque* only. The engine torque's lower limit might thus appear to be violated, but it is clear that this is actually never the case.
In reality we are able to instantaneously change the ignition timing angle six times during a single duty cycle (two revolutions) of the engine. As this would require a variable multiple-shooting grid resolution depending on the powertrain's revolutionary speed, we chose to restrict ourselves to changing the ignition timing angle on the same time basis as the plain engine torque, and argue that the incurred loss of freedom does not significantly worsen the obtained solutions.

Objective

Our objective is to diminish powertrain oscillations, effectively observed in the acceleration of the car, as far as possible. In order to define an objective function measuring oscillations, we compute the non-oriented area of the acceleration trajectory over the *final* acceleration a_f that is reached once the oscillations have decayed. This becomes clear from Fig. 6.3. The starting time t_a is scenario-dependent. The implementation is using a continuous least-squares objective function (6.24), treated by the Gauß-Newton algorithm presented in Chapter 3.

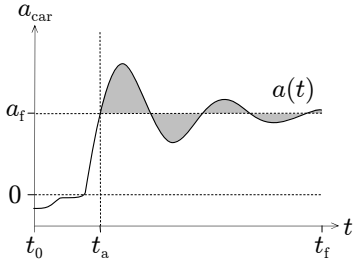
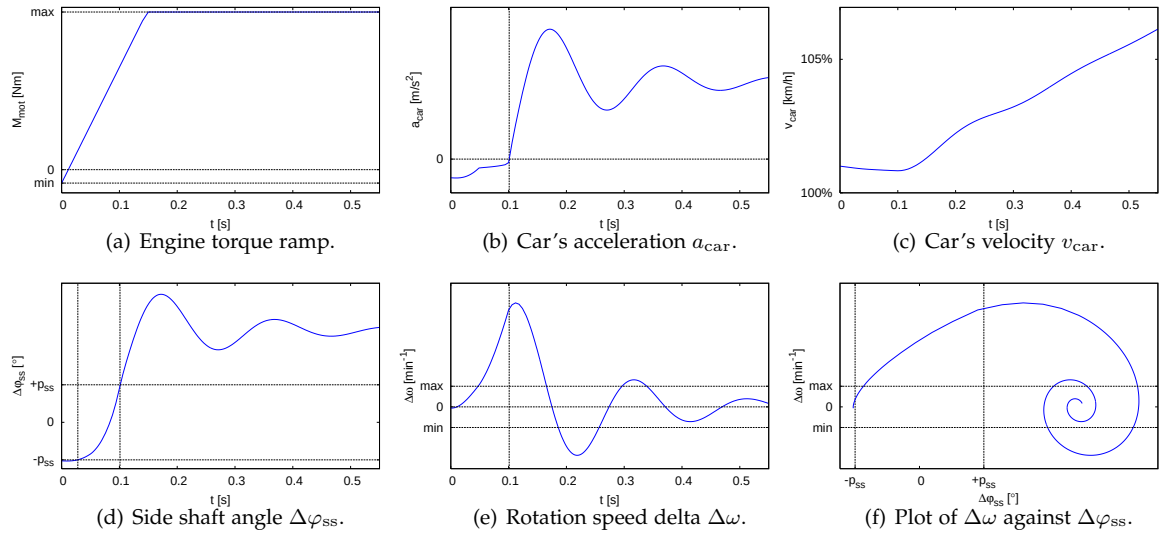


Fig. 6.3: A least-squares objective for measuring oscillations.

Graphs

Fig. 6.4 visualizes the powertrain's behavior in the uncontrolled case. For non-disclosure reasons, all graphs of simulated data remain unlabeled on the ordinate. We simply apply a plain engine torque ramp of maximum height and slope (Fig. 6.4(a)), and observe the resulting oscillations in both acceleration (Fig. 6.4(b)) and velocity (Fig. 6.4(c)).



(a) Engine torque ramp. (b) Car's acceleration a_{car}. (c) Car's velocity v_{car}.

(d) Side shaft angle $\Delta\varphi_{\mathrm{ss}}$. (e) Rotation speed delta $\Delta\omega$. (f) Plot of $\Delta\omega$ against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.4: Powertrain oscillations in coasting mode at 3,000 rpm when applying an engine torque ramp with maximum height and slope.

Constraints

The side shaft's angular delta is shown in Fig. 6.4(d) to explain the rotation speed delta constraint. As seen in the introductory chapter, the side shaft contains a mechanical play between $-p_{\mathrm{ss}}$ degrees and $+p_{\mathrm{ss}}$ degrees, indicated by vertical lines in the graph.

When starting in *traction mode*, the car is being actively accelerated by the engine, and the side shaft is twisted in positive direction, i.e., $\Delta\varphi_{\mathrm{ss}} > p_{\mathrm{ss}}$. Further acceleration of the car is then possible without the side shaft needing to traverse the play.

When starting in *coasting mode*, however, the car is moved by its own inertia and braked by the engine. The side shaft is twisted in negative direction, i.e., $\Delta\varphi_{\mathrm{ss}} < -p_{\mathrm{ss}}$, and accelerating the car requires to fully traverse the play in order to enter traction mode first. The incurred delay of approximately 100 milliseconds can be seen in Fig. 6.4(b). When leaving the play and entering traction mode (as indicated by the vertical line at around 0.1 s in the graphs above), it is important to do so with limited revolutionary speed difference of the powertrain's ends (Fig. 6.4(e)). Otherwise, the mechanical play will hit its end-stop, which is undesirable for both acoustic and durability concerns. Using the trivial engine torque ramp to accelerate the car, this limit is clearly violated as seen in Fig. 6.4(e). The corresponding constraint is given by Eq. (6.24a) in the optimal control problem formulation below.

Finally, Fig. 6.4(f) shows the side shaft's angular delta $\Delta\varphi_{\mathrm{ss}}(t)$ plotted against the powertrain's rotation speed difference $\Delta\omega(t)$. Coasting mode scenarios start in the leftmost area, while traction mode scenarios start in the rightmost. The central area indicates an active side shaft play. Engine control schemes satisfying the constraint are characterized by a trajectory that crosses the vertical line $+p_{\mathrm{ss}}$ between the horizontal lines indicating the constraint on $\Delta\omega$.

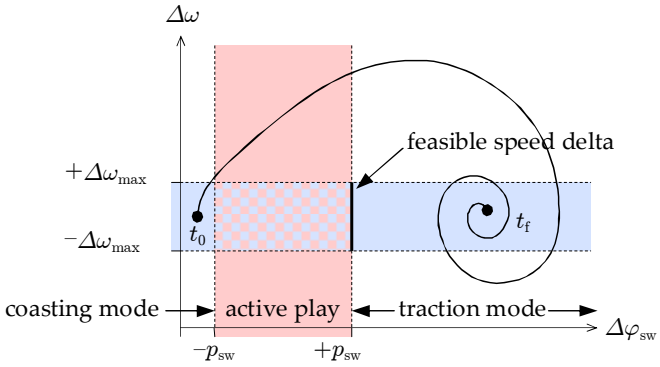


Fig. 6.5: Explanation of the plot $\Delta\varphi_{ss}$ against $\Delta\omega$. (—) is the resulting trajectory going from t_0 to t_f (•) in time. The active side shaft play area ( ) and the feasible rotation speed delta area ( ) are shaded. A trajectory is feasible if it crosses the strong vertical line (|).

6.4.2 Problem Formulation

The complete optimal control problem formulation is as follows:

$$\min_{\boldsymbol{u},\boldsymbol{x}} \quad \int_{t_a}^{t_f} \left(a_{car}(t) - a_f\right)^2 \, dt + \int_{t_0}^{t_f} \left(\gamma_1 u_1^2(t) + \gamma_2 u_2^2(t)\right) \, dt \tag{6.24a}$$

$$\text{s.t.} \quad \Delta\omega_{\max} \geqslant \frac{1}{i_{gb} i_{tr}} \omega_{dmf,1}(t_\Delta) - \omega_{wh}(t_\Delta), \tag{6.24b}$$

$$\frac{d\boldsymbol{x}}{dt}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \quad t \in \mathcal{T}, \tag{6.24c}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{6.24d}$$

$$u_1(t) \in [\underline{u}_1, \overline{u}_1], \tag{6.24e}$$

$$\frac{du_1}{dt}(t) \in [\underline{\delta u}_1, \overline{\delta u}_1] \quad \text{on} \quad (t_i, t_{i+1}) \subset \mathcal{T},\ i = 0, \ldots, m-1, \tag{6.24f}$$

$$u_2(t) \in [0, 1]. \tag{6.24g}$$

where t_0, t_a, and t_f are preselected time points. The unique point t_Δ is implicitly and defined by $\Delta\varphi_{ss}(t_\Delta) = +p_{ss}$, and is obviously effective in coasting mode ($\Delta\varphi_{ss}(t_0) < -p_{ss}$) only.
Both controls are regularized by inclusion in the objective using appropriately chosen scalers γ_1 and γ_2. We refer to Chapter 1 for a detailed presentation of the powertrain model $\boldsymbol{f}$, and the computation of $a_{car}(t)$. The states $\omega_{dmf,1}(t)$, $\omega_{wh}(t)$, and $\Delta\varphi_{ss}(t)$ are scalar components of the state vector $\boldsymbol{x}(t)$.
Over a time horizon of one second, the discretized nonlinear problem resulting from the use of 100 multiple-shooting nodes consists of 1,000 variables, 800 equality constraints, and 2,001 inequality constraints.

6.4.3 Results

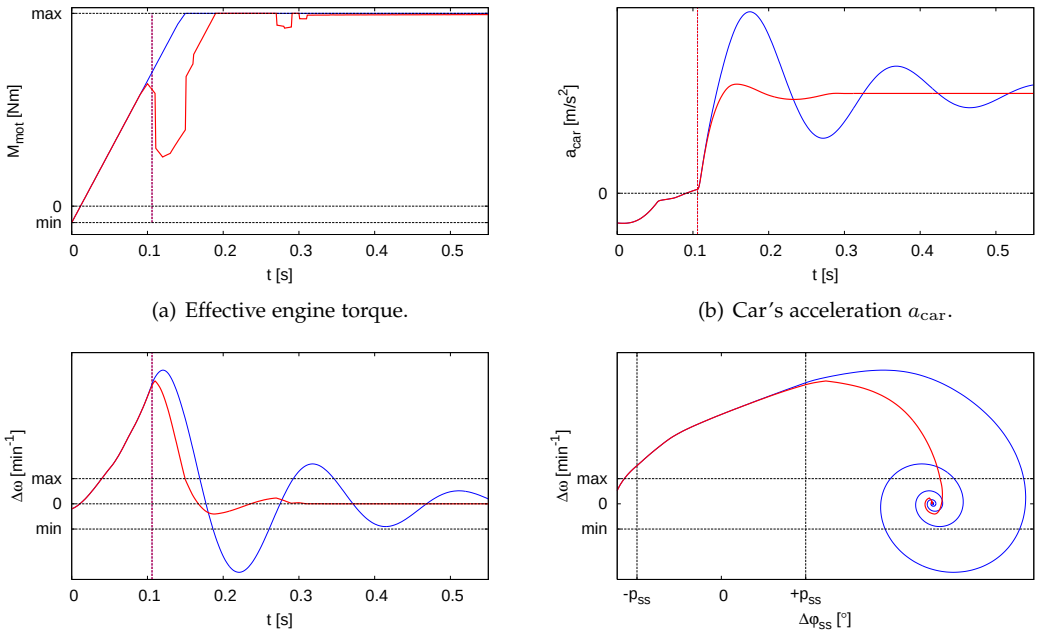
The above optimization problem was solved starting in coasting mode ($\Delta\varphi_{ss}(t_0) < -p_{ss}$) and traction mode ($\Delta\varphi_{ss}(t_0) > +p_{ss}$), for initial engine speeds $n_{mot}(t_0)$ of 2,000 and 5,000 revolutions per minute, and for desired final accelerations a_f of 50% and 100% of the maximum possible acceleration.
In general it can be seen that the oscillation phenomenon is worse in coasting mode, and worse for lower initial engine speeds. As can be seen from the graphs, we succeeded in obtaining engine control schemes that diminish powertrain oscillations to a very satisfying degree for all evaluated scenarios.

All computations were performed using the *RKFSWT* integrator presented in Chapter 4 with a tolerance TOL $= 10^{-10}$, and using the MSSQP algorithm of *MUSCOD-II* (see Diehl et al. [12], Leineweber [38]). Tab. 6.1 holds details on the numerical quality of the obtained solutions. From the column of iteration counts it is clear that the main difficulty in solving the above problem is to satisfy the rotation speed delta constraint (6.24a) in coasting mode.

Scenario				KKT	Scaled	Lagrange	Number of
Fig.	Mode	rpm	a_f	Tolerance	Infeasibility	Gradient	Iterations
6.6	C., unconstr.	2000	100%	$2.67 \cdot 10^{-7}$	$1.10 \cdot 10^{-9}$	$6.91 \cdot 10^{-4}$	4
6.7	Coasting	2000	50%	$8.97 \cdot 10^{-9}$	$2.84 \cdot 10^{-6}$	$8.37 \cdot 10^{-5}$	25
6.8	Coasting	2000	100%	$1.06 \cdot 10^{-8}$	$5.30 \cdot 10^{-5}$	$3.84 \cdot 10^{-4}$	75
6.9	Coasting	5000	50%	$5.39 \cdot 10^{-7}$	$2.14 \cdot 10^{-4}$	$8.30 \cdot 10^{-3}$	62
6.10	Coasting	5000	100%	$4.02 \cdot 10^{-8}$	$1.60 \cdot 10^{-4}$	$4.87 \cdot 10^{-2}$	99
6.11	Traction	2000	50%	$6.79 \cdot 10^{-10}$	$1.57 \cdot 10^{-10}$	$5.61 \cdot 10^{-3}$	3
6.12	Traction	2000	100%	$2.65 \cdot 10^{-7}$	$2.98 \cdot 10^{-7}$	$5.10 \cdot 10^{-3}$	3
6.13	Traction	5000	50%	$8.73 \cdot 10^{-12}$	$1.61 \cdot 10^{-11}$	$4.78 \cdot 10^{-6}$	4
6.14	Traction	5000	100%	$3.48 \cdot 10^{-9}$	$1.17 \cdot 10^{-10}$	$3.56 \cdot 10^{-4}$	3

Tab. 6.1: Minimum powertrain oscillation: Qualities of the solutions. The acceptable KKT tolerance was set to 10^{-6}. Refer to Diehl et al. [12], Leineweber [38] for a detailed discussion of these values.

Coasting Mode without the Rotation Speed Delta Constraint



(a) Effective engine torque.

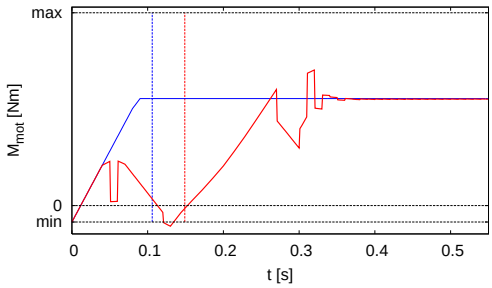
(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

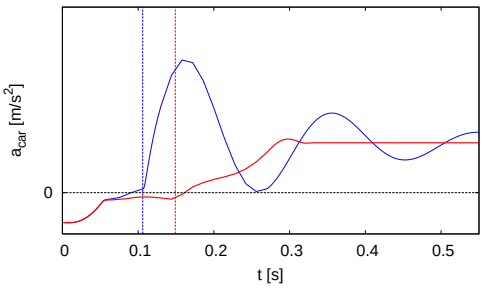
(d) The constraint on $\Delta\omega$ is violated.

Fig. 6.6: Minimum powertrain oscillations ignoring the rotation speed delta constraint. Here $n_{\mathrm{mot}}(t_0) = 2000$ min^{-1}, $t_{\mathrm{a}} = 0$ s, a_f maximal. (—) denotes the optimal solution while (—) denotes the trajectories generated by a torque ramp with maximum slope.

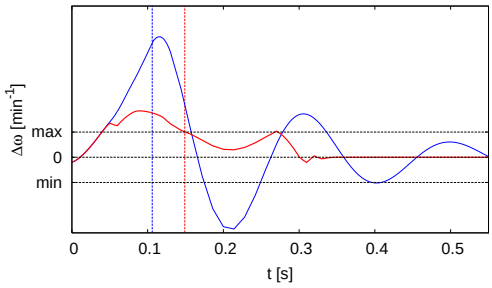
Coasting Mode at 2,000 rpm



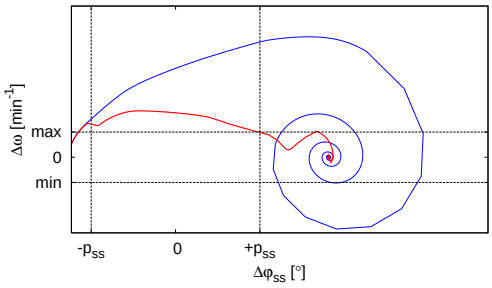
(a) Effective engine torque.



(b) Car's acceleration a_{car}.

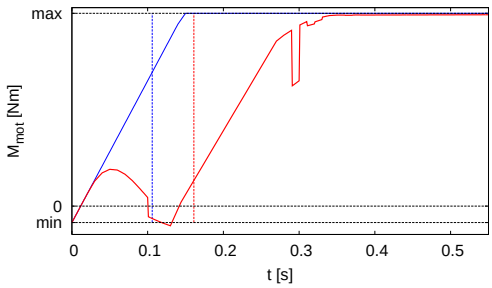


(c) Rotation speed delta $\Delta\omega$.

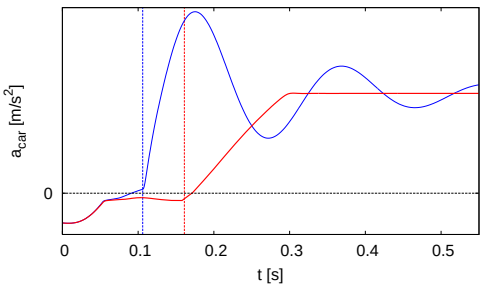


(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

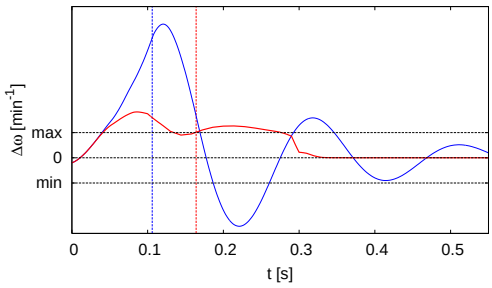
Fig. 6.7: Minimum powertrain oscillations in coasting mode. Here $n_{\mathrm{mot}}(t_0) = 2000$ min^{-1}, $t_{\mathrm{a}} = 0.25$ s, a_f at 50% of the maximum.



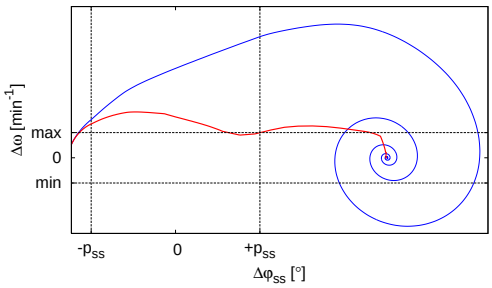
(a) Effective engine torque.



(b) Car's acceleration a_{car}.



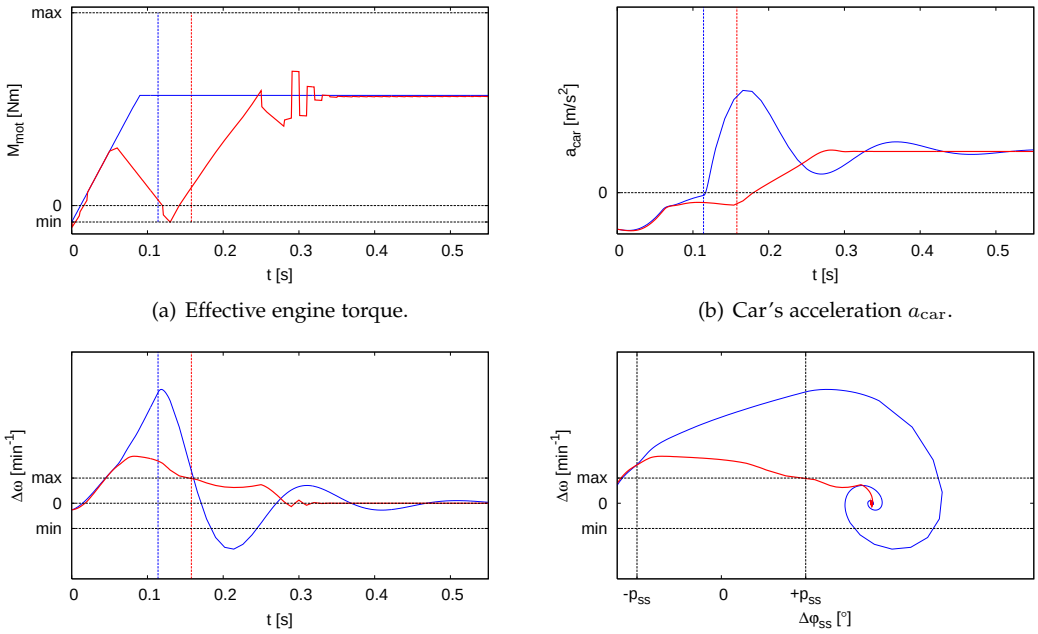
(c) Rotation speed delta $\Delta\omega$.



(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.8: Minimum powertrain oscillations in coasting mode. Here $n_{\mathrm{mot}}(t_0) = 2000$ min^{-1}, $t_{\mathrm{a}} = 0.25$ s, a_f maximal. (—) denotes the optimal solution while (—) denotes the trajectories generated by a torque ramp with maximum slope.

