Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

Minimum-Time Optimal Control for Automotive Vehicles
Nonconvex Optimisation and Space Splitting Convexification

Tadeas Sedlacek

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Klaus Diepold

Prüfende der Dissertation:

1. Priv.-Doz. Dr.-Ing. habil. Dirk Wollherr
2. Prof. Dr.techn. Manfred Plöchl,
   Technische Universität Wien

Die Dissertation wurde am 27.12.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 09.05.2022 angenommen.

Foreword

This dissertation summarises the research I conducted within the framework of a collaboration between the Chair of Automatic Control Engineering (LSR) at the Technical University of Munich (TUM) and the BMW M GmbH. The thesis contains results, figures, tables and entire paragraphs of our previous publications [183–186], to which I contributed substantially.

I am very grateful for the skills and experience I have obtained throughout my doctoral work, greatly profiting from the broad knowledge present at both institutions. It has been a thrilling journey studying interesting topics in close collaboration with talented colleagues, to whom I would like to express my gratitude.

First of all, I would like to thank my advisors Priv.-Doz. Dr.-Ing. habil. Dirk Wollherr from the LSR and Dr.-Ing. Dirk Odenthal from the BMW M GmbH for the opportunity to work as an associate within a team of excellent researchers. Besides improving my publication drafts through their great proofreading, their support and encouragement to challenge the state of the art have lead to the results presented in this thesis. Moreover, I would like to thank Prof. Dr.techn. Manfred Plöchl for his feedback on this thesis as well as Prof. Dr.-Ing. Klaus Diepold for chairing the examination.

Furthermore, I thank Dr.-Ing. Tobias Moosmayr, the head of my research group at the BMW M GmbH, who facilitated the funding of my research program and supported a free evolvement of the research. An especially warm thanks goes out to my colleagues at the BMW M GmbH, who enabled the growth and improvement of ideas via insightful discussions: M.Sc. Elias Reichensdörfer, M.Sc. Wolfgang Degel, M.Sc. Stefan Lupberger, Dipl.-Ing. Michael Sailer and Dipl.-Ing. Bernhard Seidl.

Finally, I deeply thank my family, especially my wife, for the endless support and encouragement throughout the entire process.

Munich, December 2021 Tadeas Sedlacek

Abstract

Optimal control is a powerful method that can be used to solve a variety of challenges for dynamic systems. However, the majority of real-life optimisation tasks requires the solution of a nonconvex optimisation problem. Two techniques for the solution of such optimal control problems are introduced in this thesis.
The first approach represents a general framework for the solution of optimal control problems with highly nonlinear models and difficult constraints. The procedure is applicable to a large class of nonlinear, continuous systems and aims at depicting the optimisation task as accurately as possible rather than achieving low computation times. The approach employs a direct method to generate a finite dimensional optimisation problem, which enables the use of nonlinear programming solvers. Hermite-Simpson collocation is used for increased sparsity enhancing the solvability of the optimisation problem. Several preliminaries are proposed that aim at improving the convergence behaviour and broadening the field of possible applications. The approach is used in this thesis to solve minimum lap time optimisation problems for automotive vehicles, which represent a challenging minimum-time optimal control application. The framework also enables a concurrent optimisation of selected model parameters. This provides optimal system parameters with corresponding optimal input trajectories and therefore the maximum possible benefit of individual vehicle setups. The objective comparison of various optimised vehicle concepts, with different powertrain designs or actuator setups, can accelerate the development process of vehicles. Furthermore, the optimal control trajectories can be used in a subsequent step to tune real-time capable controllers or deduce general control strategies.
While the first method pursues the goal of solving highly complicated optimisation problems accurately, the second method focusses on reducing computation time. The novel technique, which is called *space splitting convexification*, computes the solution of a special class of nonconvex optimisation problems by iteratively solving convex substitute problems. These convex quadratic programming problems are efficiently solvable by numerical solvers. The approach is capable of considering two types of nonconvexities: a broad class of two-dimensional, zonally convex sets and equality constraints with possibly multiple univariate nonlinearities. Although this method does not guarantee convergence to the global solution, the computed solution is a local optimum for a piecewise linear approximation of the original problem. The scalable approximation error is determined in advance, but is in trade-off with the size of the optimisation problem and thus the computation time. The outcome of the optimisation depends on the provided initial guess. However, the algorithm is capable of dealing with infeasible initial solutions due to a relaxation technique. Using an intermediate update routine, the relaxation is iteratively attenuated until convergence to the unrelaxed problem is achieved. The approach

is showcased using a hanging single-mass oscillator with nonlinear spring characteristic and semi-active damper as well as for a vehicular drag race application. Compared to a nonlinear programming solver, the novel iterative procedure greatly reduces computation time. Due to the low computation time and robust convergence, the presented method seems to be a promising approach for real-time optimal control applications.

Zusammenfassung

Mit Hilfe der Methode der Optimalsteuerung kann eine Vielzahl an Herausforderungen für dynamische Systeme gelöst werden. Allerdings erfordert der Großteil realer Optimierungsaufgaben die Lösung eines nicht konvexen Optimierungsproblems. In dieser Arbeit werden zwei Methoden vorgestellt, welche in der Lage sind solche Optimalsteuerungsaufgaben zu lösen.
Der erste Ansatz repräsentiert ein allgemeines Vorgehen zur Lösung von Optimalsteuerungsaufgaben für hochgradig nichtlineare Systeme mit komplizierten Nebenbedingungen. Die Methode kann für eine große Klasse nichtlinearer, kontinuierlicher Systeme verwendet werden. Dabei zielt der Ansatz darauf ab das Optimierungsproblem möglichst genau abzubilden, wobei eine kurze Rechendauer hier als zweitrangig erachtet wird. Die Verwendung einer direkten Methode ermöglicht den Einsatz numerischer Löser für nichtlineare Programme. Darüber hinaus liefert der Einsatz von Hermite-Simpson Kollokation ein dünnbesetztes Problem, was die Lösbarkeit erhöht. Weiterhin werden vorbereitende Maßnahmen präsentiert, welche das Konvergenzverhalten verbessern und das Anwendungsgebiet des Ansatzes vergrößern. Die Methode wird in dieser Arbeit zur Lösung zeitoptimaler Optimalsteuerugsprobleme für Fahrzeuge auf Rennstrecken verwendet. Diese Problemklasse stellt einen herausfordernden Sonderfall der zeitoptimalen Optimalsteuerung dar. Der Ansatz ermöglicht ebenfalls eine simultane Optimierung ausgewählter Modellparameter. Dieses Vorgehen stellt optimale Systemparameter mit zugehörigen Stellgrößenverläufen bereit und ermöglicht somit die Identifizierung der maximal möglichen Vorteile einzelner Aktuatorkonfigurationen. Der objektive Vergleich einzelner Fahrzeugkonzepte mit verschiedenen Antriebsstrangtopologien oder Aktuatorkonfigurationen kann den Entwicklungsprozess von Fahrzeugen beschleunigen. Des Weiteren können die ermittelten Stellgrößenverläufe im Nachgang genutzt werden um echtzeitfähige Regler auszulegen oder allgemeine Regelstrategien zu identifizieren.
Während die erste Methode darauf abzielt hochkomplexe Optimierungsaufgaben mit hoher Genauigkeit zu lösen, fokussiert sich der zweite Ansatz auf eine Reduktion der Rechendauer. Die neue Methode mit dem Namen *Raumteilungskonvexifizierung* ermittelt die Lösung einer speziellen Klasse nicht konvexer Optimierungsprobleme durch iteratives Lösen konvexer Ersatzprobleme. Diese konvexen quadratischen Programme können numerisch effizient gelöst werden. Der Ansatz ist in der Lage zwei Arten von Nicht-Konvexitäten zu berücksichtigen: eine große Klasse zwei-dimensionaler, zonal konvexer Sets und Gleichungsnebenbedingungen mit gegebenenfalls mehreren univariaten Nichtlinearitäten. Obwohl diese Methode die Konvergenz zum globalen Optimum nicht garantiert, stellt die Lösung ein lokales Optimum einer stückweise linearen Approximation des ursprünglichen Problems dar. Der skalierbare Approximationsfehler wird vorab festgelegt, steht jedoch im Zielkonflikt mit der Größe des Optimierungsproblems und somit