Coasting Mode at 5,000 rpm



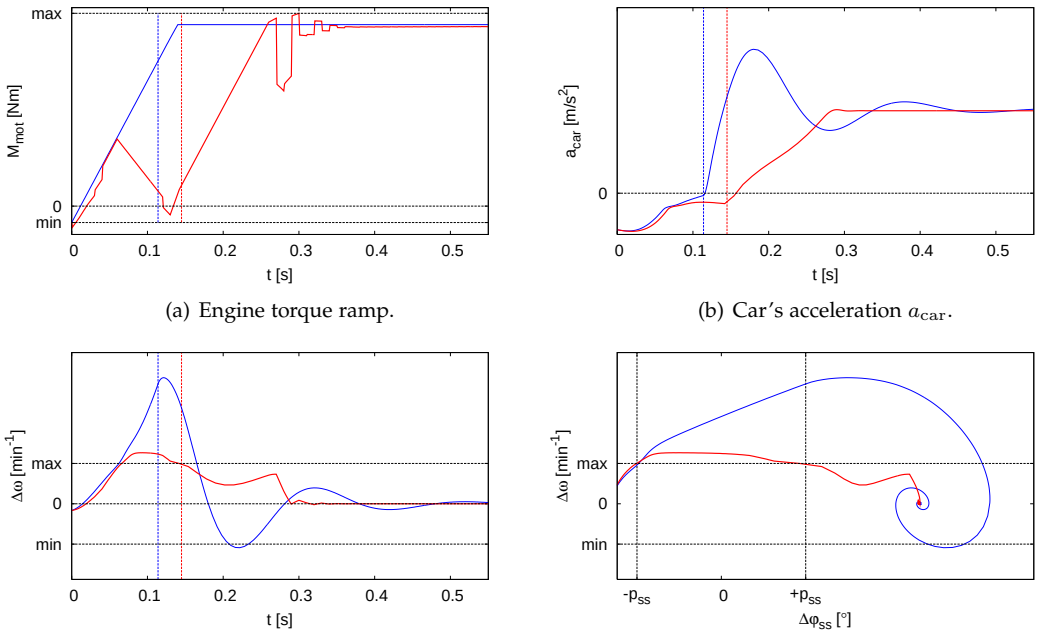
(a) Effective engine torque.

(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.9: Minimum powertrain oscillations in coasting mode. Here $n_{\mathrm{mot}}(t_0) = 5000$ min^{-1}, $t_{\mathrm{a}} = 0.1$ s, a_f at 50% of maximum.



(a) Engine torque ramp.

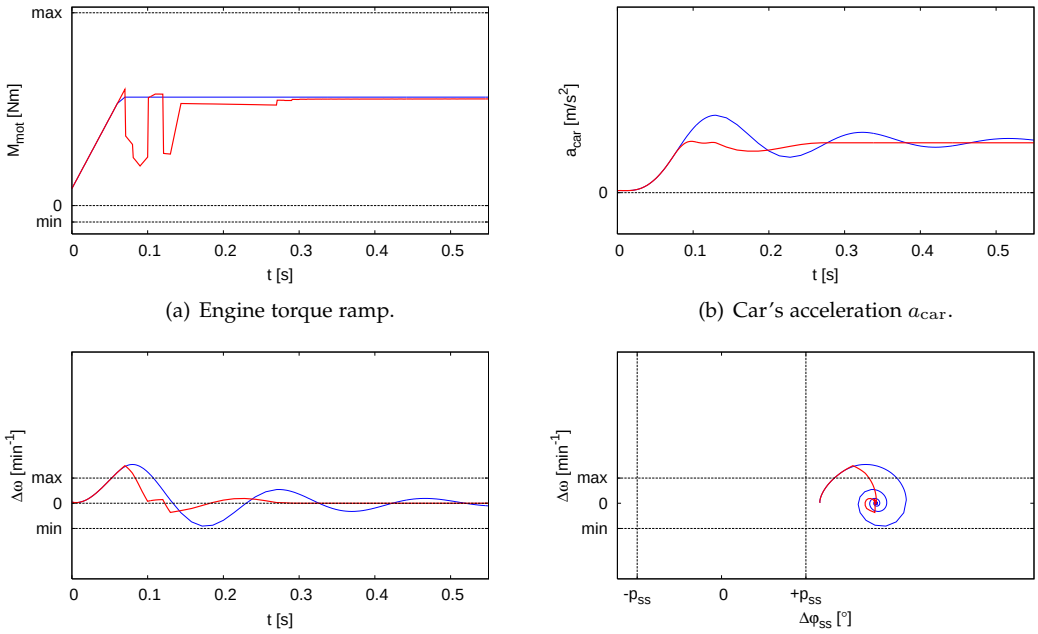
(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.10: Minimum powertrain oscillations in coasting mode. Here $n_{\mathrm{mot}}(t_0) = 5000$ min^{-1}, $t_{\mathrm{a}} = 0.1$ s, a_f maximal. (—) denotes the optimal solution while (—) denotes the trajectories generated by a torque ramp with maximum slope.

Traction Mode at 2,000 rpm



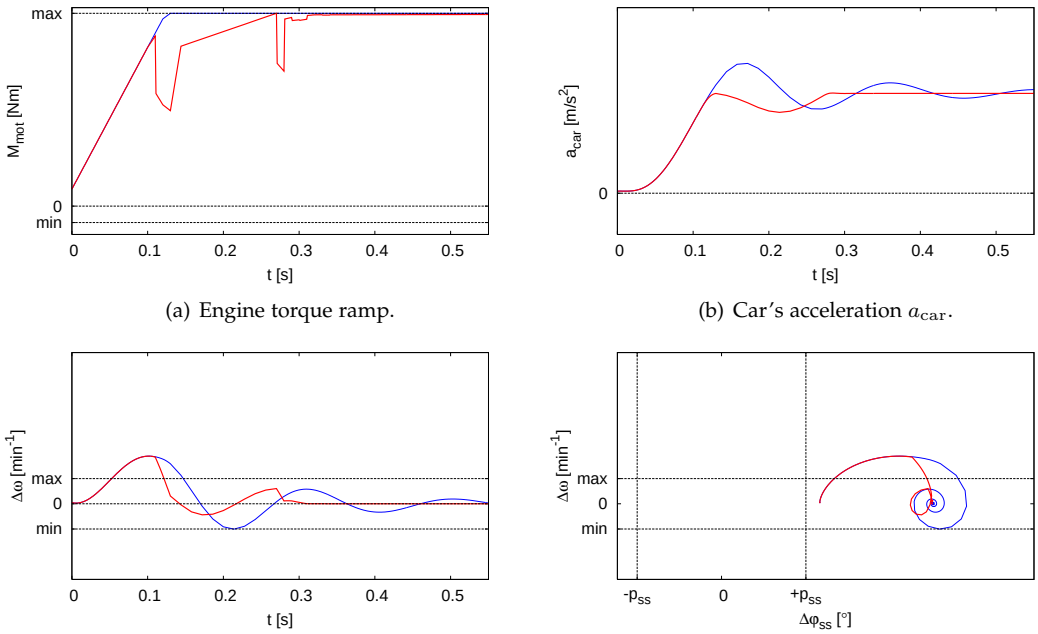
(a) Engine torque ramp.

(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.11: Minimum powertrain oscillations in traction mode. Here $n_{\mathrm{mot}}(t_0) = 2000$ min^{-1}, $t_{\mathrm{a}} = 0$ s, a_f at of 50% of maximum.



(a) Engine torque ramp.

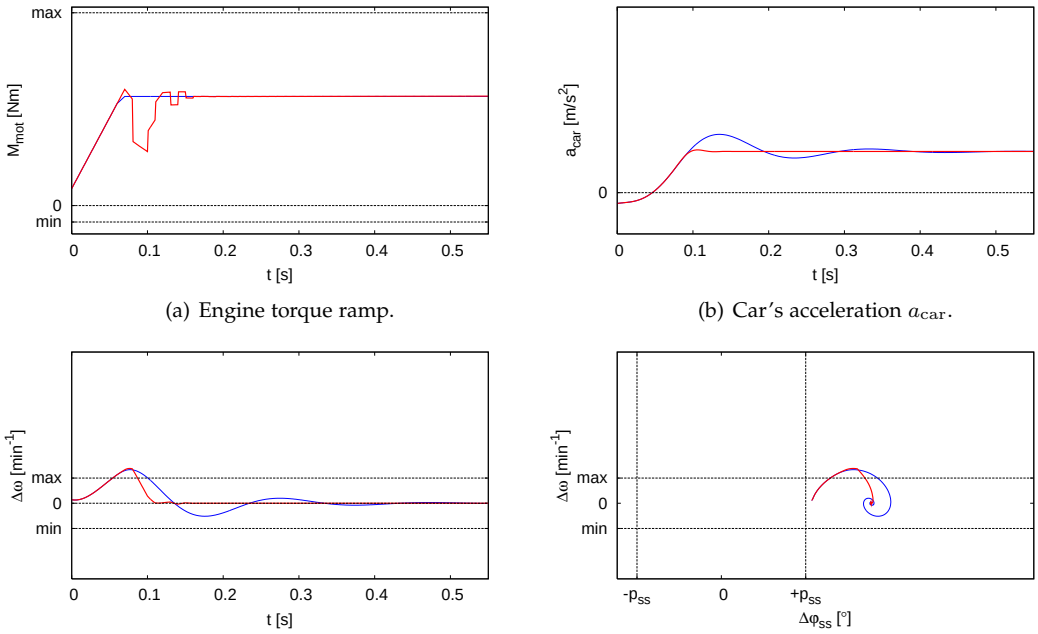
(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.12: Minimum powertrain oscillations in traction mode. Here $n_{\mathrm{mot}}(t_0) = 2000$ min^{-1}, $t_{\mathrm{a}} = 0$ s, a_f maximal. (—) denotes the optimal solution while (—) denotes the trajectories generated by a torque ramp with maximum slope.

Traction Mode at 5,000 rpm



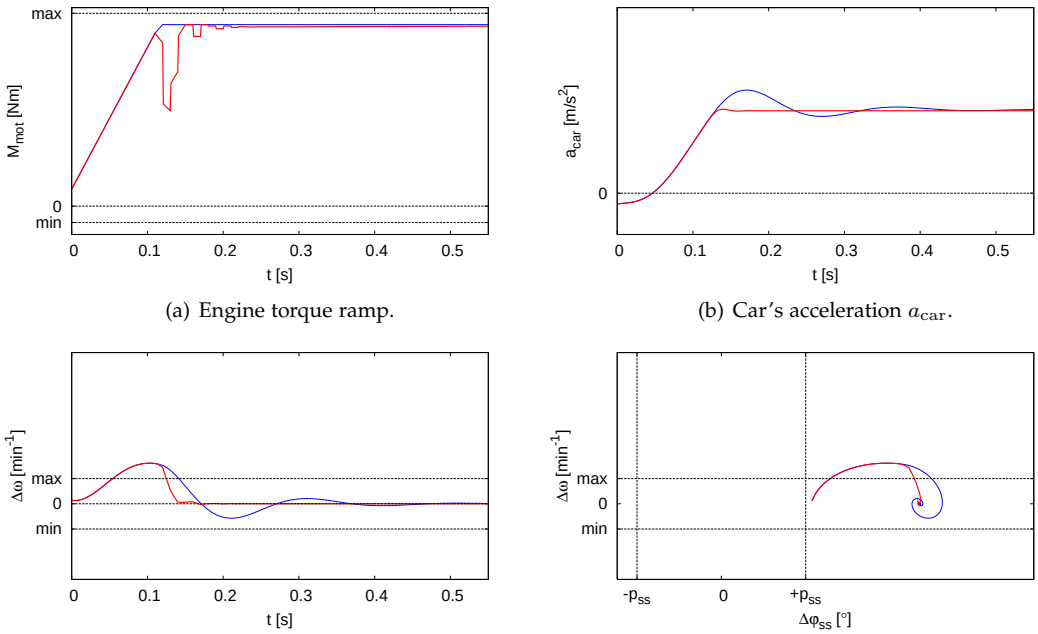
(a) Engine torque ramp.

(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.13: Minimum powertrain oscillations in traction mode. Here $n_{\mathrm{mot}}(t_0) = 5000$ min^{-1}, $t_{\mathrm{a}} = 0$ s, a_f at 50% of maximum.



(a) Engine torque ramp.

(b) Car's acceleration a_{car}.

(c) Rotation speed delta $\Delta\omega$.

(d) $\Delta\omega$ plotted against $\Delta\varphi_{\mathrm{ss}}$.

Fig. 6.14: Minimum powertrain oscillations in traction mode. Here $n_{\mathrm{mot}}(t_0) = 5000$ min^{-1}, $t_{\mathrm{a}} = 0$ s, a_f maximal. (—) denotes the optimal solution while (—) denotes the trajectories generated by a torque ramp with maximum slope.

Chapter 7

Robust Optimal Control Problems

In this chapter we consider nonlinear programs with uncertain parameters, and derive algorithms that yield solutions that exhibit robustness against the specified uncertainties. The popular worst-case approach of Ben-Tal and Nemirovskii [3] is presented, and leads to nonlinear programs with a bi-level structure that is hard to solve computationally. A linearization approach due to Diehl et al. [13] and Körkel et al. [35], independently derived by Ma and Braatz [40], restores the NLP structure and allows for treatment of the resulting robust nonlinear problems with Newton-Lagrange techniques such as the SQP method of Chapter 3.
Using the presented approach, we describe a robustified formulation of the optimal control problem class of Chapter 6. We give details on the generation of second-order sensitivities, and reveal a sparsity structure in the resulting sensitivity matrices. We further present an approach to implement proper updates to the second-order sensitivities in implicit discontinuities.

We apply the resulting algorithm to the problem of gentle powertrain acceleration, focusing on the problem of controlling a smooth transition from coasting to traction mode. Convincing results that exhibit robustness against the uncertainties identified in Chapter 5.4 are presented, and prove the applicability of the invented algorithm.

The presented algorithm has been implemented within the *ROBUST* framework extension to the optimal control software package *MUSCOD-II*, a brief description of which can be found in Appendix C.

7.1 Uncertain Nonlinear Programs

In this section we present a class of robust nonlinear equality and inequality constrained optimization problems with uncertain parameters. We address this problem class by a robust worst-case formulation due to Ben-Tal and Nemirovskii [3], which is difficult to treat computationally.

7.1.1 Problem Formulation

We consider the following class of uncertain nonlinear problems

Definition 7.1. ***Uncertain Nonlinear Problem***

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad f(\boldsymbol{x},\boldsymbol{u}) \tag{7.1a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) = \boldsymbol{0}, \tag{7.1b}$$

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u}) \geqslant \boldsymbol{0}, \tag{7.1c}$$

with an uncertain parameter vector $\boldsymbol{p} \in \mathcal{P}$. The functions f, $\boldsymbol{g}$, and $\boldsymbol{h}$ are $\mathscr{C}^2$ functions of their arguments.

The parameters $\boldsymbol{p}$ are considered uncertain, and in general, we do not know the exact value of $\boldsymbol{p}$. We are merely confident to find the actual parameter $\boldsymbol{p}$ within some uncertainty set $\mathcal{U}_\mathrm{p}$.

Note that there is no actual restriction in the fact that the objective f and the inequality constraints h are assumed to be independent of p. This apparent independence is easily cured by the introduction of trivial additional equality constraints $x_{n_x+i} = p_i$, $i = 1, \ldots, n_p$. The optimization variables of this problem class are partitioned into *states* $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and *controls* $\boldsymbol{u} \in \mathbb{R}^{n_u}$. This distinction naturally arises in model-based optimal control problems such as the nominal optimal control problem class presented in Chapter 6. The Jacobian $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$ is assumed to be non-singular, so the states $\boldsymbol{x}$ can be regarded as implicit functions of the controls $\boldsymbol{u}$ and the uncertain parameters $\boldsymbol{p}$.

7.1.2 Expressing Uncertainty of a Parameter

The parameters $\boldsymbol{p} \in \mathcal{P}$ are assumed to be uncertain, and we address this uncertainty by assuming that we may restrict $\boldsymbol{p}$ to a generalized closed ball $\mathcal{U}_\mathrm{p}$, the uncertainty set.

Definition 7.2. ***Uncertainty Set***
Let $\boldsymbol{p}$ be an uncertain parameter, and denote its nominal assumed value with $\overline{\boldsymbol{p}}$. The set $\mathcal{U}_{\boldsymbol{p}}$ of all values $\tilde{\boldsymbol{p}}$ that the parameter $\boldsymbol{p}$ may actually assume,

$$\mathcal{U}_{\boldsymbol{p}} := \left\{ \tilde{\boldsymbol{p}} \in \mathbb{R}^{n_p} \;\middle|\; ||\overline{\boldsymbol{p}} - \tilde{\boldsymbol{p}}|| \leqslant 1 \right\}, \tag{7.2}$$

*is referred to as the **uncertainty set** of $\boldsymbol{p}$. It is defined using a norm $||\cdot||$ suiting the statistical distribution of the uncertain parameter.*

Observe that the ball's center $\overline{\boldsymbol{p}}$ takes the role of a mean value for a statistical distribution of the uncertain parameter values. Here, two specially shaped uncertainty sets are of practical interest:

1. Confidence Ellipsoids:

   Confidence ellipsoids are generated by normally (Gaussian) distributed random parameters with expectation value $\overline{\boldsymbol{p}}$ and variance-covariance matrix Σ,

   $$\Sigma_{ij} := \mathrm{cov}(p_i, p_j), \quad i, j = 1, \ldots, n_p,$$

   $$\mathcal{U}_{\boldsymbol{p}} = \left\{ \tilde{\boldsymbol{p}} \in \mathbb{R}^{n_p} \;\middle|\; \left|\left| \Sigma^{-\frac{1}{2}} \left(\tilde{\boldsymbol{p}} - \overline{\boldsymbol{p}} \right) \right|\right|_2 \leqslant 1 \right\}. \tag{7.3}$$

2. Confidence Boxes:

   If lower and upper bounds $\boldsymbol{p}_\mathrm{l}$, $\boldsymbol{p}^\mathrm{u}$ for a uniformly distributed uncertain parameter $\boldsymbol{p}$ are known, expectation value $\overline{\boldsymbol{p}}$ and maximum deviation $\boldsymbol{\Delta p}$ simply are

   $$\overline{\boldsymbol{p}} := \frac{1}{2} \left(\boldsymbol{p}_\mathrm{l} + \boldsymbol{p}^\mathrm{u} \right), \qquad \boldsymbol{\Delta p} := \frac{1}{2} \left(\boldsymbol{p}_\mathrm{l} - \boldsymbol{p}^\mathrm{u} \right),$$

   and the uncertainty set

   $$\mathcal{U}_{\boldsymbol{p}} := \left\{ \tilde{\boldsymbol{p}} \in \mathbb{R}^{n_p} \;\middle|\; \boldsymbol{p}_\mathrm{l} \leqslant \tilde{\boldsymbol{p}} \leqslant \boldsymbol{p}^\mathrm{u} \right\}$$

   may be expressed by choosing the infinity norm $||\cdot||_\infty$,

   $$\mathcal{U}_{\boldsymbol{p}} := \left\{ \tilde{\boldsymbol{p}} \in \mathbb{R}^{n_p} \;\middle|\; \left|\left| \mathrm{diag}(\boldsymbol{\Delta p})^{-1} \left(\tilde{\boldsymbol{p}} - \overline{\boldsymbol{p}} \right) \right|\right|_\infty \leqslant 1 \right\}. \tag{7.4}$$

Mixed types of confidence sets for different components of the uncertain parameter vector $\boldsymbol{p}$ can easily be accounted for by composing the uncertainty set $\mathcal{U}_\mathrm{p}$ using the cartesian product of the respective uncertainty subsets of uniform type.

7.1.3 Incorporating Uncertainty into the Problem

We chose the classical worst-case formulation of Ben-Tal and Nemirovskii [3] in order to account for uncertainty of the parameters $\boldsymbol{p}$ in the formulation of problem (7.1a). We consider the uncertain scenario to be a game, in which the optimizer plays against an adverse player. Knowing the controls $\boldsymbol{u}$ chosen by the optimizer in advance, the adverse player responds by choosing uncertain parameters $\boldsymbol{p}$ and resulting states $\boldsymbol{x}$ in the most unfavorable way. Parameters $\boldsymbol{p}$ are chosen from the uncertainty set $\mathcal{U}_\mathrm{p}$, and the unknowns $\boldsymbol{x}$ are only subject to equality constraints imposed on the problem.

To compute this choice we introduce the worst-case operator W_min as a tool that, if given the optimizers choice of controls $\boldsymbol{u}$, yields the worst possible value of a scalar-valued function φ to be minimized with respect to the controls.

Definition 7.3. ***Worst-Case Response of the Adverse Player***
Let $\boldsymbol{u} \in \mathbb{R}^{n_u}$ be the set of controls chosen by the optimizer, and let $\varphi(\boldsymbol{x}, \boldsymbol{u}) \longrightarrow \mathbb{R}$ be a $\mathscr{C}^1$ function. Then, respecting the equality constraints $\boldsymbol{g}$ and the uncertainty set $\mathcal{U}_\mathrm{p}$, the worst-case response $W_\mathrm{min}(\varphi, \boldsymbol{u})$ of the adverse player is given by the solution of the following maximization problem:

$$W_\mathrm{min}(\varphi, \boldsymbol{u}) := \max_{\boldsymbol{x}, \boldsymbol{p}} \quad \varphi(\boldsymbol{x}, \boldsymbol{u}) \tag{7.5a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{0}, \tag{7.5b}$$

$$||\boldsymbol{p} - \overline{\boldsymbol{p}}|| \leqslant 1. \tag{7.5c}$$

Worst-case choices for the objective f and each of the inequality constraint functions h_i, $i = 1, \ldots, n_h$ are computed using the operator W_min. We arrive at the following worst-case robust counterpart formulation of problem (7.1a).

Definition 7.4. ***Worst-Case Robust Counterpart Problem***

$$\min_{\boldsymbol{u}} \quad W_\mathrm{min}(f, \boldsymbol{u}) \tag{7.6a}$$

$$\text{s.t.} \quad -W_\mathrm{min}(-h_i, \boldsymbol{u}) \geqslant 0, \quad i = 1, \ldots, n_h. \tag{7.6b}$$

Due to its bi-level structure, nesting a maximization problem into a minimization problem, this problem is difficult to solve without imposing further restrictions on the functions f, $\boldsymbol{g}$, and $\boldsymbol{h}$.

7.2 A Computationally Feasible Linearized Approach

In order to avoid this bi-level structure we apply a linearization approach developed by Diehl et al. [13] and independently by Ma and Braatz [40]. Related approaches may be found in Diehl [11], Diehl et al. [14]. The basic idea is to replace the worst-case operator W_min by a suitable approximation $\tilde{W}_\mathrm{min}$ that is easier to compute.

7.2.1 Worst-Case Choice as a Convex Problem

Given the controls $\boldsymbol{u}$ chosen by the optimizer, we linearize both the objective f and the inequality constraints $\boldsymbol{h}$ around the the point $(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})$ that shall satisfy the equality constraints $\boldsymbol{g}$. We then replace the worst-case operator W_min by its linearized form $\tilde{W}_\mathrm{min}$ which takes the structure of a convex maximization problem.

Definition 7.5. ***Approximating Convex Problem for the Worst-Case Choice***

$$\tilde{W}_\mathrm{min}(\varphi, \boldsymbol{u}) := \max_{\boldsymbol{\Delta x}, \boldsymbol{\Delta p}} \quad \varphi(\overline{\boldsymbol{x}}, \boldsymbol{u}) + \frac{\partial \varphi}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}) \boldsymbol{\Delta x} \tag{7.7a}$$

$$\text{s.t.} \quad \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) \boldsymbol{\Delta x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) \boldsymbol{\Delta p} = \boldsymbol{0}, \tag{7.7b}$$

$$||\boldsymbol{\Delta p}|| \leqslant 1. \tag{7.7c}$$

Again the function φ serves as a placeholder for the objective f and the inequality constraints h_i, $i = 1, \ldots, n_h$. This convex problem has an analytical solution, which we'll derive after some small preparations.

Definition 7.6. ***Dual of a Norm***
Let $||\cdot||$ denote a norm on $\mathbb{R}^n$. The associated dual norm $||\cdot||_*$ on $\mathbb{R}^n$ is defined by

$$\begin{aligned} ||\cdot||_* : \mathbb{R}^n \longrightarrow \mathbb{R}, \quad a \mapsto \max_{b \in \mathbb{R}^n} \quad & a^\mathsf{T} b \\ \text{s.t.} \quad & ||b|| \leqslant 1. \end{aligned} \tag{7.8}$$

Lemma 7.7. ***Duals of Scaled Hölder p-Norms***
We denote by

$$||a||_{A,p} := ||Aa||_p = \left(\sum_{i=1}^{n} |(Aa)_i|^p \right)^{\frac{1}{p}} \tag{7.9}$$

the scaled Hölder p-Norm of $a \in \mathbb{R}^n$, with $1 \leqslant p \leqslant \infty$, and $A \in \mathcal{M}(n, \mathbb{R})$ non-singular. Its dual is the norm

$$||a||_* := ||a||_{A^{-1}, \frac{p}{1-p}} . \tag{7.10}$$

Here we set $\frac{p}{1-p} := \infty$ for $p = 1$, and $\frac{p}{1-p} := 1$ for $p = \infty$.

Proof. A proof may be found in Werner [58]. □

For uncertainty sets with mixed confidence types, observe also from this lemma that the dual of a sum of two norms (ℓ_1-norm) is the maximum of the respective dual norms,

$$||x|| := ||x_a||_{A,p} + ||x_b||_{B,q} \quad \Longrightarrow \quad ||x||_* := \max \left\{ ||x_a||_{A^{-1}, \frac{p}{1-p}} , ||x_b||_{B^{-1}, \frac{q}{1-q}} \right\}. \tag{7.11}$$

Vice versa, the dual of the maximum (ℓ_∞-norm) of two norms is the sum of the respective dual norms. The important point here is that we may explicitly give duals of such norms that we typically use when specifying uncertainty sets.
The solution to the convex problem may now be derived analytically, as claimed by the following theorem.

Theorem 7.8. ***Analytical Solution of the Worst-Case Choice Problem***
The optimal solution to the convex maximization problem (7.7a) defining $\tilde{W}_{\min}$ is

$$\tilde{W}_{\min}(\varphi, \boldsymbol{u}) := \varphi(\overline{\boldsymbol{x}}, \boldsymbol{u}) + \left|\left| \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^\mathsf{T} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^{-\mathsf{T}} \frac{\partial \varphi}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u})^\mathsf{T} \right|\right|_* . \tag{7.12}$$

where $||\cdot||_*$ denotes the dual of the norm defining the uncertainty set of the parameter $\boldsymbol{p}$.

Proof. Obvious after eliminating $\boldsymbol{\Delta x}$ in (7.7a) and applying Lemma 7.7. □

Having derived the analytical solution of the convex problem approximating the worst-case operator $W_{\min}$ we may now cast a linearized form of the robust counterpart problem (7.6a).

Definition 7.9. ***Linearized Robust Counterpart Problem***

$$\min_{\boldsymbol{u}, \overline{\boldsymbol{x}}} \quad f(\overline{\boldsymbol{x}}, \boldsymbol{u}) + \left|\left| \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^\mathsf{T} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^{-\mathsf{T}} \frac{\partial f}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u})^\mathsf{T} \right|\right|_* \tag{7.13a}$$

$$\text{s.t.} \quad h_i(\overline{\boldsymbol{x}}, \boldsymbol{u}) - \left|\left| \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^\mathsf{T} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^{-\mathsf{T}} \frac{\partial h_i}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u})^\mathsf{T} \right|\right|_* \geqslant 0, \quad i = 1, \ldots, n_h, \tag{7.13b}$$

$$\boldsymbol{g}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) = \boldsymbol{0}. \tag{7.13c}$$

It is important to note that all derivatives depend on the controls $\boldsymbol{u}$ and are thus subject to optimization. Standard Newton-Lagrange algorithms for the solution of this nonlinear program such as the SQP method presented in chapter 3 consequentially will require the computation of second-order derivatives, and mandate the existence of third-order derivatives as well.

7.2.2 Non-Smoothness of the Dual Norms

The dual norm appearing in the objective and inequality constraint functions of either the direct or the adjoint sensitivity formulation of the linearized robust counterpart problem introduces points of non-smoothness into the problem. It is to be expected that special precaution needs to be taken if the resulting NLPs are to be solved using standard Newton-Lagrange techniques.

Addressing Non-Smoothness of scaled ℓ_1 and ℓ_∞ Norms

Non-smoothness introduced by these norms can be overcome by simple slack variable approaches and is possible for both the direct and the adjoint sensitivity formulations. Since both application and implementation are restricted to ellipsoidal uncertainty sets associated with the ℓ_2 norm, we refer to Diehl et al. [13] for details on the slack formulation.

Addressing Non-Smoothness of other scaled ℓ_p Norms

For all other scaled Hölder norms $||a||_{A,p}$, $1 < p < \infty$, and in particular for the scaled Euclidean norm $||a||_{\Sigma^{-1},2}$ associated with ellipsoidal uncertainty sets, there exists no corresponding slack variable approach. However, for the ℓ_2 norm there also exists only a single point of non-differentiability in the origin $\boldsymbol{a} = 0$.

We observe that when treating the *DaimlerChrysler* powertrain optimal control problems with the MSSQP algorithm employing BFGS updates to the Hessian, it is not necessary to pay special attention to this non-differentiability. This observation is anticipated in Diehl et al. [13] for the case that it is impossible to reduce any of the first-order sensitivities of the objective (or the inequality constraint functions) to zero. The optimizer simply stays away from the single point of non-differentiability during the computed iterations. It is further justified by successful application to robust experimental design by Körkel et al. [35].

7.2.3 Sensitivities preserving Sparsity

The problem formulation (7.13) suffers from the drawback that due to the presence of an explicit inverse, any sparsity structure is likely to get lost in both the objective and the inequality constraint derivatives. More convenient and efficient formulations that maintain sparsity are available and shall now be presented.

Direct Sensitivities

Observe that the $n_x \times n_p$ derivative matrix $\boldsymbol{D}$

$$\boldsymbol{D}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) := \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}})^{-1} \cdot \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) \tag{7.14}$$

is common to both (7.13a) and (7.13b), and obviously satisfies the following matrix equality constraint

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) \cdot \boldsymbol{D}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}}, \boldsymbol{u}, \overline{\boldsymbol{p}}) = \boldsymbol{0}. \tag{7.15}$$

By introducing the derivative matrix $\boldsymbol{D}$ as an additional variable into the linearized robust counterpart problem, we arrive at the following linearized robust counterpart problem formulation using direct sensitivities.

Definition 7.10. ***Linearized Robust Counterpart Problem with Direct Sensitivities***

$$\min_{\boldsymbol{u},\overline{\boldsymbol{x}},\boldsymbol{D}} \quad f\,(\overline{\boldsymbol{x}},\boldsymbol{u}) + \left|\left|\boldsymbol{D}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{\mathsf{T}}\frac{\partial f}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u})^{\mathsf{T}}\right|\right|_* \tag{7.16a}$$

$$\text{s.t.} \quad h_i(\overline{\boldsymbol{x}},\boldsymbol{u}) - \left|\left|\boldsymbol{D}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{\mathsf{T}}\frac{\partial h_i}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u})^{\mathsf{T}}\right|\right|_* \geqslant 0, \quad i=1,\ldots,n_h, \tag{7.16b}$$

$$\boldsymbol{g}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) = \boldsymbol{0}, \tag{7.16c}$$

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})\cdot\boldsymbol{D}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) = \boldsymbol{0}. \tag{7.16d}$$

The last equation (7.16c) is a matrix equation in $\mathcal{M}(n_x \times n_p, \mathbb{R})$.

Adjoint Sensitivities

If the problem formulation satisfies $n_p > n_h$, i.e., there are more uncertain parameters than inequality constraints to be robustified, the following adjoint formulation may be preferred. We define adjoint sensitivities $\boldsymbol{\lambda_i}$ of the objective f and the inequality constraints h_i, $i = 1,\ldots,n_h$, by

$$\boldsymbol{\lambda_0}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) := \frac{\partial f}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u})\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{-1}, \tag{7.17a}$$

$$\boldsymbol{\lambda_i}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) := \frac{\partial h_i}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u})\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{-1}, \quad i=1,\ldots,n_h. \tag{7.17b}$$

Observe now that the adjoint sensitivity row vectors $\boldsymbol{\lambda_i}$ are again common to both (7.13a) and (7.13b), and satisfy the constraints

$$\boldsymbol{\lambda_0}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) - \frac{\partial f}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u}) = \boldsymbol{0}, \tag{7.18a}$$

$$\boldsymbol{\lambda_i}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) - \frac{\partial h_i}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u}) = \boldsymbol{0}, \quad i=1,\ldots,n_h. \tag{7.18b}$$

Introducing the adjoint sensitivities as optimization variables into the problem, we arrive at the following linearized robust counterpart problem formulation using adjoint sensitivities.

Definition 7.11. ***Linearized Robust Counterpart Problem with Adjoint Sensitivities***

$$\min_{\substack{\boldsymbol{u},\overline{\boldsymbol{x}},\\ \boldsymbol{\lambda_0},\ldots,\boldsymbol{\lambda_{n_h}}}} \quad f\,(\overline{\boldsymbol{x}},\boldsymbol{u}) + \left|\left|\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{\mathsf{T}}\boldsymbol{\lambda_0}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{\mathsf{T}}\right|\right|_* \tag{7.19a}$$

$$\text{s.t.} \quad h_i(\overline{\boldsymbol{x}},\boldsymbol{u}) - \left|\left|\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{\mathsf{T}}\boldsymbol{\lambda_i}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})^{\mathsf{T}}\right|\right|_* \geqslant 0, \quad i=1,\ldots,n_h, \tag{7.19b}$$

$$\boldsymbol{g}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) = \boldsymbol{0}, \tag{7.19c}$$

$$\boldsymbol{\lambda_0}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) - \frac{\partial f}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u}) = \boldsymbol{0}, \tag{7.19d}$$

$$\boldsymbol{\lambda_i}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}})\cdot\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u},\overline{\boldsymbol{p}}) - \frac{\partial h_i}{\partial \boldsymbol{x}}(\overline{\boldsymbol{x}},\boldsymbol{u}) = \boldsymbol{0}, \quad i=1,\ldots,n_h. \tag{7.19e}$$

7.3 Optimal Control of Uncertain Systems

In this section we briefly establish the connection between the above class of uncertain discrete nonlinear problems and the class of optimal control problems from Chapter 6. Although unnecessary from a theoretical point of view, and probably a bit more cumbersome, we adhere to the problem formulation treated by *MUSCOD-II*. Initial values are considered to be independent of the model parameters, i.e., $\frac{\partial \boldsymbol{x_0}}{\partial \boldsymbol{p}} = \boldsymbol{0}$. Consequentially we separately account for uncertainty in the initial values $\boldsymbol{x_0}$. For simplicity we again consider the whole vector $\boldsymbol{p}$ of model parameters and the whole vector $\boldsymbol{x_0}$ of initial values to be uncertain.

7.3.1 Problem Formulation

Analogous to the linearized counterpart problem formulation with direct sensitivities (7.16a) we may derive a linearized robust formulation of the optimal control problem class discussed in Chapter 6. The required sensitivities $\boldsymbol{G}_\mathrm{x}$ and $\boldsymbol{G}_\mathrm{p}$ of the now time-dependent states $\boldsymbol{x}(t)$ with respect to the uncertain parameters $\boldsymbol{p}$ and the uncertain initial values $\boldsymbol{x}_0$ are obtained by augmenting the IVP by the corresponding variational differential equations.

Definition 7.12. ***Robust Optimal Control Problem with Direct Sensitivities***

Robustified objective function of Mayer type:

$$\min_{\boldsymbol{p},\boldsymbol{u},\boldsymbol{x},\boldsymbol{G}} \quad \phi\left(t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f}), \boldsymbol{p}\right) + \left|\left| \boldsymbol{G}(t_\mathrm{f})^\mathsf{T} \frac{\partial \phi}{\partial \boldsymbol{x}}\Big(t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f}), \boldsymbol{p}\Big)^\mathsf{T} \right|\right|_* \tag{7.20a}$$

Robustified inequality constraints:

$$\text{s.t.} \qquad 0 \leqslant h_i(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) - \left|\left| \boldsymbol{G}(t_\mathrm{f})^\mathsf{T} \frac{\partial h_i}{\partial \boldsymbol{x}}\Big(t_\mathrm{f}, \boldsymbol{x}(t_\mathrm{f}), \boldsymbol{p}\Big)^\mathsf{T} \right|\right|_*, \; i = 1, \ldots, n_h, \tag{7.20b}$$

Initial-value problem defining the state trajectory of the dynamic process:

$$\frac{d\boldsymbol{x}}{dt}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \qquad t \in \mathcal{T} \tag{7.20c}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x_0}, \tag{7.20d}$$

Variational differential equations for direct sensitivities:

$$\frac{d\boldsymbol{G}_\mathrm{x}}{dt}(t) = \frac{d\boldsymbol{f}}{d\boldsymbol{x}}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big) \cdot \boldsymbol{G}_\mathrm{x}(t), \quad t \in \mathcal{T} \tag{7.20e}$$

$$\boldsymbol{G}_\mathrm{x}(t_0) = \boldsymbol{I}, \tag{7.20f}$$

$$\frac{d\boldsymbol{G}_\mathrm{p}}{dt}(t) = \frac{d\boldsymbol{f}}{d\boldsymbol{x}}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big) \cdot \boldsymbol{G}_\mathrm{p}(t) + \frac{d\boldsymbol{f}}{d\boldsymbol{p}}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big), \tag{7.20g}$$

$$\boldsymbol{G}_\mathrm{p}(t_0) = \boldsymbol{0}, \tag{7.20h}$$

Implicitly defined discontinuities:

$$\boldsymbol{x}(t_\mathrm{s}^+) = \boldsymbol{\Delta}_j\left(t_\mathrm{s}, \boldsymbol{x}(t_\mathrm{s}^-), \boldsymbol{u}(t_\mathrm{s}^-), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t_\mathrm{s}^-)\right), \quad \forall t_\mathrm{s} \in \mathcal{T} : \; \exists j \in \mathbb{N} : \; \sigma_j(t_\mathrm{s}^-) = 0, \tag{7.20i}$$

First-order sensitivity updates in discontinuities:

$$\boldsymbol{G}_\mathrm{x}(t_\mathrm{s}^+) = \boldsymbol{U}_\mathrm{x} \boldsymbol{G}_\mathrm{x}(t_\mathrm{s}^-), \tag{7.20j}$$

$$\boldsymbol{G}_\mathrm{p}(t_\mathrm{s}^+) = \boldsymbol{U}_\mathrm{x} \boldsymbol{G}_\mathrm{p}(t_\mathrm{s}^-) + \boldsymbol{U}_\mathrm{p}. \tag{7.20k}$$

Here t_s^- and t_s^+ denote the left-hand vs. right-hand limit in t_s with respect to time. The implicit switch formalism used in Eq. (7.20h) to (7.20j) is presented in detail in Chapter 4. The complete matrix $\boldsymbol{G}(t) \in \mathcal{M}(n_x \times (n_x + n_p), \mathbb{R})$ of sensitivities with respect to the uncertain values is defined by

$$\boldsymbol{G}(t) := \left[\begin{array}{cc} \boldsymbol{G}_\mathrm{x}(t) & \boldsymbol{G}_\mathrm{p}(t) \end{array} \right]. \tag{7.21}$$

7.3.2 Generation of Direct Second-Order Sensitivities

As already hinted, the sensitivities of first order $\boldsymbol{G}_\mathrm{x}(t_\mathrm{f})$ and $\boldsymbol{G}_\mathrm{p}(t_\mathrm{f})$ at the end of the time horizon $\mathcal{T} = [t_0, t_\mathrm{f}]$ depend on the control profile $\boldsymbol{u}(t)$ and are thus subject to optimization. Since standard Newton-Lagrange algorithms such as the SQP method presented in Chapter 3 require derivatives of all optimization variables, the need for second-order sensitivities of the solution $\boldsymbol{x}(t)$ with respect to the uncertain parameters $\boldsymbol{p}$ and the uncertain initial values $\boldsymbol{x}_0$ arises. We discuss two possibilities to generate these sensitivities.

Variational Differential Equations

In the spirit of the discussion of first-order sensitivity generation in Chapter 4, and assuming sufficient smoothness of the right-hand side function $\boldsymbol{f}$, we can derive second-order variational differential equations (cf. Bauer [2]). These may then be solved along with the nominal solutions and the first-order sensitivity equations.

To avoid having to introduce tensor notations, we start by considering the second-order sensitivity g_{xx}^{ijk} of a scalar component $x_i(t)$ of the differential state $\boldsymbol{x}(t)$ with respect to two single scalar components $x_{0,j}$ and $x_{0,k}$ of the initial value vector $\boldsymbol{x_0}$. We also introduce second-order sensitivity matrices $\boldsymbol{G}_{\mathrm{xx}}^{j}$ associated with a selected first-order sensitivity (central index j). In the same spirit we obtain second-order sensitivities of the solution with respect to the parameters $\boldsymbol{p}$ and mixed second-order sensitivities.

$$g_{\mathrm{xx}}^{ijk}(t) := \frac{\partial^2 x_i}{\partial x_{0,k} \partial x_{0,j}}(t;\ t_0, \boldsymbol{x_0}, \boldsymbol{p}), \qquad \boldsymbol{G}_{\mathrm{xx}}^{j}(t) := \begin{bmatrix} g_{\mathrm{xx}}^{1,j,1} & \cdots & g_{\mathrm{xx}}^{1,j,n_x} \\ \vdots & \ddots & \vdots \\ g_{\mathrm{xx}}^{n_x,j,1} & \cdots & g_{\mathrm{xx}}^{n_x,j,n_x} \end{bmatrix}, \tag{7.22a}$$

$$g_{\mathrm{pp}}^{ijk}(t) := \frac{\partial^2 x_i}{\partial p_k \partial p_j}(t;\ t_0, \boldsymbol{x_0}, \boldsymbol{p}), \qquad \boldsymbol{G}_{\mathrm{pp}}^{j}(t) := \begin{bmatrix} g_{\mathrm{pp}}^{1,j,1} & \cdots & g_{\mathrm{pp}}^{1,j,n_p} \\ \vdots & \ddots & \vdots \\ g_{\mathrm{pp}}^{n_x,j,1} & \cdots & g_{\mathrm{pp}}^{n_x,j,n_p} \end{bmatrix}, \tag{7.22b}$$

$$g_{\mathrm{xp}}^{ijk}(t) := \frac{\partial^2 x_i}{\partial p_k \partial x_{0,j}}(t;\ t_0, \boldsymbol{x_0}, \boldsymbol{p}), \qquad \boldsymbol{G}_{\mathrm{xp}}^{j}(t) := \begin{bmatrix} g_{\mathrm{xp}}^{1,j,1} & \cdots & g_{\mathrm{xp}}^{1,j,n_p} \\ \vdots & \ddots & \vdots \\ g_{\mathrm{xp}}^{n_x,j,1} & \cdots & g_{\mathrm{xp}}^{n_x,j,n_p} \end{bmatrix}. \tag{7.22c}$$

These sensitivities are solutions of the following second-order variational differential equations. We use the notations $\boldsymbol{G}_{\mathrm{x}} = \left[g_{\mathrm{x}}^{ij}\right]_{ij}$ and $\boldsymbol{G}_{\mathrm{p}} = \left[g_{\mathrm{p}}^{ij}\right]_{ij}$ for the first-order sensitivity matrices.

$$\frac{dg_{\mathrm{xx}}^{ijk}}{dt}(t) = g_{\mathrm{x}}^{ik} \frac{\partial^2 f_i}{\partial x_j^2} g_{\mathrm{x}}^{ij} + \frac{\partial f_i}{\partial x_j} g_{\mathrm{xx}}^{ijk}, \tag{7.23a}$$

$$\frac{dg_{\mathrm{pp}}^{ijk}}{dt}(t) = \left(\frac{\partial^2 f_i}{\partial x_j^2} g_{\mathrm{p}}^{ik} + \frac{\partial^2 f_i}{\partial p_k \partial x_j} \right) g_{\mathrm{p}}^{ij} + \frac{\partial^2 f_i}{\partial x_j \partial p_j} g_{\mathrm{p}}^{ik} + \frac{\partial f_i}{\partial x_j} g_{\mathrm{pp}}^{ijk} + \frac{\partial^2 f_i}{\partial p_k \partial p_j}, \tag{7.23b}$$

$$\frac{dg_{\mathrm{xp}}^{ijk}}{dt}(t) = \left(\frac{\partial^2 f_i}{\partial x_j^2} g_{\mathrm{p}}^{ik} + \frac{\partial^2 f_i}{\partial p_k \partial x_i} \right) g_{\mathrm{x}}^{ij} + \frac{\partial f_i}{\partial x_j} g_{\mathrm{xp}}^{ijk}, \tag{7.23c}$$

all with initial values set to zero. Note that, given that H.A. Schwarz' theorem is satisfied, we have $g_{\mathrm{px}}^{ijk}(t) = g_{\mathrm{xp}}^{ikj}(t)$. To ease the notation, we omitted the arguments of the sensitivity matrices, as well as those of the right-hand side function $\boldsymbol{f}$.

Augmented System

Looking at the internal architecture of the optimal control software package *MUSCOD-II* we favored the following approach over the explicit solution of second-order sensitivity equations.

Since the sensitivities are subject to optimization, there must exist entries of the state vector $\boldsymbol{x}$ corresponding to the sensitivity matrix entries. Thus it is anyway necessary to solve an ODE model $\boldsymbol{f}$ augmented by the first-order variational differential equations. Consequentially, this augmentation is automatically performed by the *ROBUST* framework extension developed to implement robust optimization in *MUSCOD-II*.

Then, from the integrator's point of view, the model function $\boldsymbol{f}$ *already contains* functionality to generate first-order sensitivities. It is straightforward to let the integrator generate first-order sensitivities of this augmented system. In fact, any integrator found within *MUSCOD-II*

does this anyway as described in the discussion of sensitivity generation found in Chapter 4. As a result we obtain second-order sensitivities by *twice* generating first-order sensitivities — once within the model, and once within the integrator.