der Rechendauer. Das Ergebnis der Optimierung hängt von der zur Verfügung gestellten Initiallösung ab. Allerdings kann der Algorithmus dank einer Relaxationstechnik mit nicht-zulässigen Startlösungen umgehen. Eine zwischengeschaltete Aktualisierungsroutine reduziert kontinuierlich die Relaxation bis eine Konvergenz zum unrelaxierten Problem erfolgt. Die Methode wird anhand eines hängenden Einmassenschwingers mit nichtlinearer Feder und semi-aktivem Dämpfer sowie einer Drag Race Anwendung illustriert. Im Vergleich mit einem Löser für nichtlineare Programme wird die Rechendauer durch den neuen iterativen Ansatz deutlich reduziert. Auf Grund der geringen Rechenzeit und der robusten Konvergenz, scheint die vorgestellte Methode einen vielversprechenden Ansatz für die Lösung von Optimalsteuerungsproblemen in Echtzeit darzustellen.

Contents

List of Figures

List of Tables

Acronyms

ADMA	Automotive Dynamic Motion Analyzer.
ADMM	alternating directions method of multipliers.
ADP	approximate dynamic programming.
AVC	absolute value constraint.
BEV	battery electric vehicle.
BnB	branch-and-bound.
BVP	boundary value problem.
CCP	convex-concave procedure.
CEV	combustion engine vehicle.
COG	centre of gravity.
DDP	differential dynamic programming.
DM	direct method.
DP	dynamic programming.
EOL	electrical overloading.
EST	expansive spaces tree.
GO	global optimisation.
GP	geometric programming.
HEV	hybrid electric vehicle.
ILP	integer linear programming.
IM	indirect method.
IP	integer programming.
IPM	interior-point method.
KKT	Karush-Kuhn-Tucker.
LC	lossless convexification.
LICQ	linear independence constraint qualification.
LnP	linearise-and-project.
LP	linear programming.
LQR	linear quadratic regulator.
LTV	linear time-varying.
MINLP	mixed-integer nonlinear programming.
MIP	mixed-integer programming.
MPC	model predictive control.
MS	multiple shooting.
NLP	nonlinear programming.
OCP	optimal control problem.
ODE	ordinary differential equation.
OP	optimisation problem.
PID	proportional-integral-derivative.
QCQP	quadratically constrained quadratic programming.
QP	quadratic programming.
QSS	quasi steady-state.
RRT	rapidly-explorig random tree.
SDP	semi-definite programming.
SMO	single-mass oscillator.
SOC	state of charge.
SOCP	second-order cone programming.
SOP	static optimisation problem.
SQP	sequential quadratic programming.
SS	single shooting.
SSC	space splitting convexification.
SST	stable sparse tree.
TS	Takagi-Sugeno.
WIDV	wheel-independent drive vehicle.
ZCS	zonally convex set.

Notation and Symbols

Scalars, Vectors, and Matrices

Scalars are denoted by upper and lower case letters in italic type. Vectors are denoted by lower case letters in bold type with x_i representing the i^{th} element of the vector $\mathbf{x}$. Matrices are denoted by upper case letters in bold type.

Mathematical Notation

$\dot{\mathbf{x}}$	time derivative of vector-valued variable $\mathbf{x}$
$\frac{\partial f}{\partial x}$	derivative of function f with respect to scalar-valued variable x
$\nabla_{\mathbf{x}} f(\mathbf{x})$	gradient of function $f(\mathbf{x})$ with respect to vector-valued variable $\mathbf{x}$
$\nabla^2_{\mathbf{xx}} f(\mathbf{x})$	Hessian of function $f(\mathbf{x})$ with respect to vector-valued variable $\mathbf{x}$
$\mathbb{R}$	set of real scalars
$\mathbb{R}^n$	set of n-dimensional, real vectors
$\mathbb{R}^{n_w}_{>0}$	set of n-dimensional, real, positive vectors
$\emptyset$	empty set
$\|\cdot\|_2$	Euclidean norm of a vector
$\mathcal{O}(\cdot)$	Landau-notation for computational complexity

Optimal Control Problem

$\overline{(\cdot)}, \underline{(\cdot)}$	upper and lower bound of respective variable $(\cdot)$
$\hat{x}, \hat{u}, \hat{p}$	scaled state, input and parameter vector
$\mathcal{A}$	index set of all active constraints
$\mathbf{c}$	left-hand sides of equality constraints
$\mathbf{c}_{\text{coll}}$	left-hand sides of collocation equality constraints
$\mathcal{D}$	domain of function
Δ_i	width of i^{th} collocation segment
$\mathcal{E}$	set of equality constraints indices
$\mathbf{f}$	right-hand side of differential equations system
ϕ	intermediate objective term
$\mathbf{\Phi}$	boundary conditions
$\mathbf{h}$	left-hand sides of inequality constraints
$\mathcal{H}$	Hamiltonian function
$\tilde{\mathcal{H}}$	extended Hamiltonian function
$\mathcal{I}$	set of inequality constraints indices
$\mathcal{I}_0$	set of active inequality constraints indices