7.3.3 Improving Efficiency of the Augmented System Approach

One drawback of the augmented system approach is that we generate a lot of non-existent sensitivity information as well. Let $\hat{\boldsymbol{x}} \in \mathbb{R}^{n_x(1+n_x+n_p)}$ denote the augmented system's state

$$\hat{\boldsymbol{x}}(t) := \left[\begin{array}{ccc} \boldsymbol{x}(t)^\mathsf{T} & (\operatorname{vec} \boldsymbol{G}_\mathrm{x}(t))^\mathsf{T} & (\operatorname{vec} \boldsymbol{G}_\mathrm{p}(t))^\mathsf{T} \end{array} \right]^\mathsf{T}, \tag{7.24}$$

with initial values $\hat{\boldsymbol{x}}_\mathbf{0}$

$$\hat{\boldsymbol{x}}_\mathbf{0} := \left[\begin{array}{ccc} \boldsymbol{x}_\mathbf{0}{}^\mathsf{T} & (\operatorname{vec} \boldsymbol{G}_\mathrm{x}(t_0))^\mathsf{T} & (\operatorname{vec} \boldsymbol{G}_\mathrm{p}(t_0))^\mathsf{T} \end{array} \right]^\mathsf{T}, \tag{7.25}$$

where the vectorizing operator vec $\boldsymbol{A}$ indicates that a column vector should be composed from the individual columns of the matrix $\boldsymbol{A}$. Note that, since their elements are included in the multiple shooting state vectors $\boldsymbol{s}_i$ (6.20), the initial sensitivity matrices $\boldsymbol{G}_\mathrm{x}(t_0)$ and $\boldsymbol{G}_\mathrm{p}(t_0)$ for a subinterval equal $\boldsymbol{I}$ and $\mathbf{0}$ respectively on the first multiple shooting interval only.
Generating first-order sensitivities of this augmented system with respect to $\hat{\boldsymbol{x}}_\mathbf{0}$ and $\boldsymbol{p}$, as is done by the standard integrators, results in a sparse matrix as shown on the opposite page.

$$\frac{\partial \hat{\boldsymbol{x}}}{\partial \hat{\boldsymbol{x}}_\mathbf{0}}(t, t_0, \hat{\boldsymbol{x}}_\mathbf{0}, \boldsymbol{p}) = \left[\begin{array}{c|ccc|ccc} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{G}^1_{\mathrm{x}_0}} & \cdots & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{G}^{n_x}_{\mathrm{x}_0}} & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{G}^1_{\mathrm{p}_0}} & \cdots & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{G}^{n_p}_{\mathrm{p}_0}} \\ \hline \frac{\partial \boldsymbol{G}^1_\mathrm{x}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{G}^1_\mathrm{x}}{\partial \boldsymbol{G}^1_{\mathrm{x}_0}} & \cdots & \frac{\partial \boldsymbol{G}^1_\mathrm{x}}{\partial \boldsymbol{G}^{n_x}_{\mathrm{x}_0}} & \frac{\partial \boldsymbol{G}^1_\mathrm{x}}{\partial \boldsymbol{G}^1_{\mathrm{p}_0}} & \cdots & \frac{\partial \boldsymbol{G}^1_\mathrm{x}}{\partial \boldsymbol{G}^{n_p}_{\mathrm{p}_0}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{G}^{n_x}_\mathrm{x}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{G}^{n_x}_\mathrm{x}}{\partial \boldsymbol{G}^1_{\mathrm{x}_0}} & \cdots & \frac{\partial \boldsymbol{G}^{n_x}_\mathrm{x}}{\partial \boldsymbol{G}^{n_x}_{\mathrm{x}_0}} & \frac{\partial \boldsymbol{G}^{n_x}_\mathrm{x}}{\partial \boldsymbol{G}^1_{\mathrm{p}_0}} & \cdots & \frac{\partial \boldsymbol{G}^{n_x}_\mathrm{x}}{\partial \boldsymbol{G}^{n_p}_{\mathrm{p}_0}} \\ \hline \frac{\partial \boldsymbol{G}^1_\mathrm{p}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{G}^1_\mathrm{p}}{\partial \boldsymbol{G}^1_{\mathrm{x}_0}} & \cdots & \frac{\partial \boldsymbol{G}^1_\mathrm{p}}{\partial \boldsymbol{G}^{n_x}_{\mathrm{x}_0}} & \frac{\partial \boldsymbol{G}^1_\mathrm{p}}{\partial \boldsymbol{G}^1_{\mathrm{p}_0}} & \cdots & \frac{\partial \boldsymbol{G}^1_\mathrm{p}}{\partial \boldsymbol{G}^{n_p}_{\mathrm{p}_0}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{G}^{n_p}_\mathrm{p}}{\partial \boldsymbol{x}_0} & \frac{\partial \boldsymbol{G}^{n_p}_\mathrm{p}}{\partial \boldsymbol{G}^1_{\mathrm{x}_0}} & \cdots & \frac{\partial \boldsymbol{G}^{n_p}_\mathrm{p}}{\partial \boldsymbol{G}^{n_x}_{\mathrm{x}_0}} & \frac{\partial \boldsymbol{G}^{n_p}_\mathrm{p}}{\partial \boldsymbol{G}^1_{\mathrm{p}_0}} & \cdots & \frac{\partial \boldsymbol{G}^{n_p}_\mathrm{p}}{\partial \boldsymbol{G}^{n_p}_{\mathrm{p}_0}} \end{array} \right] \tag{7.26a}$$

$$= \left[\begin{array}{c|ccc|ccc} \boldsymbol{G}_\mathrm{x} & \multicolumn{3}{c|}{\mathbf{0}} & \multicolumn{3}{c}{\mathbf{0}} \\ \hline \boldsymbol{G}^1_\mathrm{xx} & \multicolumn{3}{c|}{\multirow{3}{*}{$\boldsymbol{G}_\mathrm{x}$}} & \multicolumn{3}{c}{} \\ \vdots & \multicolumn{3}{c|}{} & \multicolumn{3}{c}{\mathbf{0}} \\ \boldsymbol{G}^{n_x}_\mathrm{xx} & \multicolumn{3}{c|}{} & \multicolumn{3}{c}{} \\ \hline \boldsymbol{G}^1_\mathrm{px} & \multicolumn{3}{c|}{} & \multicolumn{3}{c}{\multirow{3}{*}{$\boldsymbol{G}_\mathrm{x}$}} \\ \vdots & \multicolumn{3}{c|}{\mathbf{0}} & \multicolumn{3}{c}{} \\ \boldsymbol{G}^{n_p}_\mathrm{px} & \multicolumn{3}{c|}{} & \multicolumn{3}{c}{} \end{array} \right] \tag{7.26b}$$

$$\frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{p}}(t, t_0, \hat{\boldsymbol{x}}_\mathbf{0}, \boldsymbol{p}) = \left[\begin{array}{c|ccc|ccc} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}^\mathsf{T} & \frac{\partial \boldsymbol{G}^1_\mathrm{x}}{\partial \boldsymbol{p}}^\mathsf{T} & \cdots & \frac{\partial \boldsymbol{G}^{n_x}_\mathrm{x}}{\partial \boldsymbol{p}}^\mathsf{T} & \frac{\partial \boldsymbol{G}^1_\mathrm{p}}{\partial \boldsymbol{p}}^\mathsf{T} & \cdots & \frac{\partial \boldsymbol{G}^{n_p}_\mathrm{p}}{\partial \boldsymbol{p}}^\mathsf{T} \end{array} \right]^\mathsf{T} \tag{7.26c}$$

$$= \left[\begin{array}{c|ccc|ccc} \boldsymbol{G}^\mathsf{T}_\mathrm{p} & \boldsymbol{G}^1_\mathrm{xp}{}^\mathsf{T} & \cdots & \boldsymbol{G}^{n_x}_\mathrm{xp}{}^\mathsf{T} & \boldsymbol{G}^1_\mathrm{pp}{}^\mathsf{T} & \cdots & \boldsymbol{G}^{n_p}_\mathrm{pp}{}^\mathsf{T} \end{array} \right]^\mathsf{T} \tag{7.26d}$$

For brevity, we omit the dependencies; superscript indices denote sensitivity matrix columns; blank entries are zero. The presented sparsity pattern can easily be verified from an analysis of the interdependencies of the respective first- and second-order variational differential equations. It is obvious that the sensitivity $\boldsymbol{G}^i_\mathrm{x}$ of the solution with respect to the initial value

component $x_{0,i}$ is independent of that with respect to another component $x_{0,j}$, therefore we find

$$\frac{\partial \boldsymbol{G}_{\mathrm{x}}^{i}}{\partial \boldsymbol{G}_{\mathrm{x}}^{j}} = \mathbf{0} \quad \text{for } i \neq j. \tag{7.27}$$

The same is true for any pair of global model parameters p_i and p_j, and equally well for mixed sensitivities. For the actual second-order sensitivities with respect to the initial values and parameters remaining on the diagonal of the above matrix, we look at the underlying sensitivity IVP

$$\begin{aligned} \frac{d\boldsymbol{G}_{\mathrm{xx}}}{dt}(t, \boldsymbol{x_0}, \boldsymbol{p}) &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(t, \boldsymbol{x}(t), \boldsymbol{p}) \cdot \boldsymbol{G}_{\mathrm{x}}(t), & \text{(7.28a)} \\ \boldsymbol{G}_{\mathrm{xx}}(t_0) &= \boldsymbol{G_0}. & \text{(7.28b)} \end{aligned}$$

From its explicit solution

$$\boldsymbol{G}_{\mathrm{xx}}(T) = \boldsymbol{G}_{\mathrm{x}}(T) \cdot \boldsymbol{G_0}, \quad T > t_0, \tag{7.29}$$

it becomes clear that the diagonal of the above matrix consists of identical block matrices. The same is true for the sensitivities $\boldsymbol{G}_{\mathrm{pp}}$ with respect to model parameters $\boldsymbol{p}$. Therefore, the number of sensitivities that actually need to be computed can be reduced from $\mathcal{O}(n_x^2)$ to the intuitively necessary complexity of $\mathcal{O}(n_x)$.

7.3.4 Second-Order Sensitivity Updates in Implicit Discontinuities

We observed that completely ignoring the theoretical need for second-order sensitivity updates in discontinuities of the ODE model leads to a number of problems. The SQP method frequently failed to converge, or ended up in unreliable results of inferior quality only. In this section we therefore have a look at how to efficiently implement the cumbersome second-order sensitivity updates in implicit discontinuities.

Second-Order Update Formulas

Analogous to the derivation found in Chapter 4, explicit sensitivity updates for the second-order sensitivities are derived by Mombaur [42]. They are, however, of extremely complicated shape, require the computation of an excessively large number of derivatives, and their implementation is likely to be prone to many errors. Consequently, Mombaur [42] concludes that it is doubtful whether an attempt at a direct implementation of these updates would be beneficial.

First-Order Update of the Augmented System

For the augmented system approach we present another realization of second-order sensitivity updates in implicit discontinuities. Since the integrator code presented in Chapter 4 implements the first-order sensitivity updates anyway, it is straightforward to apply this update *twice*. The first application to the sensitivities $\boldsymbol{G}_{\mathrm{x}}$ and $\boldsymbol{G}_{\mathrm{p}}$ contained within the augmented state $\hat{\boldsymbol{x}}$ properly updates the first-order sensitivities. It also possibly extends the discontinuity of $\boldsymbol{x}$ to the sensitivity part of the augmented state $\hat{\boldsymbol{x}}$. This first application has to be performed by the augmented model.
The second application is performed by the integrator anyway. Hiding the fact that a part of the augmented state $\hat{\boldsymbol{x}}$ actually holds first-order sensitivity information, the integrator updates first-order sensitivities of the augmented state as usual, thereby in fact producing a proper second-order sensitivity update.
Careful design of the code allows for re-use of the first-order sensitivity update code to perform both updates.

7.3.5 Summary of the Augmented System Approach Algorithm

We give a list of the important points of the sensitivity generation and implicit discontinuity update algorithm for the augmented state approach from the integrator's point of view here.

1. The integrator maintains an augmented state $\hat{\boldsymbol{x}} = \left[\begin{array}{ccc} \boldsymbol{x} & \boldsymbol{G}_{\mathrm{x}} & \boldsymbol{G}_{\mathrm{p}} \end{array} \right]$ along with the corresponding (here unnamed) first-order sensitivities of $\hat{\boldsymbol{x}}$. It does not need insight into the fact that the augmented state contains sensitivity components itself.

2. Evaluation of the augmented right-hand side $\hat{\boldsymbol{f}}$ yields the derivative $\frac{d\boldsymbol{x}}{dt}$, as did the original right-hand side function $\boldsymbol{f}$. Furthermore, it also yields the sensitivity derivatives $\frac{d\boldsymbol{G}_{\mathrm{x}}}{dt}$ and $\frac{d\boldsymbol{G}_{\mathrm{p}}}{dt}$ of the plain state, obtained from evaluation of the variational differential equations.

3. Sensitivity information for the augmented state $\hat{\boldsymbol{x}}$ is generated by internal numerical differentiation (IND), cf. Bock [4, 5, 6], or again by simultaneous solution of the variational differential equations, cf. Chapter 4. To improve efficiency, we may exploit our insight into the structure of the augmented state here, as detailed in Section 7.3.3.

4. If a switch (implicit discontinuity) is detected, we require updates of the first and second-order sensitivities. We therefore perform the first-order sensitivity update on the sensitivities of the plain state $\boldsymbol{x}$, found as $\boldsymbol{G}_{\mathrm{x}}$, $\boldsymbol{G}_{\mathrm{p}}$ within the augmented state $\hat{\boldsymbol{x}}$. Afterwards, we perform the same first-order update on the (here unnamed) first-order sensitivities of the augmented state. Careful design allows to use the same code here.

The direct sensitivity approach has been implemented in the *ROBUST* framework extension to the optimal control software package *MUSCOD-II*. Some technical details can be found in Appendix C. The support of implicit discontinuities has been implemented within the explicit Runge-Kutta integrator *RKFSWT* presented in Chapter 4.

7.4 Robust Optimal Control of a Powertrain

In the final section of this chapter we apply the presented numerical method for robust optimal control to the guiding example of powertrain oscillations. We derive a subset problem from the powertrain optimal control problem described and solved in Chapter 6. Analysis of the nominal solution's sensitivity with respect to selected uncertainties motivates the need for robust solutions to this problem. The presented linearized approach yields engine control schemes which are proven to be more robust against deviations of the selected parameters.

7.4.1 Overview

We saw from the nominal optimal control results that the main difficulty of the optimization problem lies in the satisfaction of the implicit rotation speed delta constraint. We can further see from the graphs of the various results that a significant acceleration of the car can be observed only *after* the implicitly defined point in time has been passed.
We therefore choose to separate the constraint fulfillment problem from the problem of diminishing powertrain oscillations (Engelhard [17]). Starting in coasting mode, we strive to accelerate the powertrain into traction mode in the fastest possible way, while fulfilling the rotation speed delta constraint. The car's acceleration will start to increase only after the powertrain has arrived in traction mode. We may thus detect traction mode checking for a positive acceleration level a_f that is still well beyond the final acceleration level that is subject to oscillations.

7.4.2 Problem Formulation

The optimal control problem formulation resulting from these findings is given in Eq. (7.30a). It is robustified according to the linearized counterpart problem with direct sensitivities (7.16a).

$$\min_{\boldsymbol{u},\boldsymbol{x},t_\mathrm{f}} \quad t_\mathrm{f} + \int_{t_0}^{t_\mathrm{f}} \left(\gamma_1 u_1^2(t) + \gamma_2 u_2^2(t)\right)\, dt \tag{7.30a}$$

$$\text{s.t.} \quad a_\mathrm{car}(t_\mathrm{f}) \geqslant a_\mathrm{f}, \tag{7.30b}$$

$$\Delta\omega_\mathrm{max} \geqslant \frac{1}{i_\mathrm{gb} i_\mathrm{tr}} \omega_{\mathrm{dmf},1}(t_\Delta) - \omega_\mathrm{wh}(t_\Delta), \tag{7.30c}$$

$$\frac{d\boldsymbol{x}}{dt}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn}\boldsymbol{\sigma}(t)\big), \quad t \in \mathcal{T}, \tag{7.30d}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{7.30e}$$

$$u_1(t) \in [\underline{u}_1, \overline{u}_1], \tag{7.30f}$$

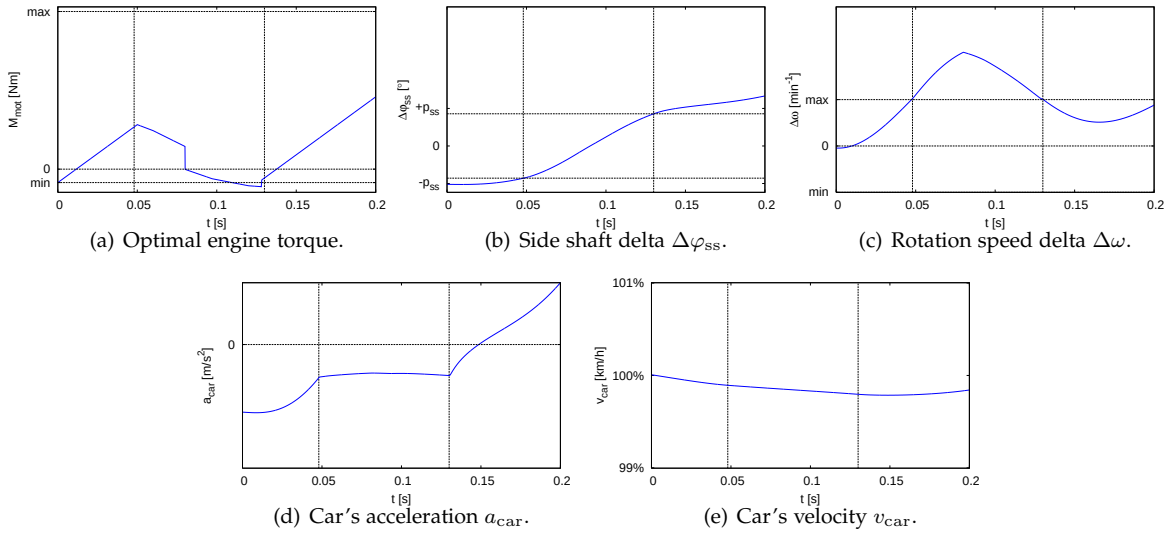
$$\frac{du_1}{dt}(t) \in [\underline{\delta u}_1, \overline{\delta u}_1] \quad \text{on} \quad (t_i, t_{i+1}) \subset \mathcal{T},\ i = 0, \ldots, m-1, \tag{7.30g}$$

$$u_2(t) \in [0, 1]. \tag{7.30h}$$

The unique point $t_\Delta \in \mathcal{T}$ is implicitly defined by $\Delta\varphi_\mathrm{ss}(t_\Delta) = +p_\mathrm{ss}$. This implicit constraint is realized as described in Section 6.3.2. Further implicit switches introduced by model non-differentiabilities are omitted from the problem formulation (7.30a) for simplicity, and can be found in the model description Chapter 2. The problem is well-defined only when starting in coasting mode ($\Delta\varphi_\mathrm{ss}(t_0) < -p_\mathrm{ss}$). Both controls are regularized by inclusion in the objective using appropriately chosen scalar values γ_1 and γ_2.
The nominal non-robust result of this optimal control problem for $n_\mathrm{mot}(t_0) = 3000\ \mathrm{min}^{-1}$ is presented in Fig. 7.1. The controls were initialized to a rough scheme taken from the optimal results presented in Chapter 6. For non-disclosure reasons, all graphs of simulated data remain unlabeled on the ordinate.

Scenario Fig.	KKT Tolerance	Scaled Infeasibility	Lagrange Gradient	Number of Iterations
7.1	$3.7 \cdot 10^{-8}$	$6.09 \cdot 10^{-11}$	$2.22 \cdot 10^{-4}$	12

Tab. 7.1: Acceleration into traction mode: Numerical quality of the nominal result. Refer to Diehl et al. [12], Leineweber [38] for a detailed discussion of these values.



(a) Optimal engine torque. (b) Side shaft delta $\Delta\varphi_{\mathrm{ss}}$. (c) Rotation speed delta $\Delta\omega$.

(d) Car's acceleration a_{car}. (e) Car's velocity v_{car}.

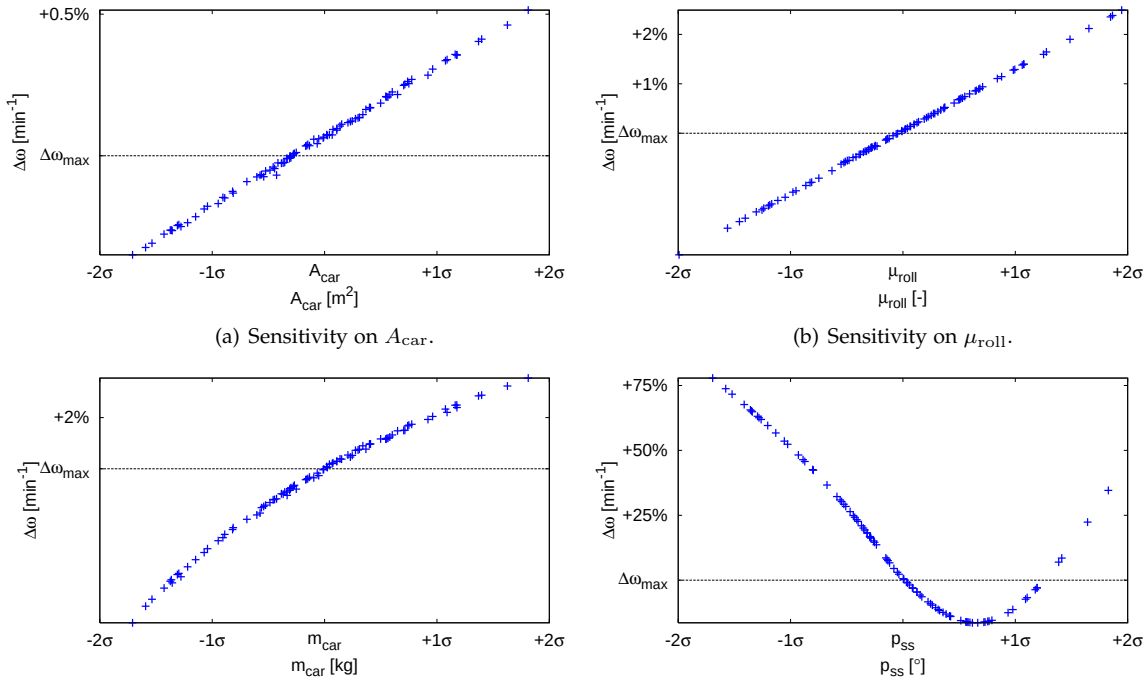
Fig. 7.1: Accelerating the powertrain into traction mode at $n_{\mathrm{mot}}(t_0) = 3000$ min^{-1}.

7.4.3 Sensitivity Analysis

We analyze the sensitivity of this nominal optimal solution against various uncertainties selected by our cooperating partner *DaimlerChrysler AG*. The selected standard deviations are assumed to be reasonable from practical experience of our cooperation partner. For the formulation of the optimization problem, however, it is important that even extreme deviations of the powertrain's internal states $\Delta\varphi_{\mathrm{dmf}}$ and $\Delta\varphi_{\mathrm{ss}}$ remain in coasting mode. We disturb the respective parameters and initial values by a random normally distributed error with zero mean and using selected standard deviations, but apply the controlled engine torque trajectory that has been optimized for the parameter's mean value. We take 100 samples of $\Delta\omega(t_\Delta)$ in order to obtain a good representation of the distribution of the resulting constraint violations (7.30b).

Parameter	Std. Dev.	Unit	Description
A_{car}	5%	m^2	Car's abutting face.
μ_{roll}	10%	–	Coefficient of rolling friction.
m_{car}	20%	kg	Car's mass.
p_{ss}	10%	°	Side shaft play size.
u_{sca}	10%	–	Scaler to the controlled engine torque.
u_{ofs}	10%	Nm	Offset to the controlled engine torque.
$\Delta\varphi_{\mathrm{dmf}}(t_0)$	20%	rad	Initial torsion of the DMF.
$\Delta\varphi_{\mathrm{ss}}(t_0)$	20%	rad	Initial torsion of the side shaft.

Tab. 7.2: Evaluated uncertainties for the robust powertrain acceleration problem.



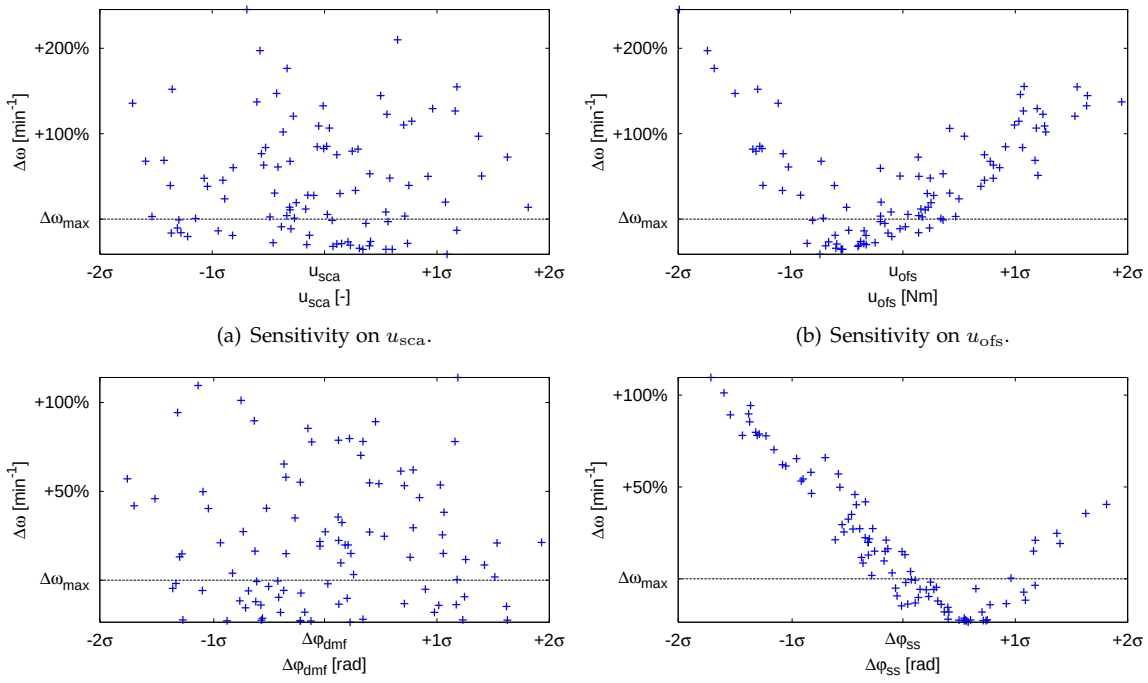
(a) Sensitivity on A_{car}.

(b) Sensitivity on μ_{roll}.

(c) Sensitivity on m_{car}.

(d) Sensitivity on p_{ss}.

Fig. 7.2: Sensitivity analysis for the rotation speed delta constraint $\Delta\omega(t_\Delta) \leqslant \Delta\omega_{\max}$. The plots show constraint violations in percent of $\Delta\omega_{\max}$ for 100 random samples.



(a) Sensitivity on u_{sca}.

(b) Sensitivity on u_{ofs}.

(c) Sensitivity on $\Delta\varphi_{\mathrm{dmf}}(t_0)$.

(d) Sensitivity on $\Delta\varphi_{\mathrm{ss}}(t_0)$.

Fig. 7.3: Constraint sensitivity analysis. Two related uncertain parameters are disturbed at once, and the projections of the constraint onto either one's space are shown.

7.4.4 Robust Results

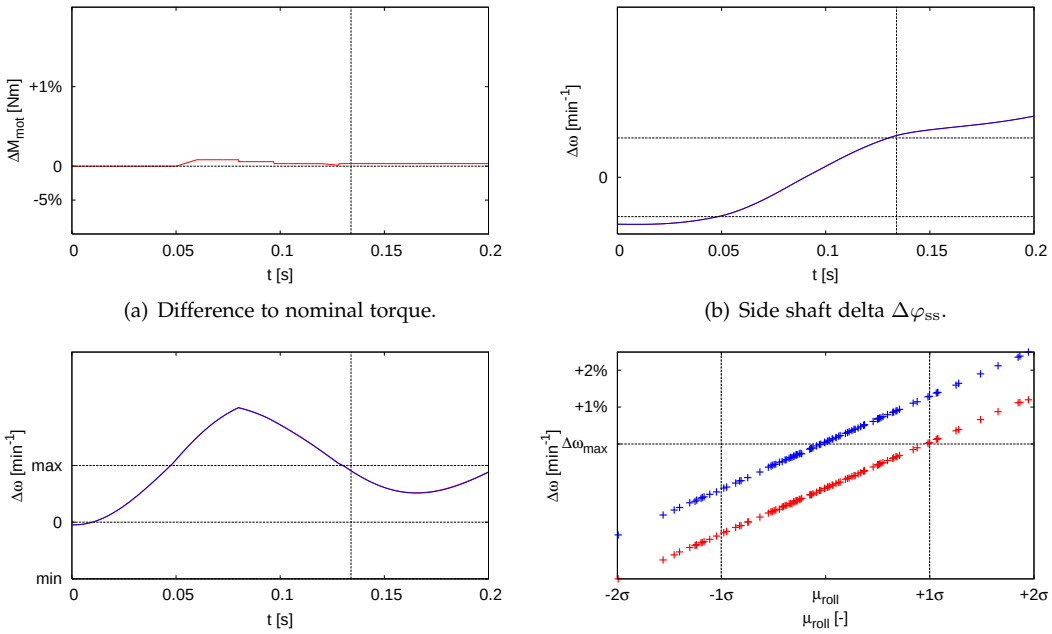
On the following pages we present robustified engine control schemes obtained from solving the above problem with the numerical method for nonlinear robust optimal control presented in this chapter. Tab. 7.3 shows some data on the numerical quality of the obtained solutions.

Scenario		KKT	Scaled	Lagrange	Number of
Fig.	Problem	Tolerance	Infeasibility	Gradient	Iterations
7.4	μ_{roll}	$7.45 \cdot 10^{-6}$	$3.04 \cdot 10^{-4}$	$1.75 \cdot 10^{-3}$	5
7.5	p_{ss}	$6.47 \cdot 10^{-6}$	$1.25 \cdot 10^{-4}$	$1.75 \cdot 10^{-2}$	6
7.6	u_{sca}, u_{ofs}	$1.95 \cdot 10^{-6}$	$5.40 \cdot 10^{-5}$	$1.17 \cdot 10^{-3}$	16
7.7	$\Delta\varphi_{\mathrm{dmf}}(t_0)$, $\Delta\varphi_{\mathrm{ss}}(t_0)$	$6.76 \cdot 10^{-7}$	$1.92 \cdot 10^{-4}$	$2.52 \cdot 10^{-3}$	23

Tab. 7.3: Robust acceleration into traction mode: Qualities of the solutions. The acceptable KKT tolerance was set to 10^{-5}. Refer to Diehl et al. [12], Leineweber [38] for a detailed discussion of these values.

In addition to the engine control schemes obtained from solving the robust optimal control problems, we present the difference to the nominal solution. For the robust results, we again performed a sensitivity analysis as described in the previous section. The results are shown together with the nominal results to stress the achieved sensitivity reductions. For multi-parameter scenarios, we analyze the sensitivity with respect to a single selected uncertain parameter within a range $[-2\sigma, 2\sigma]$ of two standard deviations around the mean. The other uncertain parameters stay within $[-1\sigma, 1\sigma]$, the uncertainty set taken into account by the formulated robust optimization problem.

Uncertainty in the Rolling Friction Coefficient



(a) Difference to nominal torque.

(b) Side shaft delta $\Delta\varphi_{\mathrm{ss}}$.

(c) Rotation speed delta $\Delta\omega$.

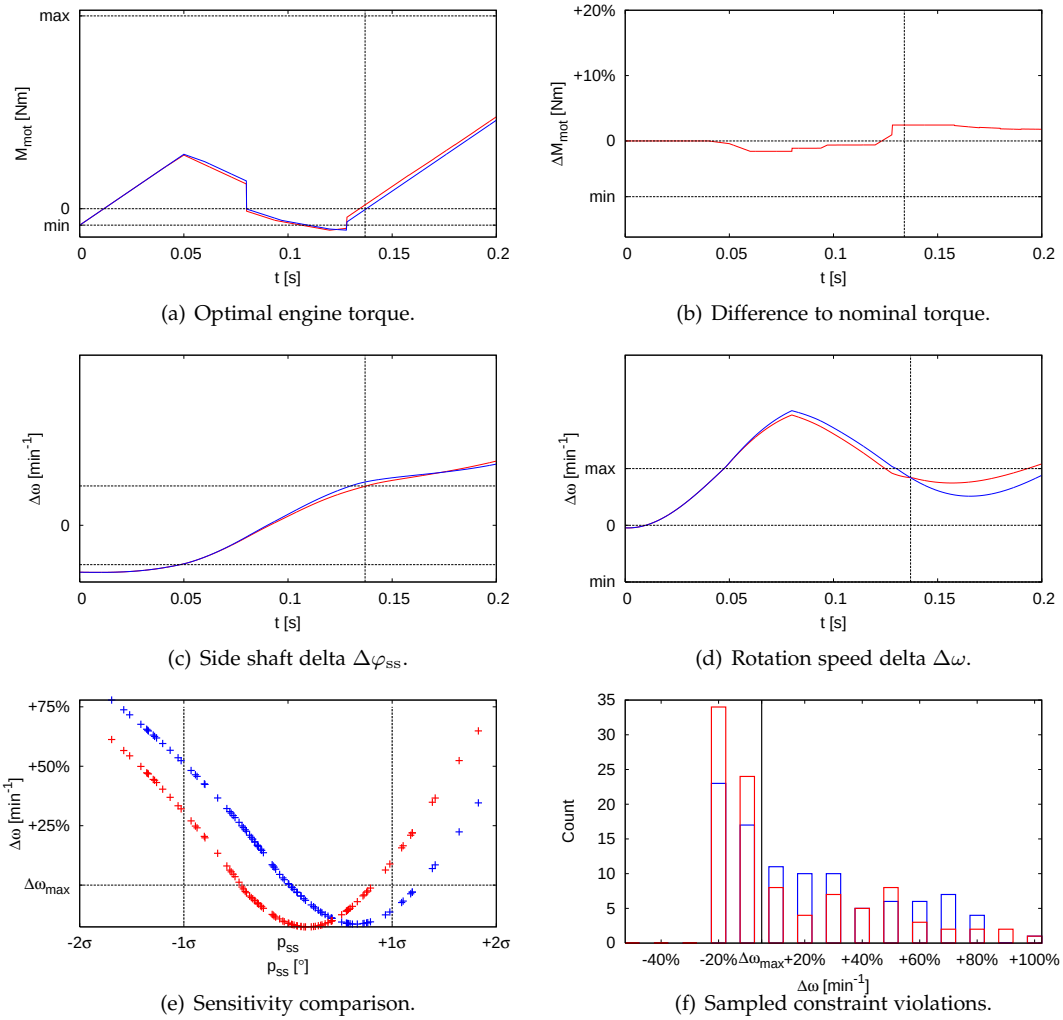
(d) Sensitivity comparison.

Fig. 7.4: Robust result for uncertainty in the rolling friction coefficient μ_{roll}. (——) indicates the nominal solution of Fig. 7.1, while (——) indicates the robust solution.

Although the constraint is not particularly sensitive to reasonable deviations in the rolling friction coefficient μ_{roll}, this result illustrates two points. First, the model appears to be linear in this region, so the linearized approach is exact. Thus it is observed that the constraint is never violated within the interval $[-1\sigma, +1\sigma]$. Second, the possibility of obtaining a robust solution by introducing a backoff when it's impossible to reduce the sensitivity itself is demonstrated.

Uncertainty in the Side Shaft Play's Size

Robust result obtained for a standard deviation of 10% in the side shaft play's size p_{ss}. The linearized sensitivity has been reduced, and a suitable additional backoff has been introduced. The (underestimating) linearized sensitivity properly hits the constraint $\Delta\omega_{\max}$ at -1σ in Fig. 7.5(e). The bar diagram in Fig. 7.5(f) sorts the constraint violations of the 100 samples in 10% steps. The significant increase in feasibility, seen to the left of $\Delta\omega_{\max}$, is obvious.



(a) Optimal engine torque.

(b) Difference to nominal torque.

(c) Side shaft delta $\Delta\varphi_{\text{ss}}$.

(d) Rotation speed delta $\Delta\omega$.

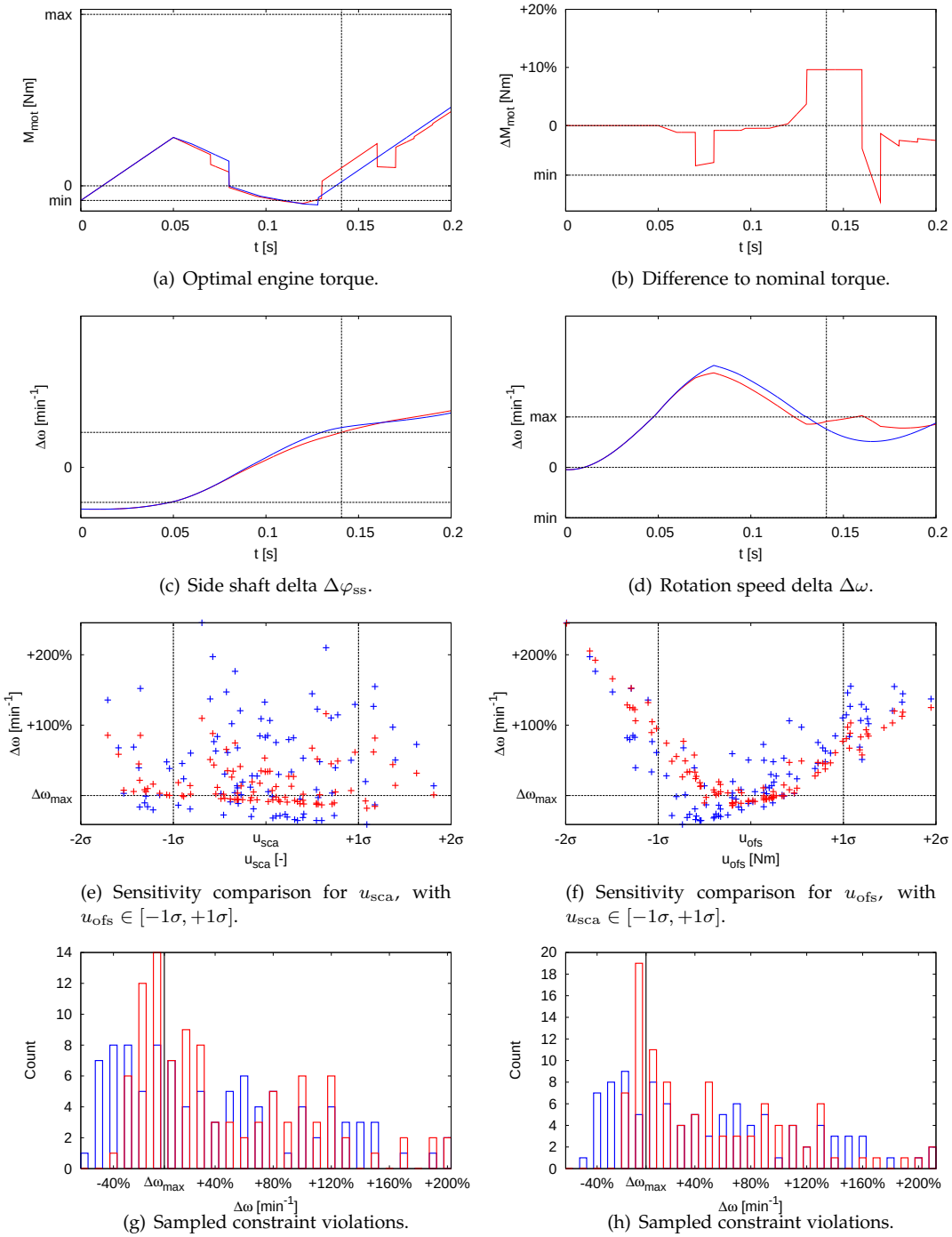
(e) Sensitivity comparison.

(f) Sampled constraint violations.

Fig. 7.5: Robust result for uncertainty in the side shaft play's size p_{ss}. (——) indicates the nominal solution of Fig. 7.1, while (——) indicates the robust solution.

Uncertainty in the Controlled Engine Torque

Fig. 7.6 shows the robust result for uncertainty in the controlled engine torque's scale and offset. In this two-parameter scenario the sensitivity with respect to both of the uncertain parameters is taken into account. Observe that the sensitivity with respect to u_{ofs} has been much reduced in a neighborhood of the mean. Figures 7.6(g) and 7.6(h) clearly show that the constraint violations have shifted to concentrate around the limit $\Delta\omega_{\max}$.



(a) Optimal engine torque.

(b) Difference to nominal torque.

(c) Side shaft delta $\Delta\varphi_{\mathrm{ss}}$.

(d) Rotation speed delta $\Delta\omega$.

(e) Sensitivity comparison for u_{sca}, with $u_{\mathrm{ofs}} \in [-1\sigma, +1\sigma]$.

(f) Sensitivity comparison for u_{ofs}, with $u_{\mathrm{sca}} \in [-1\sigma, +1\sigma]$.

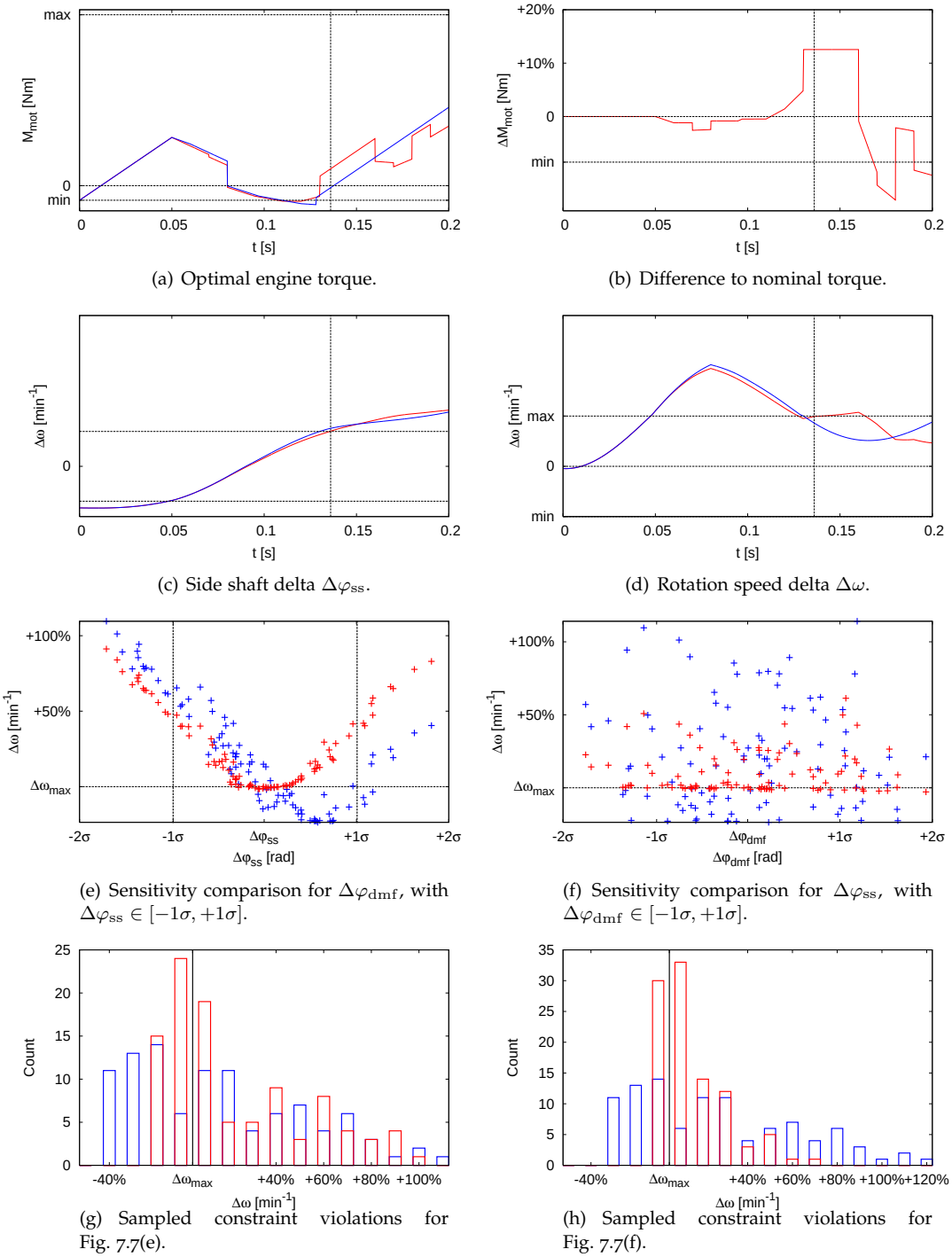
(g) Sampled constraint violations.

(h) Sampled constraint violations.

Fig. 7.6: Robust result for uncertainty in the controlled engine torque's scale and offset. (——) indicates the nominal solution of Fig. 7.1, while (——) indicates the robust solution.