$\mathcal{I}_-$	set of inactive inequality constraints indices
J	objective function
$\mathcal{L}$	Lagrangian function
$\boldsymbol{\lambda}$	Langrange multiplier for equality constraints
$\boldsymbol{\mu}$	Langrange multiplier for inequality constraints
n_c	number of equality constraints
n_{coll}	number of collocation points
n_h	number of inequality constraints
n_p	number of adjustable, time-invariant model parameters
n_{seg}	number of collocation segments
n_u	number of input decision variables
n_w	number of original decision variables
n_x	number of state decision variables
$\mathbf{p}$	vector of adjustable, time-invariant model parameters
$\mathcal{P}$	bounding-box set for adjustable, time-invariant model parameters
t	time
ϑ	terminal objective term
$\mathbf{u}$	system input vector
$\mathcal{U}$	bounding-box set for system inputs
$\mathbf{w}$	decision variable
$\mathbf{w}^\star$	(locally) optimal solution for decision variable $\mathbf{w}$
Ω_f	feasible set
$\mathbf{x}$	system state vector
$\mathcal{X}$	bounding-box set for system states

Curvilinear Coordinates

$\mathcal{B}$	path boundary
$d_{\mathcal{B}}$	lateral distance of path boundary to tangent of reference curve
d_{hw}	half-width of path boundary
$d_{\mathcal{R}}$	lateral distance of COG to tangent of reference curve
$\kappa_{\mathcal{R}}$	curvature of reference curve
$\mathcal{R}$	reference curve
$s_{\mathcal{R}}$	arc-length of reference curve
Θ	error angle between reference curve tangent and COG-orientation
$\Theta_{\mathcal{R}}$	tangent angle of reference curve
v_{tot}	total velocity at COG of moving body
v_x	longitudinal velocity at COG of moving body
v_y	lateral velocity at COG of moving body
ψ	yaw angle

Two-Track Vehicle Model for NLP-Algorithm

A_{air}	cross-section area of vehicle
a_x, a_y	longitudinal/lateral acceleration at COG
b_f, b_r	track width of front/rear axle
b_v	vehicle chassis width
$c_{\text{air},x}, c_{\text{air},y}$	longitudinal/lateral drag coefficient
$c_{\text{air},z,k}$	aerodynamic lift coefficient for corresponding wheel
C_d	capacitance of battery pack
δ_f	front steering angle
ΔP_{E_b}	normalised power flow into overload tank of battery pack
$\Delta P_{E_{m,k}}$	normalised power flow into overload tank of corresponding electric machine
E_b	overload tank of battery pack
$E_{m,k}$	overload tank of corresponding electric machine
$f_{\text{roll},0}$	rolling resistance coefficient
$F_{x,k}, F_{y,k}$	longitudinal/lateral tire force for corresponding wheel in the body coordinate frame
$\mathcal{F}_{x,k}, \mathcal{F}_{y,k}$	longitudinal/lateral tire force for corresponding wheel in the wheel coordinate frame
g	gravitational acceleration
h_v	COG height
$\eta_{g,k}$	efficiency factor of transmission for corresponding wheel-unit
I_b	battery pack current
$i_{g,k}$	gear ratio for corresponding wheel-unit
$J_{w,\text{in},k}$	wheel-unit inertia on input-side for corresponding wheel
$J_{w,k}$	total wheel-unit inertia for corresponding wheel
$J_{w,\text{out},k}$	wheel-unit inertia on output-side for corresponding wheel
J_{zz}	vehicle inertia around vertical axis
k_{mot}	performance shift variable
l_f, l_r	distance from COG to front/rear axle
$\lambda_{x,k}, \lambda_{y,k}$	longitudinal/lateral tire slip for corresponding wheel
$\lambda^\star_{x,k}, \lambda^\star_{y,k}$	optimal longitudinal/lateral tire slip for corresponding wheel
m	total vehicle mass
μ	road friction coefficient
n_s	number of auxiliary decision variables
P_{l_b}	power losses of battery pack
$P_{l_i,k}$	power losses of corresponding inverter
$P_{l_m,k}$	power losses of corresponding electric machine
Q_b	capacity of battery pack
R_d	internal resistance of battery pack for short-term dynamic behaviour depiction
R_i	internal resistance of battery pack
r_k	dynamic rolling radius of corresponding wheel
$T_{br,k}$	friction brake torque at corresponding wheel
$T_{m,k}$	torque at corresponding electric machine
$T_{w,k}$	total torque at corresponding wheel
τ_{acc}	time constants for acceleration low-pass filter

$\tau_{\mathrm{bst},b}$	boosting time of battery pack
$\tau_{\mathrm{bst},m}$	boosting time of electric machines
V_0	terminal voltage of battery pack
V_d	capacitor voltage of battery pack for short-term dynamic behaviour depiction
V_{oc}	open circuit voltage of battery pack
v_x	longitudinal velocity at COG
v_y	lateral velocity at COG
$v_{x,k}, v_{y,k}$	velocities of wheel centre points in the body coordinate frame
$\mathrm{v}_{x,k}, \mathrm{v}_{y,k}$	velocities of wheel centre points in the corresponding wheel coordinate frame
ω_k	rotational speed of corresponding wheel
$\omega_{m,k}$	rotational speed of corresponding electric machine
ξ_{roll}	roll moment distribution factor
ψ	yaw angle
$\dot{\psi}$	yaw rate
ζ	battery pack state of charge

Space Splitting Convexification

$\hat{(\cdot)}$	transformed constraints depending on $\mathbf{z}$
$\breve{(\cdot)}$	projected initial solution of $(\cdot)$
$(\cdot)^\star$	previous optimal solution of $(\cdot)$
$(\cdot)_{\mathrm{tr}}$	transition point for space bisection
$(\cdot)_{n/\mathrm{neg}}$	negative region of bisected space
$(\cdot)_{p/\mathrm{pos}}$	positive region of bisected space
$\alpha(\cdot)$	smoothable absolute value function
$\hat{\mathbf{A}}, \hat{\mathbf{b}}$	matrix and vector of affine transformation function
$\mathbf{A}, \mathbf{B}, \mathbf{k}$	matrices and vector for system behaviour description
$\mathbf{A}_{\mathrm{zcs}}, \mathbf{b}_{\mathrm{zcs}}$	matrix and vector describing affine ZCSs
$\tilde{\mathbf{c}}$	left-hand sides of continuous-time equality constraints
$\mathbf{c}_{\mathrm{tot}}$	left-hand sides of aggregated equality constraints
ε_g	error tolerance for linearised AVCs
ε_s	smoothing parameter for absolute value function
$\mathbf{E}_{\Sigma/\Delta/z}$	matrices for selecting specific decision variables
$\mathbf{f}_\Phi$	affine approximation of right-hand side of model equations
$\mathbf{f}_{\mathrm{pwl}}$	composition of univariate nonlinearities approximable by piecewise linear curves
$\mathbf{\Phi}_{x/u/\mathrm{pwl}}$	mapping functions affine in the decision variables
$\mathbf{g}_{\mathrm{abs}}$	left-hand sides of AVCs for space splitting
$\mathbf{g}_{\mathrm{aff}}$	left-hand sides of affine constraints for space splitting
$\tilde{\mathbf{h}}$	left-hand sides of continuous-time inequality constraints
$\mathbf{h}_{\mathrm{abs}}$	left-hand sides of relaxed (inequality) AVCs for space splitting
$\mathbf{h}_{\mathrm{tot}}$	left-hand sides of aggregated inequality constraints
$\mathbf{h}_{\mathrm{zcs}}$	left-hand sides of ZCS inequality constraints
$\mathcal{I}_{\mathrm{coll}}$	set of collocation segment indices
J	objective function

$\tilde{J}$	continuous-time objective functional
k_{init}	perturbation parameter for initialisation error
$\mathcal{K}$	set of collocation point indices
$\mathbf{l}_{\text{abs}}$	left-hand sides of linearised AVCs for space splitting
q	number of superordinate iterations in SSC algorithm
$q_{\max}$	maximum number of superordinate iterations in SSC algorithm
s_w	slack variable for AVC
$\mathcal{S}_{(\cdot)}$	set described by inequality constraints
$\sigma_{(\cdot)}$	sign of corresponding AVC
$\mathbf{\Sigma}$	matrix containing all linearisation points of AVCs
$t_{\text{opt},\Sigma}$	cumulative time spent in solver
t_Σ	total computation time
τ	penalty parameter for additional objective terms
$\mathbf{U}_0^\star$	initial solution for input decision variables
$\mathbf{w}$	vector of original decision variables
$\mathbf{w}_{\text{aux}}$	vector of auxiliary decision variables
$w_{\text{low/up}}$	decision variable depicting lower/upper region of bisected space for w
$\mathbf{X}_0^\star$	initial solution for state decision variables
$\mathbf{z}$	vector of accumulated decision variables
$\tilde{\mathbf{z}}$	argument vector for absolute value function
$\tilde{\mathbf{z}}^\star$	linearisation points for AVCs

Single-Mass Oscillator Model for SSC-Algorithm

$c_{\text{low/mid/up}}$	spring stiffness of corresponding linear segment
d	damper coefficient
$\overline{e}_{F_d}$	mean absolute deviance in damper force trajectories between NLP and SSC solution
e_{ss}	steady-state position deviation
$\overline{e}_{\text{ss}}$	mean absolute deviance between position trajectory and steady-state position
F_c	spring force
F_d	damper force
g	gravitational acceleration
J_{ss}	objective function for steady-state position deviation
m	mass
v	deflection velocity
x	deflection position
$\mathbf{x}_{\text{iv}}$	initial value of system state
x_{ss}	steady-state position
$x_{\text{low}/\overline{\text{low}}}$	decision variable for lower/upper region of position space bisection
$x_{\text{mid/up}}$	decision variable for lower/upper region of bisected position space $x_{\overline{\text{low}}}$

Single-Track Vehicle Model for SSC-Algorithm

B_x, C_x, D_x	friction coefficients for longitudinal tire force
$c_{\text{air,low}}, c_{\text{air,up}}$	slope of lower/upper region of normalised aerodynamic force curve
$c_{f,\text{low}}, c_{f,\text{up}}$	slope of lower/upper region of inverse tire force shape curve
$\overline{f}$	maximum value for tire force shape curve
f_{tr}	transition point for tire force shape curve
$f_{x,\text{air}}$	normalised curve for aerodynamic drag force
$f_{x,r}$	shape curve for longitudinal tire force
$F_{x,f}, F_{x,r}$	longitudinal tire force of front/rear wheel
J_f, J_r	inertia of front/rear wheel
$\lambda_{x,r}$	rear wheel tire slip
$\overline{P}$	maximum power
r_f, r_r	dynamic rolling radius at front/rear wheel
$\overline{T}$	maximum torque
$\overline{T}_{\text{lin}}$	linearised speed-dependent maximum torque
$\overline{T}_{\text{nl}}$	nonlinear, speed-dependent maximum torque
T_f, T_r	torque at front/rear wheel
$\overline{v}$	maximum velocity
v_{tr}	transition point for velocity
ω_f, ω_r	rotational speed of front/rear wheel

1 Introduction

By nature, humankind strives to make the best decision when confronted with problems. People try to find the path of shortest time considering the current traffic situation and infrastructure when aiming to reach a destination. In order to generate the best possible business results, decisions are made under consideration of all relevant business goals and business rules. In engineering, the best product design is often a trade-off between individual subgoals like engine power, engine weight and installation space. All these problems can be expressed in form of an optimisation problem (OP) that can be solved using mathematical programming. Mathematical programming or mathematical optimisation is a powerful tool that enables finding the best element from a set of alternatives considering an evaluation criterion and given restrictions. Growing economic pressure and high performance requirements often raise the interest of companies to optimise technical systems in regards to design and control. Many of these systems can be described using models: engines, electric motors, chemical reactors, industrial robots, trains, airplanes, vehicles and more. Optimal control deals with solving constrained OPs that consider such dynamic system models. The goal is defined by minimising a functional under consideration of given constraints and equations describing the temporal development of the dynamic system. Frequent goals of optimal control are minimisation of time consumption, energy consumption, tracking errors or operating effort. Thus, optimal control can be used to identify optimal control trajectories, optimal strategies or an optimal design. Depending on the contemplated system model and optimisation task, the resulting OPs get very complicated and can often only be solved numerically. The progress in computing power continuously lowers computation times, which steadily increases the appeal of numerical optimisation.

This thesis is comprised of two parts. The first part deals with optimal control methods for possibly complicated systems aiming to solve nonconvex OPs. For this approach, an accurate solution is valued more than short computation times. The thesis focuses on minimum-time optimal control, which aims at computing time-optimal solutions. Although applicable to many technical systems, this thesis commits to automotive applications aiming at minimising lap times on racetracks. Due to the highly nonlinear model equations and complicated path constraints, minimum lap time optimal control for vehicular applications represents a challenging problem. Since few other real-life problems are comparatively difficult, the approach is capable of solving a large class of optimal control problems (OCPs) for many different applications. This first approach is capable of concurrently optimising passive vehicle parameters and control inputs enabling the performance improvement of vehicles. The following idea motivates such simultaneous optimisations: Different actuators with adequate control systems can enable a different design of the passive system, which can improve agility. This is a well-known design