Uncertainty in the Powertrain's Initial Torsion

Fig. 7.7 shows the robust result for uncertainty in the powertrain's initial torsion angles $\Delta\varphi_{\mathrm{dmf}}(t_0)$ and $\Delta\varphi_{\mathrm{ss}}(t_0)$. As seen from Fig. 7.7(d) the linearized sensitivity has been reduced to almost zero. Figures 7.7(e) and 7.7(f) confirm the much improved robustness.



(a) Optimal engine torque.

(b) Difference to nominal torque.

(c) Side shaft delta $\Delta\varphi_{\mathrm{ss}}$.

(d) Rotation speed delta $\Delta\omega$.

(e) Sensitivity comparison for $\Delta\varphi_{\mathrm{dmf}}$, with $\Delta\varphi_{\mathrm{ss}} \in [-1\sigma, +1\sigma]$.

(f) Sensitivity comparison for $\Delta\varphi_{\mathrm{ss}}$, with $\Delta\varphi_{\mathrm{dmf}} \in [-1\sigma, +1\sigma]$.

(g) Sampled constraint violations for Fig. 7.7(e).

(h) Sampled constraint violations for Fig. 7.7(f).

Fig. 7.7: Robust result for uncertainty in the powertrain's initial torsions. (——) indicates the nominal solution of Fig. 7.1, while (——) indicates the robust solution.

Chapter 8

Conclusions and Outlook

This final chapter is divided into three parts matching the major topics of this diploma thesis. We summarize the presented theory as well as the implemented algorithms, and collect the achievements concerning the guiding example of powertrain oscillations. Furthermore, we mention open topics and give an outlook on possible further research and work on the software implementations.

Parameter Estimation

We presented the theory of parameter estimation using least-squares fits against observed data, and motivated this approach by aiming at identifying maximum-likelihood estimators for normally distributed observation errors. We gave details on applying this parameter estimation methods to models described in terms of parameter-dependent ordinary differential equations (ODEs) with possibly unknown initial values. The constrained Gauß-Newton (CGN) method together with the direct multiple shooting method was chosen to solve the discrete nonlinear least-squares problems obtained from discretization of the initial-value problem (IVP) formulations. In addition, we showed how to obtain information about the uncertainty of the obtained solution by specifying confidence sets for the estimated parameter values.

Achievements The presented and numerical method for parameter estimation on IVPs was applied to the ODE model of a car's powertrain. It allowed us to precisely identify previously unknown parameters. The achieved match against data observed on the test tracks constitutes a significant improvement over previously available parameterizations. An analysis of the model's eigenvalues and their dependency on certain model parameters allowed us to derive a reduced non-stiff powertrain ODE model that was used for subsequent optimal control computations.

Implementation The described parameter estimation method was implemented on top of the optimal control software package *MUSCOD-II*. The new parameter estimation software *QuickFit* allowed for easy and user-friendly setup and treatment of the presented problem class. Internally, the resulting discretized nonlinear problems were solved with the constrained Gauß-Newton (CGN) version of the MSSQP algorithm of the *MUSCOD-II* optimal control software package.

Outlook The software is now being actively used in the *REI/EP* department of *Daimler-Chrysler AG* in *Stuttgart-Untertürkheim*, *Germany*, in order to obtain precise models of automatic gear-shift powertrains.

Implicitly Defined Discontinuities

We presented a class of explicit Runge-Kutta methods capable of detecting and handling implicit discontinuities of the ODE model equations (switch events). The solution of parameter estimation and optimal control problems using Newton-Lagrange techniques requires derivatives of the solution with respect to model parameters and initial values. Special updates to these sensitivities of the solution, which are required during switch events, were presented.

Achievements The efficient detection of switch events, which are marked as zero-crossings of the switching function, required a continuous representation of the discrete approximation to the IVP's solution. For that purpose, we discussed continuous extensions to classical discrete Runge-Kutta methods, which do not naturally have available such a continuous representation. The inverse quadratic interpolation algorithm by Brent and Decker was presented to reliably detect zero-crossings of the switching function.
In addition, the availability of a continuous representation made it possible to evaluate continuous least-squares objective functions.
The new integrator method enabled us to treat the important discontinuities of the guiding example in a precise manner. Furthermore, the support of implicit switches allowed for the formulation of objective and constraint functions evaluated at specific implicitly defined points in time, rather than at selected multiple shooting nodes only.

Implementation The presented continuous Runge-Kutta method was implemented as the *RK-FSWT* integrator method within *MUSCOD-II*. The implementation of previously unavailable continuous least-squares objectives benefited from the continuous extension. This feature was an essential requirement for the efficient solution of parameter estimation problems within the multiple shooting framework.

Outlook We had to put minor effort in the fact that the ODE model discontinuities need to be hidden from the sequential quadratic programming (SQP) level of the presented numerical method. It appeared extremely desirable to have at hand an SQP algorithm capable of handling implicitly defined discontinuities occurring when an integrator-handled switch crosses a multiple shooting node.
Since absent from the powertrain ODE model, the implemented method currently lacks treatment of inconsistent switching behaviour.

Robust Optimal Control

We have presented an approach for nonlinear programming under uncertainties that yields a problem structure that is computationally difficult to solve. A simplifying approach based on linearization has been used, and two different formulations preserving sparsity of the involved derivative matrices have been presented. After a transfer of the described methods to a class of optimal control problems, we have obtained a problem formulation that has been matched to the *MUSCOD-II* software architecture.

Achievements We have seen that in robust optimal control problems, sensitivities of the solution with respect to model parameters and initial values are themselves subject to optimization. Consequentially, second-order sensitivities are required within the Newton-Lagrange method solving the underlying discretized problem. We have developed an efficient approach for automatic generation of these second-order sensitivities, and have presented a way to implement proper updates of these sensitivities in implicit discontinuities, circumventing the need for extremely involved and error-prone explicit second-order updates.

The resulting numerical method has been applied to robustly solve a problem of optimal powertrain control. A mechanical limit imposed on the acceleration of the powertrain from coasting to traction mode has been evaluated, and the developed method has allowed us to obtain engine control schemes that implement this type of acceleration while exhibiting robustness against uncertainty in the model parameters, states, and the applied motor controls.

Implementation The developed numerical method has been implemented in the new *ROBUST* framework within the optimal control software package *MUSCOD-II*. The framework covers automatic generation of second-order sensitivities, as well as proper updates to all sensitivities during switch events. It has been designed to allow for very quick and convenient robustification of existing problem implementations.
To solve the discrete nonlinear programs resulting from our optimal control and robust optimal control problem formulations, we employed the MSSQP sequential quadratic programming algorithm of *MUSCOD-II* with BFGS updates to the Hessian.

Outlook The presented possibility for efficient second-order sensitivity generation could be further extended to include a modification of the condensing method (cf. Leineweber [38]) that exploits the sparsity structure of the sensitivity matrix.
We saw from the presented robust results that the linearization approach frequently underestimates the sensitivity of the highly nonlinear model. Approaches incorporating second-order approximations to the sensitivities are known, but lack possibilities for efficient implementation analogous to those we presented for the linearizing approach.
Finally, the framework does not currently support other uncertainty set shapes than those of ellipsoid shape described by scaled euclidean norms.

Appendix A

The Parameter Estimation Tool *QuickFit*

This chapter describes the parameter estimation tool *QuickFit* based on the optimal-control software package *MUSCOD-II*, which has been completed as a part of this thesis in order to simplify the process of estimating parameters in the presented powertrain models. The first version of *QuickFit* had been created and baptized by *Samuel Bandara* during a student research assistantship in the *Simulation & Optimization* work group.

An overview of the features, architecture, and implementation of the tool is given, aiming at creating a starting point for developers willing to extend *QuickFit*'s functionality. In addition, since *QuickFit*'s applicability is in no way restricted to the discussed powertrain models, this chapter also intends to serve as a user's guide for anyone interested in tackling his or her specific parameter estimation problem using that tool.

QuickFit is now being actively used in the *REI/EP* department of *DaimlerChrysler AG* in *Stuttgart-Untertürkheim*, *Germany*.

A.1 Input and Output Files

In this section we present in detail the files introduced by *QuickFit*, while we refer to Diehl et al. [12] for an extensive description of the *MUSCOD-II* files. Tab. A.1 holds a short listing of all *QuickFit* and *MUSCOD-II* input and output files.

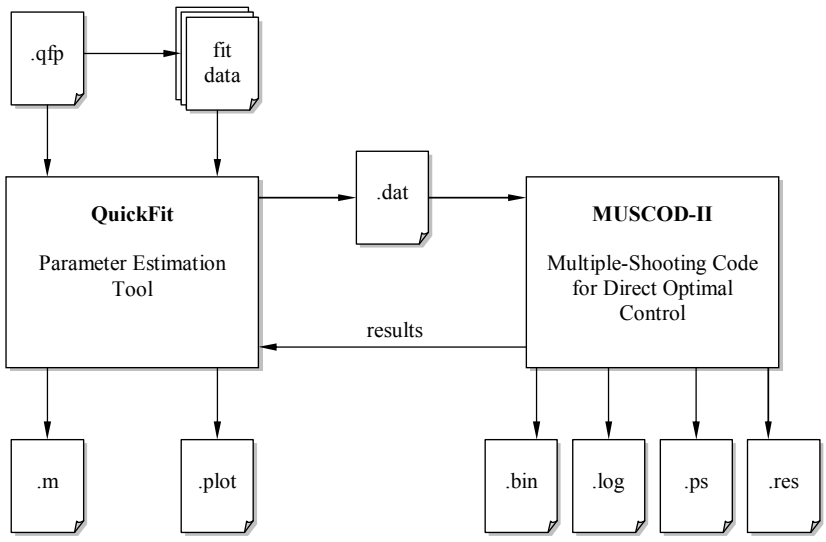


Fig. A.1: *QuickFit & MUSCOD-II* input and output files.

Path	File	Description
QuickFit		
`./`	`.qfp`	*QuickFit* input project file,
	`.m`	*QuickFit* output results file readable by *MATLAB*,
	`.plot`	*QuickFit* output plot file readable by *MATLAB* and *gnuplot*,
MUSCOD-II		
`./DAT/`	`.dat`	*MUSCOD-II* input file generated by *QuickFit*,
`./RES/`	`.log`	*MUSCOD-II* error log file,
	`.txt`	*MUSCOD-II* console output log file,
	`.bin`	*MUSCOD-II* binary file for restarts,
	`.ps`	PostScript pictures of the *MUSCOD-II* windows.

Tab. A.1: *QuickFit* and *MUSCOD-II* input/output file and directory structure.

A.2 The Project File

In this section we take a closer look at the *QuickFit* project file (QFP), which holds the major part of the parameter estimation problem's description. Syntax and semantics of the project file are described. Familiarity with the layout and functionality of *MUSCOD-II*'s DAT files described in Diehl et al. [12], however, is of advantage.
From the *QuickFit* project file (QFP) the DAT file containing the problem setup for *MUSCOD-II* is automatically generated. Whenever the QFP is modified, the DAT file is updated accordingly. You may force the update of the DAT file using the '`-F`' command line switch.

A.2.1 Lexical Conventions

Project file authors need to adhere to a few lexical conventions, all of which are listed in this section.

Comments start with an asterisk ("`*`") and run till the end of the same line. There is currently no provision for multi-line comments.

Blocks are introduced by a *signature* immediately followed by a colon ("`:`"). What follows is a set of numbered *objects* contained in this block.

Objects are contained in a *block* and occupy a single line of the project file each. The line consists of *columns* that list the object's *properties* and are separated by at least one whitespace character, e.g., a blank or a horizontal tabulator. The first column holds the objects *number* within the current block; objects must be given consecutive numbers starting from zero, but the list of objects may be unordered.

Properties of an object have a fixed *column* and *type* associated with them. Possible types include:

- integral constant ("int"),
- double-precision floating-point constant ("double"),
- boolean switch ("bool"),
- textual string ("string").

Integral and floating-point values must conform to the C/C++ syntax. Acceptable values for boolean switches are "y" and "n" to enable vs. disable the switch. Textual strings must not contain whitespace characters, quoting strings has no effect. All property values *must* be specified.

A.2.2 Project File Syntax

The following tables describe the contents of all project file blocks. Compare this to listing A.2 on page 112 showing a typical project file. A more formal specification of the project file is given in listing A.1 on page 109, which encodes the syntax using EBNF (cf. [31]).

```
eol          ::= '\n';
digit        ::= '0'..'9';
whitespace   ::= ' ' | '\t';
char         ::= any character - whitespace;

bool         ::= ('n' | 'N' | 'y' | 'Y'), *char;
integer      ::= ['-' | '+'], +digit;
double       ::= integer, [ '.', +digit ], [ ('e' | 'E'), integer ];
string       ::= +char

property     ::= bool | double | integer | string;
object       ::= integer, [ property, *( whitespace, property) ]

signature    ::= 'cntrls' | 'geometry' | 'outputs'
               | 'params' | 'states' | 'timing';
comment      ::= '*', any character - eol;
block        ::= signature, ':', [ comment ], eol,
                 *( [ object ], [ comment ], eol );

keyvaluepair ::= string, '=', *whitespace, string;

start        ::= +( block | keyvaluepair | comment );
```

Listing A.1: EBNF specification of the *QuickFit* project file syntax.

Controls

Unlike controls in *MUSCOD-II* that are part of the solution of an optimal control problem, and thus are subject to modifications by the solver, controls in a parameter estimation scenario are simply inputs of fixed observed data into the ODE model.

Signature	`cntrls:`	
Column	Type	Description
1	int	Object number.
2	string	Name of the control.
3	string	Unit of the control.
4	string	Relative path and name of the file containing the control trajectory.
5	bool	"y" to smoothen the trajectory by applying a Gaussian filter.
6	double	Sample time of the trajectory data.

Key/Value Pairs

In addition to objects contained in blocks, the project file recognizes key/value pairs, that is, identifiers being assigned an arbitrary value. Both the identifier name and the value must not contain whitespace characters.

Key	Type	Description
model	string	Name of the ODE model to be used.

Geometry

The observed data specified in the outputs block can be shifted in time (delay) and level (offset) using the geometry block. Both delay and offset may either be fixed to known initial values, or be included in the parameter estimation problem.

Signature	`outputs:`	
Column	Type	Description
1	int	Object number of the corresponding output object.
2	double	Lower bound for the delay.
3	double	Initial delay of the corresponding observed output data.
4	double	Upper bound for the delay.
5	bool	"y" to fix the delay to its initial value, "n" to estimate it.
6	double	Lower bound for the offset.
7	double	Initial offset of the corresponding observed output data.
8	double	Upper bound for the offset.
9	bool	"y" to fix the offset to its initial value, "n" to estimate it.

Objective

Although the block layout of the file allows for more than one objective to be defined, only the objective with an object number of zero will actually be used.

Signature	`objective:`	
Column	Type	Description
1	int	Object number.
2	string	Name of the objective function.
3	string	Unit of the objective function.
4	double	Lower bound of the range of expected objective values.
5	double	Upper bound of the range of expected objective values.
6	double	Scale of the expected objective values.
7	int	Number of least-squares residuals to be taken per multiple-shooting interval, using an equidistant time grid.

Outputs

The outputs block specifies the files containing observed data used to fit the model. Smoothing and weighting options may be selected.

Signature	`outputs:`	
Column	Type	Description
1	int	Object number.
2	string	Name of the model output.
3	string	Unit of the model output.
4	string	Path and name of the file containing the observed output data.
5	bool	"y" to smoothen the trajectory by applying a Gaussian filter.
6	double	Sample time of the trajectory data.
7	double	Confidence in the computed model output.

8	double	Factor for curvature-dependent weighting.
9	bool	"y" to include the output in the objective, "n" to exclude it.

Parameters

This block contains information about fixed and unknown parameters.

Signature	`params:`	
Column	Type	Description
1	int	Object number.
2	string	Name of the parameter.
3	string	Unit of the parameter.
4	double	Lower bound of the range of feasible parameter values.
5	double	Initial value of the parameter.
6	double	Upper bound of the range of feasible parameter values.
7	double	Confidence in the initial value, if it is not fixed.
8	double	Scale of the expected parameter values.
9	bool	"y" to fix the parameter's value, "n" to estimate it.

States

This block describes the initial values for the ODE model. Initial values may either be fixed or be included in the parameter estimation problem.

Signature	`states:`	
Column	Type	Description
1	int	Object number.
2	string	Name of the differential state.
3	string	Unit of the differential state.
4	double	Lower bound of the range of feasible state values.
5	double	Initial value of the differential state to its initial value.
6	double	Upper bound of the range of feasible state values.
7	double	Confidence in the initial value, if it is not fixed.
8	double	Scale of the expected differential state values.
9	bool	"y" fixes the state's value to its initial value.

Timing

Time slices are appended to each other, ordered by their object number. Thus the start time of any time slice matches the preceding time slice's end time, and is implicitly set to zero for slice number 0.

Signature	`timing:`	
Column	Type	Description
1	int	Object number.
2	double	End time of the time slice.
3	int	Number of multiple-shooting intervals on this time slice.

```
timing: 2
**   t_end   #shoot
 0   43      43

objective: 1
** name       unit    min    max    sca    nstop
 0 LSQ        -       0      1e2    1e2    200

states: 6
** name       unit    min    init   max    sigma  sca    fix
 0 x_w_dmf2   rad/s   176.5  177.1  177.5  177    177    n
...
 5 x_dphi_dmf rad     -0.15   -0.1    0.1    0.1    0.1 n

params: 20
** name       unit    min    init   max    sigma  sca    fix
 0 p_i_g      -        1.6     1.6    1.6    1.6    1.6 y
...
19 p_J_dmf1   kgm^2    0.01    0.1    1      0.1    0.1 n

controls: 1
** name       unit    file                 smooth   sampletime
 0 u_M_mot    Nm      M_mot_calc.txt       no       0.01

outputs: 8
** name       unit    file                 smooth   sampletime sigma  f  use
 0 z_M_cs     Nm      M_cs.txt             no       0.001       2     5  y
 1 z_M_ss     Nm      M_ss.txt             no       0.001       2     5  n
 2 z_a_car    m/s^2   ax_car.txt           no       0.001      10     0  n
 3 z_n_cs     rpm     n_cs.txt             no       0.001      10     0  y
 4 z_n_mot    rpm     n_mot_avg.txt        no       0.01       10     0  y
 5 z_n_ss     rpm     n_ss.txt             no       0.001      10     0  y
 6 z_v_car    km/h    v_wh_front_avg.txt   no       0.02        1     0  y
 7 z_v_wh     km/h    v_wh_rear_avg.txt    no       0.02        1     0  y

geometry: 8
** min_d init_d max_d fix_d     min_o init_o max_o fix_o
 0 -0.1  0      0     y          -30  0       30   y
 1 -0.1  0      0     y          -30  0       30   y
 2 -0.1  0      0     y           -1  0        1   y
 3 -0.1  0      0     y         -100  0      100   y
 4 -0.1  0      0     y         -200  0      200   y
 5 -0.1  0      0     y          -50  0       50   y
 6 -0.1  0      0     n            0  0        0   y
 7 -0.1  0      0     n            0  0        0   y
```

Listing A.2: A typical *QuickFit* project file.

A.3 Data Files

Plain text files are used to provide *QuickFit* with observed data for each of the model outputs. The files are expected in either *Windows* or *UNIX* format and must contain one double-precision floating-point value per line, conform to the C/C++ syntax.

Plain text data is parsed, preprocessed, and possible smoothed at the start of every run of the *QuickFit* software. With large parameter estimation scenarios consisting of many model outputs observed over a longer time or recorded with a very fine resolution, this process may easily take several minutes. For a significant speed-up, the processed data is cached on disk in a binary format that allows for very fast loading on the next startup. The cached files are consistently checked against the plain text data files and get automatically updated if the plain text data changes.

A.4 Output Files

In addition to the files created by *MUSCOD-II*, *QuickFit* creates a set of two output files of its own to document its findings for further use.

A.4.1 Results Output File

The estimated values and uncertainties of all free parameters and free output signal delays can be found in the results output file. In addition, the ℓ_2 residuals overall and per sample are given for each output. The data output file carries the extension ".m" indicating compatibility with *MATLAB*.

A.4.2 Trajectory Output File

The trajectory output file (".plot") holds the discretized IVP solution of the most recent iteration. Each column conforms to the layout shown in table A.10; each column entry is a double-precision floating-point value. The resolution is currently fixed to 1000 equidistant samples per multiple-shooting interval. The trajectory output file can be read by gnuplot [59] as well as by *MATLAB*.

First Column	# of Columns	Content
1	1	Time.
2	n_x	Differential states of the ODE model.
$2 + n_x$	n_o	Model outputs fitted against observed trajectories.
$2 + n_x + n_o$	n_o	Observed trajectories from the fit data input file.

Tab. A.10: Column layout of the *QuickFit* trajectory output file.

A.5 Command Line Arguments

Table A.11 holds a list of useful *QuickFit* and *MUSCOD-II* command line switches. We refer to Diehl et al. [12] for a full list of the *MUSCOD-II* command line switches with a detailed description.

Switch	Parameter	Default	Description
QuickFit			
`-F`	–	–	Forces an update of the DAT file.
`-R`	–	–	Prints the ℓ_1 and ℓ_2 residual norms per output.
MUSCOD-II			
`-a`	double	10^{-6}	Sets the KKT tolerance.
`-c, -w`	–	–	Continues from previous solution.
`-i`	long	100	Sets the maximum number of SQP iterations.
`-l`	double	0	Sets the Levenberg-Marquardt regularization factor.
`-t`	double	10^{-8}	Sets the integrator tolerance TOL.

Tab. A.11: *QuickFit* and *MUSCOD-II* command line switches. For a full list with detailed description, we refer to Diehl et al. [12].

A.6 Software Architecture

In this section we give a very brief overview of the software architecture of *QuickFit* by presenting a diagram of the various classes. We describe in detail how ODE models are coupled to *QuickFit* and how new model implementations can be added.

A.6.1 Class Diagram

As an extensive presentation and description of *QuickFit*'s internals is beyond the scope of this thesis, figure A.2 on page 116 shows a UML class diagram [26] of the important core classes of *QuickFit*. Arrows visualize dependencies and relationships according to the UML specification. The design is modular and allows for easy extension of the project file's structure.

A.6.2 Adding new ODE Model Implementations

Before *QuickFit* can treat a parameter estimation problem, it is necessary to include an implementation of the underlying ODE model in the application. Since this involves a small amount of modifications to the existing code base, the process is documented in this section to some detail. Notice that *QuickFit* is designed to hold a collection of different models all at once. Thus there is no need for models to be replaced; they can simply be added to the application.