concept in aviation, which is utilised to enhance manoeuvrability of combat aircrafts: The centre of gravity (COG) is placed in a way which renders the system statically unstable, whereas the control systems assure that the airplane remains controllable by the pilot [72, p.468]. Thus, the concurrent optimisation enables identifying optimal passive vehicle setups with corresponding optimal control strategies. The objective comparison of various optimised vehicle concepts, with different powertrain designs or actuator setups, can accelerate the development process of vehicles. Furthermore, the optimal control trajectories can be used to tune real-time capable controllers or deduce general control strategies in a subsequent step.
While the first part of this thesis primarily targets the accurate computation of the optimal solution, the space splitting convexification (SSC) approach presented in the second part aims at reducing computation times for numerical optimal control. By introducing convex approximations in an iterative algorithm, the computation time is greatly lowered. This approach is only applicable to a selected class of nonconvex OPs but provides convexification measures that can be combined with other approaches to extend the application range. Due to its low computation times and robust convergence, the SSC method seems to be a promising approach for real-time optimal control applications.
This chapter recapitulates the state of the art regarding lap time optimisation and convexification in optimal control in Section 1.1 and Section 1.2, respectively. These sections partly contain paragraphs of our previous publications [185], [183] and [186]. Summarising remarks and an outline of this thesis are provided in Section 1.3.

1.1 State of the Art for Lap Time Optimisation

Minimising an objective for a dynamic system under given restrictions can be cast as a trajectory OP. Typically this task is tantamount to finding the time-dependent inputs for the dynamical system that minimise or maximise a cost functional while satisfying additional constraints. The goal for time-optimality is generally final time minimisation, velocity maximisation or distance maximisation over a fixed time period. The methods used to solve such problems can be split into four groups: variational methods, quasi steady-state (QSS) optimal control, graph search methods and incremental search methods. Furthermore, the difficulty of the problem can be reduced by approximating the task via optimal tracking of a predetermined trajectory. The reference trajectories can be identified by preceding trajectory optimisations, geometrical computations or measurements. Various methods for tracking have been used in the automotive field including QSS simulations, model predictive control (MPC), linear and nonlinear feedforward and feedback structures as well as iterative learning controllers. Generally, the methods differ in the compromise between accuracy and required computation time.
This section reviews the methods and applications for lap time optimisation of vehicles used in literature. Fig. 1.1 depicts the classification of the individual approaches, which are discussed more thoroughly in the following sections.

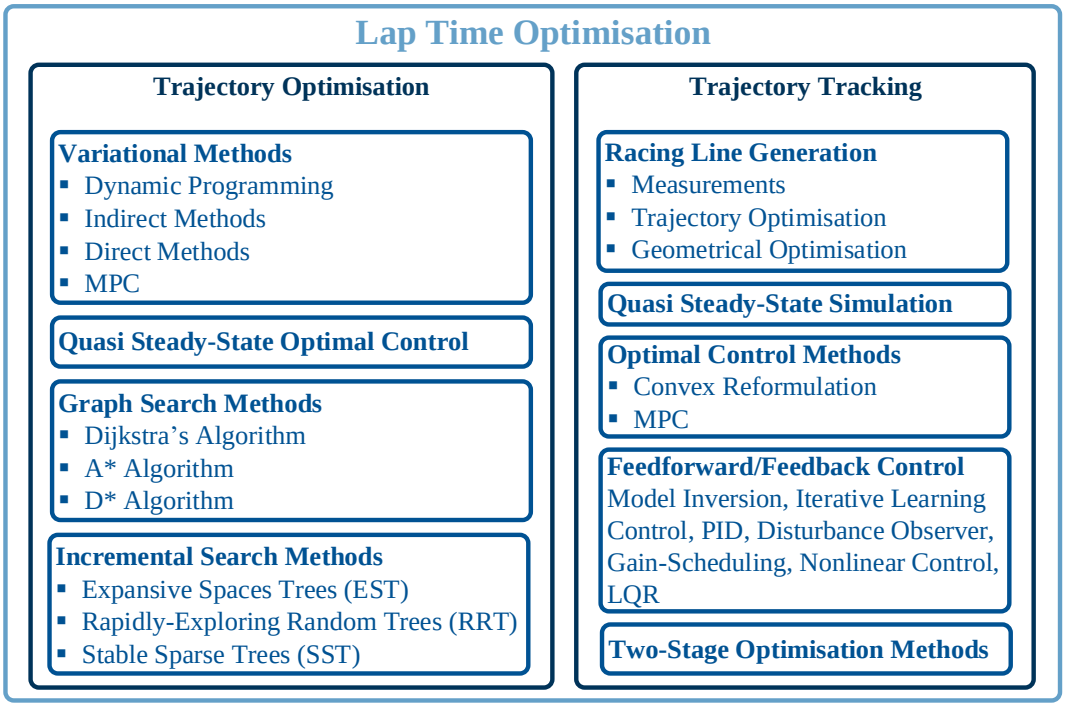


Figure 1.1: Classification of literature for lap time optimisation.

1.1.1 Trajectory Optimisation Methods

The straightforward approach for coping with the previously mentioned optimisation task is solving a trajectory OP. This approach provides flexibility in problem formulation however generally requires longer computation times. This section presents literature regarding trajectory optimisation methods applied for vehicular applications.

Variational methods Using calculus of variations, optimal control methods consider the dynamic system behaviour and compute the system inputs that optimise a predetermined objective while meeting specified constraints. Variational methods also provide the optimal state trajectories enabling the identification of the optimal racing line. Due to highly nonlinear vehicular system models and nonlinear path constraints as well as the minimum-time objective, accurate lap time optimisations for vehicles generally result in nonlinear, nonconvex OPs. These problems are generally hard to solve and require longer computation times. Many implementations use nonlinear programming (NLP) solvers that compute a solution based on derivative information. These solvers tend to converge to local optima unless an appropriate initial guess is provided. One way to mitigate this problem is using evolutionary algorithms to find the global optimum, like the genetic algorithm in [131, 132]. The downside of these algorithms is a generally slower convergence rate. Variational methods can be clustered into dynamic program-

ming (DP) [53, 56, 198, 224], indirect methods (IMs) [10, 15, 16, 39, 80, 124, 125, 177, 203] and direct methods (DMs) [29, 30, 54, 95, 117, 118, 172, 183, 185]. DP decomposes the OCP into simpler subproblems that are solved recursively from back to front. IMs solve a boundary value problem (BVP) that arises when applying Pontryagin's minimum principle to the OCP. DMs discretise the OCP of infinite dimension yielding a static optimisation problem (SOP), which can be solved using NLP solvers. More details on the individual methods are given in Section 2.3.1. A direct comparison of DMs and IMs for vehicular applications is shown in [16].
Technically, discrete decisions like gear shifting have to be considered when controlling vehicular systems. Sometimes these decisions can be approximated enabling the use of NLP solvers. Otherwise, the solution of a mixed-integer programming (MIP) problem is necessary. However, these kind of problems require specialised solution methods and generally longer computation times. In order to include optimal gear selection or engine on/off choices for hybrid electric vehicles (HEVs), MIP has been applied for DMs [66, 101, 172] and IMs [151].
Due to the flexibility and accuracy of variational methods, a large body of research has been conducted in the field of vehicular optimal control. Table 1.1 provides an overview over the available literature addressing the individual focal points for various applications with combustion engine vehicles (CEVs), HEVs, battery electric vehicles (BEVs) and wheel-independent drive vehicles (WIDVs).
Numerous studies have employed moving horizon techniques to enhance computation time of vehicular OCPs. The computational complexity of OPs increases among other

Table 1.1: Vehicular applications using variational methods.

application focus	**literature**
effect of vehicle parameter variation on lap times	[158]
accuracy analysis for model simplifications	[124]
suspension design for minimum lap times	[137],[15]
varying road friction conditions on racetracks	[35]
effect of road surface on lap times	[203]
longitudinal torque allocation	[203]
passive limited-slip differentials	[95],[158],[118],[116],[213]
semi-active limited-slip differentials	[212]
thermodynamic tire model and tire wear	[95],[96],[212],[131],[227]
energy management and energy recovery for HEVs	[118]
fuel consumption for HEVs	[136]
straight acceleration with gear shifting for BEVs	[151]
energy management for BEVs	[81]
thermodynamic modelling of electrical components	[82]
torque allocation for WIDVs	[43],[32],[31],[44]
real-time torque allocation for WIDVs with simplified EOL	[55]

things with the number of decision variables. Thus, reducing the size of the OP by considering a limited preview horizon enhances solvability. Especially for shooting methods, convergence behaviour improves by shortening the considered time horizon due to following reasons. Firstly, a shorter time horizon yields simpler constraints with less couplings between the decision variables. Secondly, stability problems may occur when numerically integrating over longer time periods. However, other problems can arise from moving horizon techniques, as will be explained in Section 2.3.3. In [65, 67], a minimum lap time OCP has been solved by dividing the racetrack into segments yielding multiple smaller OCPs. The individual problems have been solved sequentially whereas the initial condition for the individual OCPs has been determined by the preceding OCP. This approach represents a modified MPC approach since an OP is not solved in every time step but rather at the transition to the next segment. Classical MPC has been used for vehicle models of various complexities. An early implementation of nonlinear MPC for a minimum lap time objective is given in [163]. In order to further reduce computation times, MPC has been applied to linear time-varying (LTV) vehicle models in combination with a system linearisation around estimated trajectories [207–209, 211]. By introducing approximations, the OP has been transformed into a convex quadratic programming (QP) problem, which can be solved efficiently. Additionally, disturbance rejection has been improved in [210] by extending the MPC-approach with an ancillary gain-scheduled linear quadratic regulator (LQR). By approximating the NLP problem using local convex QP approximations at each sampling time, a real-time application of nonlinear MPC for vehicle systems has been presented in [119, 220].

Quasi Steady-State Optimal Control A hybrid method between QSS simulation and optimal control has been presented in [126, 217]: QSS limitations on the acceleration of a body are computed in form of a g-g-v-diagram, which is extended if three-dimensional tracks are investigated. For vehicles, these acceleration limits represent the summary of individual effects like tire friction characteristics, aerodynamic effects and power limits. Using the body accelerations as system input, an OCP is solved for a point mass model with path constraints for the racetrack boundaries and acceleration limits via the g-g-v diagram defining physical performance limits. Once the trajectories for the accelerations and velocity are computed, the nonlinear equations describing the QSS behaviour of a more elaborate vehicle model are solved for its states and inputs. Hence, more specific constraints on the states and inputs of the elaborate vehicle model are not directly included in the OP but rather represented to some extent by the acceleration limits. This greatly reduces the size and complexity of the OCP, which enhances computation time. Opposed to QSS simulations, the approach concurrently optimises the travelled path identifying the optimal racing line. However, various simplifying assumptions are used to derive the g-g-v maps and only steady-state conditions are considered resulting in a suboptimal solution. Nevertheless, the method offers a compromise between accuracy and required computation time.

Graph Search Methods, Incremental Search Methods and Hybrid Methods Alternative trajectory planning methods are based on searching spatially or spatiotemporally discretised graphs. In the automotive field, these techniques have been mainly applied in motion planning for autonomous driving. For completeness, this paragraph outlines these procedures however a more detailed overview is given in [70, 92, 150].
In order to mitigate the problem of local convergence, graph search methods identify the optimal trajectory by searching the minimum-cost path in a graph. The graph represents the discretised configuration space of the vehicle with vertices depicting vehicle configurations and edges the transitions between them [150]. The search graph can be generated by recursively applying motion primitives starting from the initial vehicle state. Possible motion primitives for vehicular applications are arcs with different steering angles, simulations with a sampled number of inputs, clothoid segments and recorded vehicle motions driven by experts. Popular graph search strategies are Dijkstra'a Algorithm [50], the heuristic search algorithm A* [76], the real-time replanning algorithm D* [195] and various extensions [150]. Exemplary applications of graph search methods for vehicles are given in [194, 231]. However, graph search methods only search over the finite set of paths constructable from the motion primitives in the graph, which may return suboptimal paths or even fail to return a feasible one.
In order to overcome the drawbacks of graph search methods, incremental search methods build increasingly finer discretisations of the configuration space until a satisfactory solution is found. The resulting reachability graph, which is often a tree, maintains a discrete set of reachable configurations and feasible transitions between them [150]. Popular randomised tree-based incremental planners are expansive spaces trees (ESTs) [83], rapidly-explorig random trees (RRTs) [112], stable sparse trees (SSTs) [113] and various extensions [150]. Exemplary racing applications of incremental search methods are given in [84, 85] using RRTs.
Hybrid procedures of the previously mentioned methods enable capitalising on the advantages of the individual methods. In [51], a modified A* search algorithm has been used to compute a solution that is then improved via numeric nonlinear optimisation. MPC and an extended version of the D* algorithm have been used in [214] for autonomous driving. In [4], SSTs have been combined with an extended version of RRTs to reduce the computational burden yielding a fast convergence to the optimal solution. Then, nonlinear MPC has been used to find the attracting area for the generated trajectory under consideration of the constraints on the system.

1.1.2 Trajectory Tracking Methods

In order to reduce computation time, various methods have been proposed that approximate the trajectory OP by the task of tracking a specified reference trajectory. However, by fixing the trajectory, these approaches generally do not consider the influence of vehicle setups on the optimal racing line and thus yield suboptimal solutions. This section presents literature regarding trajectory tracking strategies for vehicular applications.