Step 1: Preparing the Model Sources

Model sources together with a suitable makefile are to be placed in a separate directory per model located in `SRC/Models`. The ODE model needs to be implemented in a class named `Model` contained in a namespace of your choice. The class declaration is shown in listing A.3. The functions and parameters conform to the *MUSCOD-II* naming conventions described in Diehl et al. [12]. The function `getName` must return a unique model name. The *DaimlerChrysler* powertrain ODE models that may serve as a starting point for new model implementations can be found in the existing code base.

```
namespace SmallPowertrain    // Your model name here
{
    class Model : public qModelInterface
    {
    public:
        Model ();
        virtual ~Model ();
        virtual bool ffcn ( double t, double *xd, double *xa, double *p,
                            double *u, double *xd_dot, double *y );
        virtual bool gfcn ( double t, double *xd, double *xa, double *p,
                            double *u, double *xa_res );
        static const char *getName ();
    };
}
```

Listing A.3: Model class declaration expected by *QuickFit*.

Step 2: Coupling to the Existing Code Base

QuickFit dynamically selects one of the models included in its code base depending on the value of the `model` key set in the current project file, see Section A.2.2. In order to properly extend the list of recognised model names, you need to make a one-line modification to the source file `SRC/Common/qmodels.cpp` as indicated in Listing A.4.

```
qModelInterface* qModelFactory::createModel (
    const char* a_modelname
)
{
    assert ( a_modelname != 0 );

    qModelInterface *iface = 0;
    if ( ( iface = test_create< ATS_mit_Zusatzsensorik_deltaphi::Model >
         ( a_modelname ) ) != 0 ) return iface;
    if ( ( iface = test_create< ATS_mit_Zusatzsensorik_ohnegw AC::Model >
         ( a_modelname ) ) != 0 ) return iface;
    if ( ( iface = test_create< SmallPowertrain::Model >
         ( a_modelname ) ) != 0 ) return iface;

    // Add further models here in the same manner.

    throw qException ( format ( "Unknown␣model␣'%s'␣!", a_modelname ),
                       __FILE__, __LINE__, __FUNCTION__ );
}
```

Listing A.4: Update to `qmodels.cpp` required to add a new ODE model to *QuickFit*.

Step 3: Modifications to the Makefiles

Two straightforward extensions to the makefiles need to be carried out, as shown in listings A.5 and A.6.

```
all: smallpowertrain deltaphi smalldeltaphi
# Append your model's name to this list.

smallpowertrain:
    pushd .; cd SmallPowertrain; \
    ${MAKE} "CXXC=${CXXC}" "CFLAGS=${CFLAGS}" \
            "CXXCSFLAGS=${CXXCSFLAGS}" "RM=${RM}" all; \
    popd

# Copy the above instruction block and adjust its name and directory.
```

Listing A.5: Update to `SRC/Models/makefile` to add a new ODE model to *QuickFit*.

```
MODEL OBJECT = \
    ${MODELSPATH}/ATS_mit_Zusatzsensorik_deltaphi/ \
        ATS_mit_Zusatzsensorik_deltaphi_class.o \
    ${MODELSPATH}/ATS_mit_Zusatzsensorik_ohnegw AC/ \
        ATS_mit_Zusatzsensorik_ohnegw AC_class.o \
    ${MODELSPATH}/SmallPowertrain/SmallPowertrain_class.o

# Add you model's object file to this list.
```

Listing A.6: Update to `SRC/makefile` to add a new ODE model to *QuickFit*.

Step 4: Recompiling

Finally, executing `make` in the directory `SRC/` will update *QuickFit*'s executable `qfitgn`.

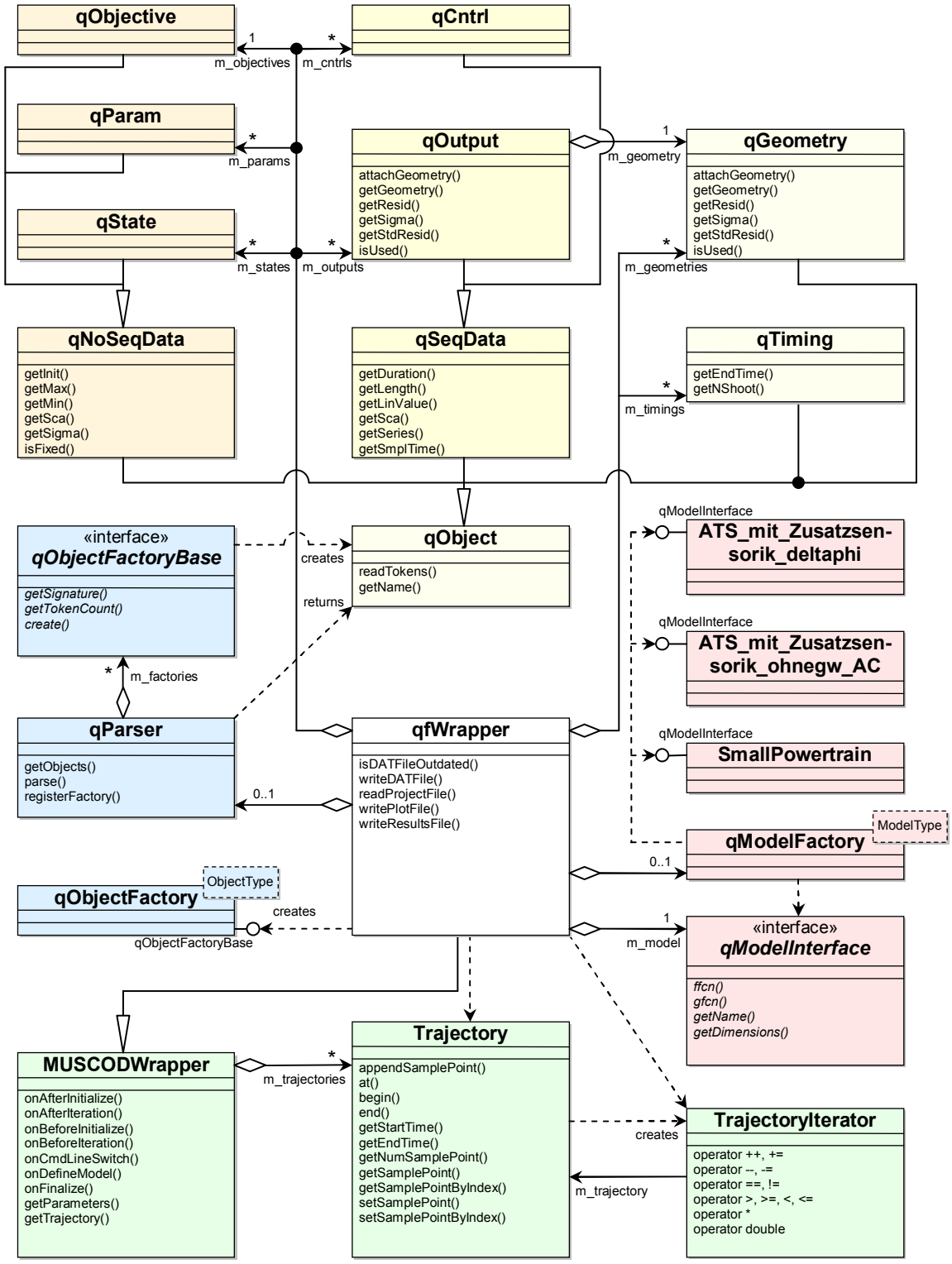


Fig. A.2: A UML class diagram of the *QuickFit* software architecture.

Appendix B

Implicit Switches in *MUSCOD-II* and *MATLAB/Simulink*

This appendix presents the new extensions to the *MUSCOD-II* user interface for the definition of ODE models with implicit switches, and extends the User's Manual [12] which should be consulted for further reference. Since the *DaimlerChrysler* powertrain model has been implemented in MATLAB/Simulink, we also implemented a method to realize the implicit switch detection feature within this modeling tool. The presented idea is of general applicability and may be used for any existing Simulink model that is to be connected to *MUSCOD-II*.

B.1 Extensions to the *MUSCOD-II* User Interface

B.1.1 Defining Switches in the Model

`def_msolver ( 1, def_RKF45SWT );`

As a first step, command *MUSCOD-II* to use the switch-detecting integrator `RKFSWT` presented in Chapter 4; here the $4^{\text{th}}/5^{\text{th}}$-order Fehlberg variant is used. You are of course free to use other solvers on additional model stages not containing implicit switches.

`def_swt ( imos, nswt, swtdtcfcn, swtexecfcn )`

This new function extends the *MUSCOD-II* user interface for the definition of models. Call it to define `nswt` switch event types on model stage `imos`. The switch-detecting integrator `RKFSWT` will call the user-supplied function `swtdtcfcn` during each integrator step to detect the presence of a switch event. Upon successful detection, the user-supplied function `swtexecfcn` is called to allow for discontinuous changes of the ODE system's differential states.

```
swtdtcfcn ( imos, t, xd, xa, u, p, nstep, iswt, nswsta, res, rwh,
            iwh, info )
```

This function is part of the model implementation to be supplied by the user. Its purpose is to evaluate the function (4.22)

$$\boldsymbol{\sigma}\big(t;\ \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t), \boldsymbol{p}, \operatorname{sgn} \boldsymbol{\sigma}(t)\big)$$

to determine the new switch signature in the time point t as presented in Chapter 4. Several arguments require explanation:

`nstep`

This argument is currently unused, and users are advised not to access it.

`iswt`

A pointer to a list of switch event indices whose respective scalar switch functions σ_i are to be evaluated. Your model's implementation of the function `swtdtcfcn` should be prepared to safely ignore indices out of range.

`nswsta`

The number of valid switch event indices to be found in the list pointed to by `iswt`. `nswsta` is guaranteed to be at least one, and not greater than the number of switch event types `nswt` specified by the respective call to `def_swt`.

`res`

A pointer to the output list of residuals σ_i to be filled by this function call.

The following code snippet shows how to iterate through the list of switch event indices i and fill in the residuals σ_i.

```
void
swtdtcfcn ( long *imos, double *t, double *xd, double *xa, double *u,
            double *p, long *nstep, long *iswt, long *nswsta, double *res,
            double *rwh, long *iwh, long *info )
{
    long ii;

    // iterate through the list of switch event indices
    for ( ii = 0; ii < *nswsta; ++ii )
    {
        // validate switch event index. Let NSWT be the number
        // of switch events on stage imos defined by def_swt.
        if ( ( iswt[ii] >= 0 ) && ( iswt[ii] <= NSWT ) )
        {
            // evaluate component iswt[ii] of sigma here.
            res[ii] = ...;
        }
    }
}
```

Listing B.1: Sample implementation of the switch detection function.

`swtexecfcn ( imos, t, xd, xa, u, p, iswt, rwh, iwh, info )`

This last model function is used to execute a jump $\boldsymbol{\Delta}$ (4.25) in the differential states $\boldsymbol{x}(t)$ by providing a chance to explicitly modify the state vector `xd`. It is perfectly valid to provide an empty function, though, if the implicit switch feature is being used to realize discontinuities $\boldsymbol{\delta}$ (4.28) in the ODE system's right-hand side only. The integer value `iswt` holds the index of the switch event to be executed, and ranges from `0` to `nswt-1`, see the description of `def_swt`.

B.1.2 Obtaining the Current Switch Signature

When evaluating the ODE model's right-hand side `ffcn`, the model implementer will want to react to the current switch signature as specified by the switch-detecting integrator `RKFSWT`. Ideally, one would have liked to extend the right-hand side function's list of arguments by a switch signature vector. In order to maintain compatibility with the vast collection of *MUSCOD-II* problems already implemented, however, we chose an approach that effectively hides the switching extension from model implementations that are unaware of its existence. As described in the *MUSCOD-II* User's Manual [12], the model's right-hand side is defined as

`ffcn ( t, xd, xa, u, p, rhs, rwh, iwh, info )`.

In detail, `info` is a pointer to an integer error code to be returned by `ffcn`. Without modifying the definition of `ffcn` agreed upon, the switch-detecting integrator hides a structure of the following layout (see the file `IND/RKFSWT/ind_rkfXXswt.h`)

```
typedef struct
{
    long  info;
    long *swt;
}
rkfXXswt_info_t;
```

Listing B.2: Extended structure containing the switch signature, pointed to by `info`.

behind that pointer in such a way that makes it safe to type-cast the `info` pointer in order to obtain the switch signature. Accessing it is as easy as shown in listing B.3.

```
#include "ind_rkfXXswt.h"

void
ffcn ( double *t, double *xd, double *xa, double *u, double *p,
       double *rhs, double *rwh, long *iwh, long *old_info )
{
    rkfXXswt_info_t* info = (rkfXXswt_info_t *) old_info;

    // info->swt[ ... ] contains the switch signature
}
```

Listing B.3: Obtaining the switch signature within the model's right-hand side `ffcn`.

B.2 Realisation of Switches in *MATLAB/Simulink*

The powertrain model presented in the introductory Chapter 1 was designed in *MATLAB/Simulink*; the C code generated from this model was coupled to the software package *MUSCOD-II* using a MATLAB interface specially tailored to fit this task. We present here a brief overview over the extensions to this interface we made in order to realize the proper handling of implicit model switches also in a *MATLAB/Simulink*-designed model.
Fig. B.1 visualizes the building blocks required to implement an implicit switch. In a pure *Simulink* environment the model developer would choose a `Switch` component that drives its output using one out of two input signals, depending on, e.g., the positiveness of a third input signal which we'll call the *selector*. The model itself would appropriately drive that selector input.

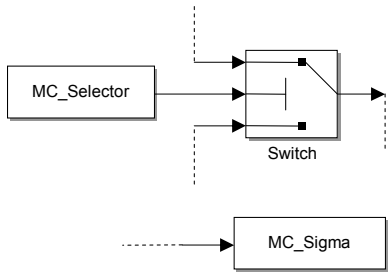


Fig. B.1: General realisation of a switch in *MATLAB/Simulink*.

As discussed in Chapter 4, however, the selection of the state of any switch within the model must be entirely up to the switch-handling integrator `RKFSWT`. We therefore introduce a new building block `MC_Selector` that routes the model's current switch signature vector (cf. Section B.1.2) and makes it available to the *Simulink* model as an input signal.
Furthermore, we're interested in automatic evaluation of the switch detection function (4.22) by the *Simulink* model so as to avoid having to add hand-written switch detection code to the automatically generated mode code. For this purpose, any `MC_Selector` input block is

complemented by an `MC_Sigma` output block. *MUSCOD-II* expects these outputs to provide the current value of σ_i for the respective switch. Evaluating the switch detection function $\boldsymbol{\sigma}$ then means nothing but (again) evaluating the model's right-hand side code exported from *Simulink*.
Finally, Fig. B.2 shows the implementation of the side shaft spring's play, combining two switches to accurately model the two non-differentiabilities.

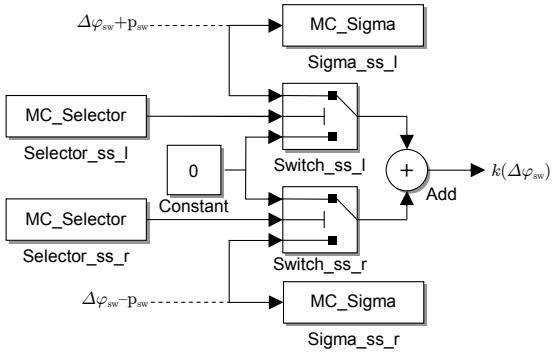


Fig. B.2: Implementation of the side shaft spring switches in *MATLAB/Simulink*.

Appendix C

A Framework for Robust Optimal Control in *MUSCOD-II*

In this appendix we present the robust optimization framework that extends the optimal control software package *MUSCOD-II* [38] by the robustification algorithm presented in this thesis. The appendix extends the *MUSCOD-II* User's Manual [12], which has been updated appropriately and should be consulted for further reading.

C.1 Robustifying a Problem Formulation

In this section we consider the robustification of an existing optimal control problem implementation. This robust optimization framework has been carefully designed to require as little changes to existing implementations as possible.

C.1.1 Source Files

The model source file requires the straightforward replacement of the *MUSCOD-II* model definition calls contained within `def_model()` by their robust counterparts presented in this section. We list all newly introduced functions, but assume familiarity with the *MUSCOD-II* model definition interface as described in Diehl et al. [12] and restrict ourselves to a discussion of the differences that users should be aware of.

`def_robust_mdims ( NMOS, NP, NRC, NRCE )`

Define global model dimensions. This function must be called prior to any call to `def_robust_mstage`.

```
def_robust_mstage ( IMOS, NXD, NXA, NU, mfcn, lfcn, jacmlo, jacmup,
                    astruc, afcn, ffcn, gfcn, rwh, iwh, ISLV )
```

Define a model stage and prepare it for robustification. This includes increasing `NXD` to contain sensitivity states, augmenting `ffcn` to automatically compute these directional sensitivities, and also the Mayer-term penalty if the objective is to be robustified.

`def_robust_mpc ( IMOS, N, NRD, NRDE, rdfcn, rcfcn, rdfcn_der )`

Defines `"NRD"`decoupled point-constraints on multiple shooting nodes of the model stage `IMOS`. The affected nodes are selected by the argument `N`, which may assume the values `s`, `i`, `e`, or `*` for the first, the interior, the last, or all nodes on the model stage. The first `NRDE` constraints are considered equality constraints, while the remaining ones are treated as inequality constraints. Only the latter may be selected for robustification using `rd_rob`. `rdfcn_der` may be specified to supply analytical derivatives of `rdfcn`, e.g., for coupling to automatic differentiation tools, cf. Griewank et al. [25]. The model stage `IMOS` must have been defined by a call to `def_robust_mstage`. In addition, this function defines the coupled point-constraints function `rcfcn`.

`def_robust_swt ( IMOS, NSWT, swtdtcfcn, swtexecfcn )`

Define `NSWT` implicit switches on the model stage with index `IMOS`, as described in appendix B. The crucial first-order sensitivity updates on implicit switch events are automatically performed, while the integrator `RKFSWT` deals with second-order sensitivity updates as described in Chapter 7. The model stage `IMOS` must have been defined by a call to `def_robust_mstage`.

C.1.2 Data File

The model data file has to be extended by several new keys which provide uncertainty and robustification information, and are described in this section.

`mfcn_rob(*), type long`

This key holds a single binary flag which should be set to a one in order to robustify the Mayer-term objective function of the selected stages. Set it to zero to use a conventional non-robust Mayer objective function.

`rd_rob(*,*), type LVec`

A list of `NRD-NRDE` binary flags to enable the robustification of the decoupled inequality points constraints. Each flag should be set to one in order to robustify the respective inequality constraint. Reset it to zero to disable the robustification of the constraint.

`p_rob, type LVec`

A list of `NP` binary flags to incorporate uncertainty of the respective model parameter. The model parameters must previously have been defined using the standard keys `p`, `p_min`, `p_max`, `p_sca`, and `p_fix`. Only fixed parameters may be attributed with an amount of uncertainty. You may account for uncertainty of free model parameters that are subject to optimization by splitting the parameter into a fixed uncertain mean value parameter and a free but certain offset to the mean that is subject to optimization. Variance and covariances are specified using the `covariance` key, see below.

`sd0_rob, type LVec`

A list of `NX` binary flags to incorporate uncertainty of the respective initial value. The differential states must previously have been defined using the standard keys `sd`, `sd_min`, `sd_max`, `sd_sca`, and `sd_fix`. Uncertainty in free initial values may be accounted for as described for uncertain free model parameters. Variance and covariances are specified using the covariance key, see below.

`covariance, type DMat`

For the initial values and parameters selected by the keys `sd0_rob` and `p_rob`, this key holds the variance and covariance information in the following format.

Suppose we defined n_{sd} uncertain initial values $\boldsymbol{x_0} \in \mathbb{R}^{n_{\mathrm{sd}}}$ and n_{p} uncertain parameters $\boldsymbol{p} \in \mathbb{R}^{n_{\mathrm{p}}}$. The variance-covariance matrix $\boldsymbol{\Sigma_x}$ must be of full rank, and is expected to obey the following layout:

$$\boldsymbol{\Sigma_x} = \begin{bmatrix} \boldsymbol{\Sigma_{\mathbf{sd0,sd0}}} & \boldsymbol{\Sigma_{\mathbf{sd0,p}}} \\ \boldsymbol{\Sigma_{\mathbf{sd0,p}}}^{\mathsf{T}} & \boldsymbol{\Sigma_{\mathbf{p,p}}} \end{bmatrix} \in \mathcal{M}(n_x + n_p, \mathbb{R}), \tag{C.1}$$

where the two square sub-matrices are

$$\boldsymbol{\Sigma_{\mathbf{sd0,sd0}}} = \Big[\mathrm{cov}(x_{0,i}, x_{0,j})\Big]_{ij} \in \mathrm{Sym}(n_{\mathrm{sd}}, \mathbb{R}), \tag{C.2}$$

$$\boldsymbol{\Sigma_{\mathbf{p,p}}} = \Big[\mathrm{cov}(p_i, p_j)\Big]_{ij} \in \mathrm{Sym}(n_{\mathrm{p}}, \mathbb{R}), \tag{C.3}$$

and for the matrix $\mathbf{\Sigma_{sd0,p}}$ we have without further restrictions

$$\mathbf{\Sigma_{sd0,p}} = \Big[\mathrm{cov}(x_{0,i}, p_j)\Big]_{ij} \in \mathcal{M}(n_{\mathrm{sd}} \times n_{\mathrm{p}}, \mathbb{R}). \tag{C.4}$$

The covariance matrix is thus symmetric, and only the lower left triangular sub-matrix needs to be specified in the DAT file.

`gamma, type double`

A common factor for scaling the covariance matrix. When working your way towards a robust solution of an optimal control problem, one quite often finds it impossible to immediately obtain robust solutions using uncertainty informations from the unscaled covariance matrix Σ. In this case a convenient method is to start with an optimal non-robust solution, and obtain increasingly robust solutions by way of successive and appropriately chosen homotopy steps.

C.1.3 Makefile

In order to use the robustification algorithm with an existing *MUSCOD-II* optimal control problem, edit the `makefile` and replace the file `MODEL/model.o` by its new counterpart `MODEL/model_rob.o`.
In addition, make sure that the file `ROBUST/robust_framework.o` is included in the list of object files to be linked[1]. For example, a robustified Gauß-Newton algorithm could look like this:

```
MC2OBJS = \
    ${MC2PATH}/MODEL/model_rob.o \
    ${MC2PATH}/PDAUX/pdaux.o \
    ${MC2PATH}/MSSQP/mssqp.o \
    ${MC2PATH}/EVAL/eval.o \
    ${MC2PATH}/SCALE/scale.o \
    ${MC2PATH}/PRSQP/prsqp_den.o \
    ${MC2PATH}/MSPLOT/msplot.o \
    ${MC2PATH}/HESS/hess_std.o \
    ${MC2PATH}/SOLVE/solve_slse.o \
    ${MC2PATH}/COND/cond_std.o \
    ${MC2PATH}/TCHK/tchk.o \
    ${MC2PATH}/UTIL/util.o \
    ${MC2PATH}/ROBUST/robust_framework.o
```

Listing C.1: Object files for a robustified Gauß-Newton algorithm.