Racing Line Generation Trajectory tracking methods with the goal of minimum lap times require a predefined racing line. If experimental data is available, a driven line can be used as reference trajectory. Alternatively, the solution of previous computations, based for instance on variational methods, can be used. However, racing lines can also be computed geometrically. As shown in [20], the optimal racing line is a linear combination of the shortest path and the minimum-curvature path. However, the authors have shown that the minimum-curvature path is generally close to the fastest racing line since it allows highest cornering speeds at a given maximum lateral acceleration. Thus, the fastest path can be identified by sampling various linear combinations of the shortest path and the minimum-curvature path [20] or the racing line can be approximated by the minimum-curvature path [20, 78]. Another approach for the generation of racing lines based on track partitioning has been presented in [64, 205]: Each corner of the racetrack is decomposed into straights, an entry clothoid, a constant radius arc and an exit clothoid. The parameters of these subsections are identified by consideration of initial conditions and continuity conditions. A similar approach based on the concatenation of circular arcs and straights has been presented in [171].
Once a racing line is generated, it can be used as reference for the tracking approaches presented in the following paragraphs.

Quasi Steady-State Simulation QSS simulations offer good robustness, short computation times and the capability to consider the limitations of complicated models via diagrams, however generally neglect most of the transient behaviour [16]. The approach requires a predefined racing line and thus falls into the group of trajectory tracking. In order to achieve minimum lap times, friction forces from the tires should be maximised. However, the transmittable tire force is generally different for each tire since it depends on the normal loads, slip conditions and suspension geometries. Additionally, limiting effects given by the powertrain, like engine constraints, have to be considered. Complicated equations are necessary to depict all these relations. However, complexity is reduced by using g-g-v diagrams instead, which lump the transmittable forces of all tires together depicting limits on the longitudinal and lateral acceleration as a function of the vehicle speed [142, 170]. These limits are identified for steady-state operating points. Using these bounds and the given curvature profile of the racing line, the optimal speed profile of a point mass model is computed via forward-backward integration [22, 25, 145, 190, 223]. Assuming zero longitudinal acceleration in a first step, the initial speed profile is computed using the racing line curvature and the maximum lateral acceleration. The resulting speed profile is saturated by the predefined maximum speed value of the vehicle. Starting at the apexes of the curvature profile, the velocity of the next discretisation point is computed using the velocity and available longitudinal acceleration of the current point. This procedure is repeated until intersecting the previously computed speed profile. The same approach is used in a backwards integration step considering the deceleration limits yielding the braking points and therefore the final speed profile. In [77], the authors have improved computation time by primarily using forward integration and only temporary backward loops if needed. Furthermore, the QSS approach

has been extended in [156] to consider quasi-transient conditions by additionally using acceleration limits on yaw movement.

Optimal Control Methods In [120, 218], the authors have shown that the minimum-time optimal tracking problem can be transformed into a convex OP via a change of variables. The resulting problem can be solved efficiently using optimal control methods. This approach has also been pursued in [44] for an electric vehicle with four wheel-independent drives. Therein the authors have presented a convex formulation of the time-optimal path following problem using direct collocation together with various model simplifications.
Various MPC approaches have been used for tracking. An MPC approach using direct collocation has been employed in [73] to track an offline computed racing line and simultaneously avoid obstacles. Time-variant predictive path tracking control has been presented in [93, 94] using time-varying models linearised around different sets of side slip angles. The sequence of linearisation points has been chosen according to an estimated trajectory based on predictions from the preceding iteration. By iteratively linearising the system around predicted trajectories in [28], the moving horizon, nonlinear OCP has been approximated by a QP problem using a direct shooting scheme. The presented controller is capable of tracking a predefined line and simultaneously avoid collision with obstacles.

Feedforward and Feedback Controllers Various control strategies have been introduced for vehicular applications aiming at tracking position and velocity profiles. The curvature and velocity profiles identified via QSS simulations can be used as reference trajectories. Furthermore, the associated acceleration profile can be utilised to consider wheel load distributions. Tracking of the velocity and curvature trajectories has been accomplished via linear and nonlinear feedback controllers, gain-scheduling controllers, feedforward structures deduced via model inversion and disturbance observers [20, 62, 78, 90, 108, 109]. Furthermore, assuming constant disturbances over the course of one lap, iterative learning control has been used in [88] for lap to lap improvements.
Various preview controllers based on LQR theory have been used for tracking, computing individual gains for the preview errors that vary over the preview horizon. This approach has been used to compute front steering angles that minimise position and orientation errors between the vehicle and a position trajectory at constant longitudinal velocities [38, 206]. The same approach has been used in [189], however a decoupled longitudinal vehicle model has been inverted to compute the longitudinal forces required to simultaneously follow a velocity profile. In [188], the LQR preview approach has been adjusted for tracking of a velocity profile.
As mentioned, the reference trajectories for tracking can also be generated using optimal control methods. Indirect optimal control has been used in [123] to compute time-optimal lap trajectories for a simplified model. These trajectories have been tracked using proportional, integral and derivative feedback controllers that have been designed via the

Nyquist theorem. In [23, 24], optimal trajectories have been computed using MPC with direct collocation and have been tracked in real-time via a nonlinear position controller.

Two-Stage Optimisation Methods Several two-level procedures have been developed to improve accuracy while keeping small computation times. As mentioned before, using a predetermined racing line yields a suboptimal solution since the influence of the vehicle setup on the driven line is neglected. In order to mitigate this problem, these approaches iteratively solve a trajectory tracking problem and appropriately adjust the reference trajectory based on the result.
Such a scheme has been illustrated in [89]. Starting from a specified racing line given by its curvature trajectory, the optimal velocity and acceleration profile are computed using a QSS simulation. The resulting speed profile is then used to determine a sequence of LTV vehicle models that vary over velocity. A convex OCP aiming at minimising the driven curvature while staying on the racetrack is solved for this previously computed model sequence. The novel curvature profile is subsequently used to compute a new speed profile and the procedure is repeated until the predicted lap time no longer improves.
A different two-level approach, which is real-time capable, has been presented in [221]. An upper level algorithm has been used to identify the reference trajectory by either minimising its curvature or its length. Then, this reference path has been tracked by a low-level MPC controller.
Another method for identifying the ideal reference path has been employed in [119]. Performing a stationary velocity analysis for discrete values of longitudinal speeds and steering angles generates a family of trajectories. The feasible, most cost-optimal trajectory is selected as reference trajectory for tracking via MPC. Similarly, a set of trajectories has been generated in [199, 225, 226] by using state space sampling and solving boundary value problems for multiple terminal points under the assumption of polynomial trajectories. The feasible trajectory with the best cost has been used for a subsequent trajectory tracking via optimal control methods. However, due to required model simplifications, these approaches have been rather used for online trajectory planning with collision avoidance than for controlling vehicles at the limits of handling.
A real-time capable two-stage MPC algorithm for controlling the energy management of HEVs aiming at minimum lap times has been presented in [176]. Considering only longitudinal dynamics of a point mass model, a high-level MPC has been used to compute the optimal system trajectories by solving a second-order cone programming (SOCP) problem. Linearising the dynamics around the optimal trajectories, a low-level zone MPC scheme has been employed to track time-varying sets defined by the high-level solution.