C.2 Framework Components

The robust optimal control framework consists of the following object files.

Object File	Contents
`ROBUST/robust_module.o`	DAT file I/O routines to read information as described in Section C.1.2.
`ROBUST/robust_framework.o`	User interface routines, cf. Section C.1.1.
`MODEL/model_rob.o`	Replacement for `model.o` that connects to the ROBUST framework.

Tab. C.2: Object files of the robust optimal control framework.

[1] It should be noted that with the introduction of dynamic linking into *MUSCOD-II*, this procedure is likely to change in the near future.

List of Figures, Listings, and Tables

List of Figures

List of Listings

List of Tables

Bibliography

[1] Y. Bard. *Nonlinear Parameter Estimation*. Academic Press, New York, 1974.

[2] I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik*. PhD thesis, Ruprecht-Karls-Universität, Heidelberg, 1999.

[3] A. Ben-Tal and A. Nemirovskii. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS-SIAM Series on Optimization. MPS-SIAM, Philadelphia, 2001.

[4] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In Jäger Ebert, Deuflhard, editor, *Modelling of Chemical Reaction Systems*, volume 18 of *Springer Series Chemical Physics*, pages 102–125. Springer Verlag, Berlin Heidelberg New York, 1981.

[5] H.G. Bock. Recent advances in parameter identification techniques for ordinary differential equation. In *Progress in Scientific Computing*, volume 2. Birkhäuser, Boston, 1983.

[6] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. University of Bonn, Bonn, 1987.

[7] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9^{th} World Congress of the International Federation of Automatic Control, Budapest, Hungary*, pages 243–247. Pergamon Press, 1984.

[8] U. Brandt-Pollmann. *Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering*. PhD thesis, University of Heidelberg, Heidelberg, 2004.

[9] R.P. Brent. *Algorithms for minimization without derivatives*. Prentice Hall series in automatic computation. Prentice Hall, 1973.

[10] J.C. Butcher. Coefficients for the study of Runge-Kutta integration processes. *J. Australian Math. Soc.*, 3:185–201, 1963.

[11] M. Diehl. A heuristic for chance constrained optimization and application to a batch distillation process. *Optimization and Engineering (OPTE)*, 2006. (submitted).

[12] M. Diehl, D.B. Leineweber, and A.A.S. Schäfer. *MUSCOD-II User's Manual*. IWR Preprint 2001-25. Interdisciplinary Center for Scientific Computing, University of Heidelberg, 2001.

[13] M. Diehl, H.G. Bock, and E. Kostina. An approximation technique for robust nonlinear optimization. *Math. Prog. Series B*, 107(1–2):213–230, 2006.

[14] M. Diehl, J. Gerhard, W. Marquardt, and M. Mönnigmann. Numerical solution approaches for robust nonlinear optimal control problems. *Comp. Chem. Eng.*, 2006. (submitted).

[15] J.R. Dormand and P.J. Prince. A family of embedded Runge-Kutta formulae. *J. Comput. appl. Math.*, 6(1):19–26, 1980.

[16] J.R. Dormand and P.J. Prince. Runge-Kutta triples. *Comp. & Maths. with Appls.*, 12A(9): 1007–1017, 1986.

[17] T. Engelhard. Personal communication on June 14th, 2006.

[18] W.H. Enright. The relative efficiency of alternate defect control schemes for high-order continuous Runge-Kutta formulas. *SIAM Journal on Numerical Analysis*, 30(5):1419–1445, October 1993.

[19] W.H. Enright, K.R. Jackson, S.P. Nørsett, and P.G. Thomsen. Interpolants for Runge-Kutta formulas. *ACM Transactions on Mathematical Software (TOMS)*, 12:193–218, 1986.

[20] W.H. Enright, D.J. Higham, B. Owren, and P.W. Sharp. A survey of the explicit Runge-Kutta method. *Technical Report 291/94, Department of Computer Science, University of Toronto*, 1995.

[21] E. Fehlberg. Classical fifth-, sixth-, seventh-, and eighth order Runge-Kutta formulas with step size control. *Computing*, 4:93–106, 1969. Tech. Rep. 287, NASA, 1968.

[22] A.F. Filippov. Differential equations with discontinuous right-hand side. *Amer. Math. Soc. Trans.*, 42:199–231, 1961.

[23] P.E. Gill and W. Murray. Numerically stable methods for quadratic programming. *Math. Prog.*, 14:349–372, 1978.

[24] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Math. Prog.*, 27:1–33, 1983.

[25] A. Griewank, D. Juedes, H. Mitev, J. Utke, O. Vogel, and A. Walther. A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software (TOMS)*, 22:131–167, 1996. Algor. 755.

[26] Object Management Group. Unified modelling language superstructure specification. Website (valid Feb 13, 2006), 2005. `http://www.omg.org/docs/formal/05-07-04.pdf`.

[27] S.P. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Math. Progr.*, 11:263–282, 1976.

[28] S.P. Han. A globally convergent method for nonlinear programming. *J. Optimization Theory Applications (JOTA)*, 22:297–310, 1977.

[29] D. Higham. Highly Continuous Runge-Kutta Interpolants. *ACM Transactions on Mathematical Software*, 17:368–386, 1989.

[30] M. Horn. Fourth- and fifth-order, scaled Runge-Kutta algorithms for treating dense output. *SIAM J. Numer. Anal.*, 20:558–568, 1983.

[31] ISO/IEC. International Standard ISO/IEC 14977:1996(E). Website (valid Sep 1st, 2006), 1996. `http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf`.

[32] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, University of Chicago, 1939.

[33] K. Königsberger. *Analysis 1*. Springer-Verlag, Berlin Heidelberg New York, 5th edition, 2001.

[34] K. Königsberger. *Analysis 2*. Springer-Verlag, Berlin Heidelberg New York, 3th revised edition, 2000.

[35] S. Körkel, E. Kostina, H.G. Bock, and J.P. Schlöder. Numerical methods for optimal control problems in design of robust optimal experiments for nonlinear dynamic processes. *Optimization Methods and Software*, 19:327–338, 2004.

[36] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, Berkeley, 1941.

[37] W. Kutta. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeitschrift für Mathematik und Physik*, 46:435–453, 1901.

[38] D.B. Leineweber. The Theory of MUSCOD in a Nutshell – Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Diploma thesis, University of Heidelberg, Heidelberg, 1995.

[39] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math. 2*, pages 164–168, 1944.

[40] D.L. Ma and R.D. Braatz. Worst-case analysis of finite-time control policies. *IEEE Transactions on Control Systems Technology*, 9(5):766–774, 2001.

[41] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math. 11*, pages 431–441, 1963.

[42] K.D. Mombaur. *Stability optimization of open-loop controlled walking robots*. PhD thesis, University of Heidelberg, Heidelberg, 2002.

[43] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[44] B. Owren and M. Zennaro. Continuous explicit Runge-Kutta methods. In *Proceedings of the London 1989 Conference on Computational ODEs*, 1989.

[45] B. Owren and M. Zennaro. Order barriers for continuous explicit Runge-Kutta methods. *Math. Comp.*, 56:645–661, 1991.

[46] B. Owren and M. Zennaro. Derivation of efficient continuous explicit Runge-Kutta methods. *SIAM J. Sci. Stat. Comp.*, 13:1488–1501, 1992.

[47] H.B. Pacejka and E. Bakker. The magic formula tyre model. In *Proc. 1st International Colloquium on Tyre Models for Vehicle Dynamics Analysis*, Delft, The Netherlands, 1991.

[48] A. Potschka. Handling Path Constraints in a Direct Multiple Shooting Method for Optimal Control Problems. Diploma thesis, University of Heidelberg, 2006.

[49] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, Dundee, 1977*, Lecture Notes in Mathematics No. 630. Springer-Verlag, Berlin, 1978.

[50] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, MA, 2nd corrected edition, 2002.

[51] C. Runge. Über die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen*, 46:167–178, 1895.

[52] J.P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*, volume 187 of *Bonner Mathematische Schriften*. University of Bonn, Bonn, 1988.

[53] L.F. Shampine. Interpolation for Runge-Kutta methods. *SIAM J. Numer. Analysis*, 22: 1014–1027, 1985.

[54] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[55] C. Stelzer. Regelungstechnisches Modell eines KFZ-Antriebsstranges zur Simulation von Lastwechselschwingungen. Diploma thesis, Trier University of Applied Sciences, Trier, 2005.

[56] J. Stoer and R. Bulirsch. *Numerische Mathematik 2*. Springer-Verlag, Berlin Heidelberg New York, 4th edition, 2000.

[57] M. von Schwerin. *Numerische Methoden zur Schätzung von Reaktionsgeschwindigkeiten bei der katalytischen Methankonversion und Optimierung von Essigsäure- und Methanprozessen*. PhD thesis, University of Heidelberg, Heidelberg, 1997.

[58] D. Werner. *Funktionalanalysis*. Springer-Verlag, Berlin Heidelberg New York, 4th revised edition, 2002.

[59] T. Williams and C. Kelley et al. Gnuplot, a portable command-line driven interactive data and function plotting utility. Website (valid Feb 13, 2006), 2004. `http://www.gnuplot.info/`.

[60] R.B. Wilson. *A simplicial algorithm for concave programming*. PhD thesis, Graduate School of Business Administration, Harvard University, 1963.

Nomenclature

Decorations of Symbols

Symbol	Description
a, α	Scalar values in lowercase roman and greek letters.
$\boldsymbol{a}, \boldsymbol{\alpha}$	Vector values in lowercase roman and greek letters with boldface style.
$\boldsymbol{A}$	Matrix values in uppercase roman and greek letters with boldface style.
$\mathcal{A}$	Sets and subsets in uppercase roman letters with calligraphic style.
$\mathscr{A}$	Function space in uppercase roman letters with script style.
kg, s	Physical units in sans serif style.
$\overline{a}$	Supremum or upper bound of a. Also the mean value of a.
$\underline{a}$	Infimum or lower bound of a.
$\hat{a}$	Unknown real value belonging to an estimated value a.
$\tilde{a}$	A value that somehow deviates from a.
a^*	Optimum, final accepted iterate.
$a^{(k)}$	a belongs to step or iteration k.
$a^{[k]}$	a belongs to a one-step method of order k.
a_-	Left-hand side limit of a.
a_+	Right-hand side limit of a.
a_0	Initial value of a.
A^{-1}	Inverse of the regular matrix A.
A^+	Generalized inverse of the possibly singular matrix A.
A^T, a^T	Transpose of matrix A or vector a.
A_i, a_i	Row i of matrix A, the column index is missing. Element i of vector a.
$A_{\bullet j}$	Column j of matrix A, the row index is skipped.
A_{ij}	Element in row i and column j of matrix A.

Roman Symbols

Symbol	Domain	Description
$\boldsymbol{b}$	$\mathcal{T} \times \mathbb{R}^{k_i} \to \mathcal{U}$	Control discretization base functions.
c	$\mathbb{R}$	Spring coefficient (powertrain models).
$\boldsymbol{c}$	$\mathcal{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_c}$	Continuous inequality constraint function.
$\boldsymbol{c}$	$\mathbb{R}^s$	Vector of Runge-Kutta step coefficients (Chapter 4 only).
d	$\mathbb{R}$	Damping coefficient (powertrain models).
$\boldsymbol{e^i}$	$\mathbb{R}^n$	i-th unit vector.
f	$\mathcal{X} \to \mathbb{R}$	Objective function of an optimization problem.
$\boldsymbol{f}$	$\mathbb{R}^{n_r} \to \mathbb{R}$	Objective function of a least-squares problem.
$\boldsymbol{f}$	$\mathcal{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_x}$	Right-hand side of an ODE system.
$\boldsymbol{g}$	$\mathcal{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_g}$	Equality constraint function of an optimization problem.
$\boldsymbol{h}$	$\mathcal{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_h}$	Inequality constraint function of an optimization problem.
$\hat{\boldsymbol{h}}$	$\mathcal{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_{\hat{h}}}$	Active inequality constraint function.
h	$\mathbb{R}$	Integrator step size
$\imath$	$\mathbb{C}$	Imaginary unit, $\imath^2 := -1$.

Symbol	Domain	Description
i, j, l	$\mathbb{N}$	Subscript indices denoting components (A_{ij}, y_l).
k, n	$\mathbb{N}_0$	Sub- or superscript indices denoting a step or count.
m	$\mathbb{N}$	Number of multiple shooting subintervals.
n_c	$\mathbb{N}_0$	Number of continuous inequality constraints.
n_g	$\mathbb{N}_0$	Number of equality constraints.
n_h	$\mathbb{N}_0$	Number of inequality constraints.
$\tilde{n}_h$	$\mathbb{N}_0$	Number of active inequality constraints in NLPs.
n_o	$\mathbb{N}$	Number of observable model outputs.
n_r	$\mathbb{N}$	Number of residuals in parameter estimation problems.
$\boldsymbol{p}$	$\mathcal{P} \subset \mathbb{R}^{n_p}$	Vector of global model parameters.
$\boldsymbol{r}$	$\mathcal{X} \to \mathbb{R}^{n_r}$	Residual function in least-squares problems.
s	$\mathbb{N}$	Number of stages of a Runge-Kutta method.
$\boldsymbol{s}$	$\mathcal{M}(m \times n_x, \mathbb{R})$	Initial values of the multiple shooting subproblems.
t	$\mathcal{T} \subset \mathbb{R}$	Current time.
t_0, t_f	$\mathcal{T} \subset \mathbb{R}$	Start- and end-time.
t_s	$\mathcal{T} \subset \mathbb{R}$	Activation time of an implicit switch.
$\boldsymbol{u}$	$\mathcal{T} \to \mathcal{U}$	Control function in optimal control problems.
$\boldsymbol{x}$	$\mathcal{T} \to \mathcal{X}$	Differential states trajectory.
$\boldsymbol{y}$	$\mathcal{X} \to \mathbb{R}^{n_r}$	Valid model in parameter estimation problems.
$\boldsymbol{B}$	$\mathcal{M}(s-1, \mathbb{R})$	Part of the Butcher tableau of a Runge-Kutta method.
$\boldsymbol{E}$		Random var. from which observation errors $\boldsymbol{\varepsilon}$ are drawn.
F	$\mathbb{R}$	Force (powertrain models).
$\boldsymbol{G}_\mathrm{p}$	$\mathcal{T} \to \mathcal{M}(n_x \times n_p, \mathbb{R})$	Sensitivity of an IVP's solution $\boldsymbol{x}$ w.r.t. the parameters $\boldsymbol{p}$.
$\boldsymbol{G}_\mathrm{x}$	$\mathcal{T} \to \mathcal{M}(n_x, \mathbb{R})$	Sensitivity of an IVP's solution $\boldsymbol{x}$ w.r.t. the initial values $\boldsymbol{x_0}$.
$\boldsymbol{I}_n$	$\mathcal{M}(n, \mathbb{R})$	Unit matrix of size $n \times n$.
J	$\mathbb{R}$	Mass moment of inertia (powertrain models).
$\boldsymbol{J}^+$	$\mathbb{R}^{n_f+n_g} \to \mathbb{R}^{n_x}$	Generalized inverse solution operator.
$\boldsymbol{O}$		Zero matrix.
$\boldsymbol{T}$	$\mathcal{M}(n_x, \mathbb{R})$	Variance-covariance matrix of the estimate $\boldsymbol{x}$.
$\boldsymbol{U}_\mathrm{x}$	$\mathcal{M}(n_x, \mathbb{R})$	Update with respect to the states in an implicit switch.
$\boldsymbol{U}_\mathrm{p}$	$\mathcal{M}(n_x \times n_p, \mathbb{R})$	Update with respect to the parameters in an implicit switch.
W	$\mathscr{C}^1 \times \mathcal{U} \to \mathbb{R}$	Adverse player's worst-case response operator.
$\tilde{W}$	$\mathscr{C}^1 \times \mathcal{U} \to \mathbb{R}$	Convex approximation to the worst-case response operator.

Greek Symbols

Symbol	Domain	Description
$\boldsymbol{\alpha}$	$\mathbb{R}^s$	Vector of Runge-Kutta stage evaluation time points.
β	$\mathbb{R}$	Interpolation base polynomial.
$\boldsymbol{\delta}$	$\mathbb{R}^{n_x}$	Discontinuity of the right-hand side.
$\boldsymbol{\varepsilon}$	$\mathbb{R}^{n_r}$	Observation error in parameter estimation problems.
$\boldsymbol{\varepsilon}, \bar{\varepsilon}$	$\mathbb{R}^{n_x}, \mathbb{R}$	Global error of a one-step method.
$\boldsymbol{\eta}$	$\mathbb{R}^{n_x}$	Numerical approximate to the solution $\boldsymbol{y}$ of an IVP.
λ	$\mathbb{C}$	Eigenvalue.
$\boldsymbol{\lambda}$	$\mathbb{R}^{n_g}$	Lagrange multiplier of the equality constraints.
$\boldsymbol{\mu}$	$\mathbb{R}^{n_h}$	Lagrange multiplier of the inequality constraints.
ν	$\mathbb{N}$	Degrees of freedom of a statistical distribution.
ϕ	$\mathcal{T} \times \mathcal{X} \to \mathbb{R}$	Mayer-type objective function.
$\boldsymbol{\varrho}$	$\mathcal{T} \to \mathcal{X}$	Continuous interpolant of a Runge-Kutta method.
σ	$\mathbb{R}$	Standard deviation of a statistical distribution.
$\boldsymbol{\sigma}$	$\mathcal{T} \times \mathcal{X} \times \mathcal{P} \to \mathbb{R}$	Switching function.

Symbol	Domain	Description
$\boldsymbol{\sigma}, \overline{\sigma}$	$\mathbb{R}^{n_x}, \mathbb{R}$	Local error of a one-step method.
$\boldsymbol{\tau}, \overline{\tau}$	$\mathbb{R}^{n_x}, \mathbb{R}$	Consistency error of a one-step method.
$\boldsymbol{\xi}$	$\mathbb{R}^{n_r}$	Observed data, subject to observation errors $\boldsymbol{\varepsilon}$.
$\hat{\boldsymbol{\xi}}$	$\mathbb{R}^{n_r}$	Unobservable but error-free data underlying $\boldsymbol{\xi}$.
ω	$\mathbb{R}$	Angular velocity (powertrain models).
$\boldsymbol{\Delta}$	$\mathbb{R}^{n_x}$	Discontinuity in the states.
$\Delta\varphi$	$\mathbb{R}$	Angle difference (powertrain models).
Λ	$\mathcal{T} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$	Lagrange-type objective function.
$\boldsymbol{\Phi}$	$\mathcal{T} \times \mathcal{R} \times \mathcal{X} \times \mathscr{C}^1 \to \mathcal{X}$	Generating function of a one-step method.
$\boldsymbol{\Sigma}$	$\mathcal{M}(n_r, \mathbb{R})$	Variance-covariance matrix of the observations.
$\boldsymbol{\Xi}$		Random variable from which observations are drawn.

Calligraphic Symbols

Symbol	Domain	Description
$\mathcal{A}(\boldsymbol{x})$	$\subseteq \mathbb{N}$	Set of active inequality constraints in the point $\boldsymbol{x}$.
$\tilde{\mathcal{A}}(\boldsymbol{x})$	$\subseteq \mathbb{N}$	Set of strictly active inequality constraints in the point $\boldsymbol{x}$.
$\mathcal{F}_{\nu_1,\nu_2}$		Fisher's F-distribution with ν_1 and ν_2 degrees of freedom.
$\mathcal{M}(n, \mathbb{K})$		Set of $n \times n$ square matrices over the field $\mathbb{K}$.
$\mathcal{M}(m \times n, \mathbb{K})$		Set of $m \times n$ matrices over the field $\mathbb{K}$.
$\mathcal{N}_\nu$		Normal (Gaussian) distribution with ν degrees of freedom.
$\mathcal{O}$		Landau symbol.
$\mathcal{P}$	$\subseteq \mathbb{R}^{n_p}$	Global ODE model parameter space.
$\mathcal{T}$	$\subseteq \mathbb{R}$	Time horizon for initial-value problems.
$\mathcal{U}$	$\subseteq \mathbb{R}^{n_u}$	Control space for ODE optimal control problems.
$\mathcal{U}_\mathrm{p}$	$\subseteq \mathbb{R}^{n_p}$	Uncertainty set of the uncertain parameter $\boldsymbol{p}$.
$\mathcal{X}$	$\subseteq \mathbb{R}^{n_x}$	State space for IVPs and ODE optimal control problems.
$\mathcal{X}^2_\nu$		Chi-square distribution with ν degrees of freedom.

Blackboard, Fracture, and Script Symbols

Symbol	Description
$\mathscr{C}^n(\mathcal{D}, \mathcal{V})$	Set of n-times ($0 \leqslant n \leqslant \infty$) continuously differentiable functions over the domain $\mathcal{D}$ with values in the domain $\mathcal{V}$.
$\mathbb{C}$	Set of complex number.
$\mathbb{N}, \mathbb{N}_0$	Set of natural numbers (excluding, and including zero).
$\mathbb{P}(A\|B)$	Probability of the event A, given satisfied preconditions B.
$\mathbb{R}, \mathbb{R}^+, \mathbb{R}^+_0$	Sets of real (positive real, non-negative real) numbers.
$\mathbb{Z}$	Set of integer numbers.
$\Im$	Imaginary part of a complex numbers.
$\Re$	Real part of a complex number.

Other Symbols

Symbol	Description
$\|\|\cdot\|\|$, $\|\|\cdot\|\|_p$	Euclidean (ℓ_2-) norm, ℓ_p-norm ($1 \leqslant p \leqslant \infty$) of a vector.
$\ll, \gg$	Much less than, much greater than.
$\nabla_{\boldsymbol{x}} \boldsymbol{f}(\ldots)$	Vector (gradient) or matrix (Jacobian) of first partial derivatives of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$.
$\nabla^2_{\boldsymbol{x}} f(\ldots)$	Matrix (Hessian) of second partial derivatives of the scalar function f with respect to $\boldsymbol{x}$.
sgn	Signum function.
vec	Vectorization operator.

Abbreviations

Abbreviation	Description
BDF	Backward Differentiation Formula.
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BVP	Boundary Value Problem
CGN	Constrained Gauß-Newton
DAE	Differential-Algebraic Equation
DMF	Dual-Mass Flywheel
EBNF	Extended Backus-Naur Form
END	External Numerical Differentiation
IND	Internal Numerical Differentiation
IVP	Initial Value Problem
KKT	Karush-Kuhn-Tucker
LCQP	Linearly Constrained Quadratic Program
MUSCOD	Multiple-Shooting Code for Direct Optimal Control
NLP	Nonlinear Program
ODE	Ordinary Differential Equation
QFP	QuickFit Project File
QP	Quadratic Program
RKF	Runge-Kutta-Fehlberg
s.t.	subject to
SQP	Sequential Quadratic Programming
w.r.t.	with respect to

Index