1.2 State of the Art for Convexification in Optimal Control

Many OCPs in engineering applications require the solution of nonconvex OPs. Unfortunately, these problems are much harder to solve and generally demand longer computation times. However, in order to apply optimal control within a real-time capable framework usually requires a fast and robust computation of the solution. Convex optimisation is beneficial in this regards since it enables using efficient solvers that rapidly converge to the single global optimum. Thus, a large body of research has been conducted to find convex formulations or approximations of OCPs. The nonconvexity in OCPs can have various causes: nonconvex objectives, restrictions in the operating range of actuators that result in nonconvex sets, nonlinear equality constraints due to nonlinear system models and more. A classification of OPs and subclasses of convex OPs are provided in Section 2.4.1.
Optimal control in form of an on-board controller is generally realised using MPC. Based on a predefined model, this control approach computes optimal input trajectories by solving an OCP for a limited time horizon and a given initial solution. The initial solution is updated in each time instant based on measurements or estimates of the current state, which closes the control loop. Solvability of the OP is enhanced due to the limited time horizon and thus smaller problem size. However, considering only limited time horizons in the presence of constraints requires precautions to ensure recursive feasibility [122]. Further information on MPC is given in Section 2.3.3. Nevertheless, limiting the time

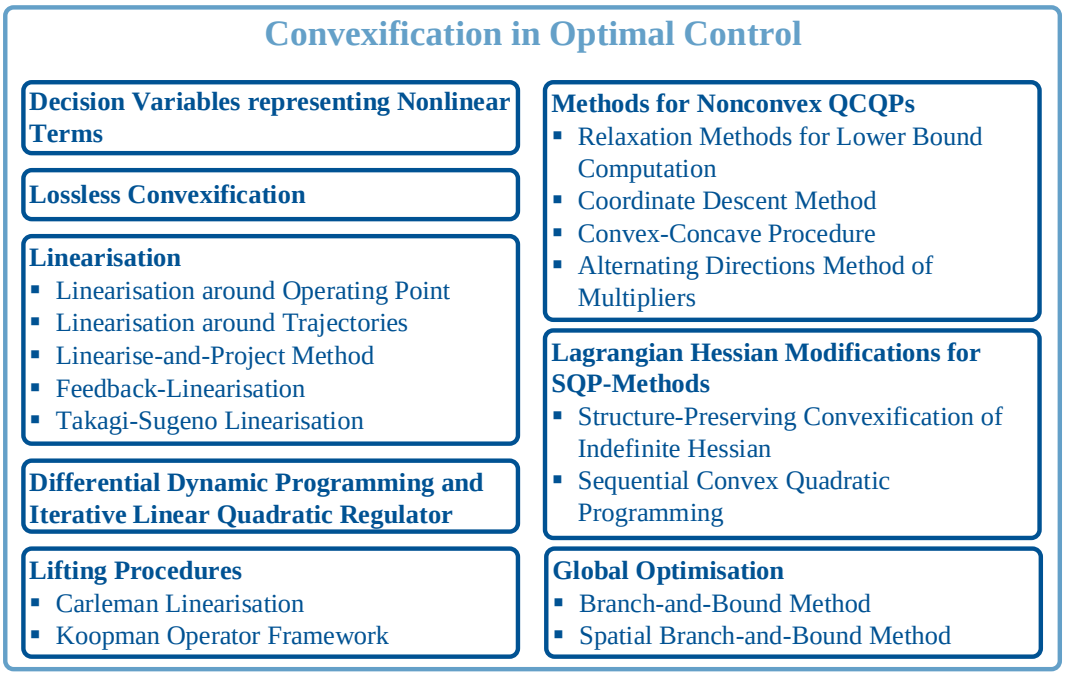


Figure 1.2: Classification of literature for convexification in optimal control.

horizon is often not sufficient for real-time capability. Thus, the following paragraphs present approaches that apply convex approximations or reformulations, mainly to reduce computation time. These approaches are visually classified in Fig. 1.2.

Decision Variables representing Nonlinear Terms When formulating OCPs, a suitable choice of decision variables can sometimes reduce the complexity of the OP. Replacing a nonlinear term by a decision variable excludes the nonlinearity from the OP, which can render it convex. After solving the problem, the original variables can be identified by inverting the nonlinear term. For this approach to work, the excluded nonlinearity must be invertible. However, simple constraints restricting the original variables incorporated in the nonlinear term are cast into a more complicated constraint if a nonlinear transformation needs to be considered. Hence, this approach is only applicable to few OCPs. For example, choosing the forces as decision variables for vehicles with wheel-independent drives and wheel-independent steering greatly simplifies the OP enabling a fast computation of the optimal solution via QP [31, 44]. The torques and steering angles associated with the computed optimal forces can be deduced afterwards by inversion of the tire force equation. Unfortunately, this approach is generally not possible for industrial vehicles, since the centralised steering system and propulsion system introduce physical limitations that would result in nonlinear and nonconvex constraints.

Lossless Convexification A special class of nonconvex sets can be transformed into convex ones via lossless convexification (LC) [1, 2]. This method lifts the optimisation space into a higher dimension using additional decision variables. The approach is applicable to nonconvex sets that can be generated by removing a convex subset from a convex set. This has been used in aeronautic optimal control applications to successfully convexify annulus-shaped sets. The method produces a SOCP problem, which represents a convex OP that is efficiently solvable. Under the assumption that the state space is strongly controllable, the solution of the relaxed OP is equivalent to the solution of the original problem. An overview regarding the individual findings on LC is given in [165, p.338].

Linearisation Another often employed technique for convex approximation is linearisation since linear objectives and constraints are convex [19]. However, linearisation can introduce conservativeness and infeasibility since the resulting accuracy strongly depends on the nonlinearity as well as the linearisation point. Various linearisation techniques have been used to approximate nonlinear systems when applying optimal control methods.
In the simplest case, the system is linearised at each time instant around the current operating point, exemplarily implemented in [57]. Using the resulting LTV system, a controller based on linear MPC computes the steering angle of an autonomous vehicle that aims at tracking a predefined trajectory. Additional constraints limiting the front slip angle are necessary to avoid the vehicle becoming unstable. This illustrates the general problem that the resulting linear model approximation is only sufficiently valid in a close neighbourhood around the linearisation point. With increasing distance to this

point, the linearised OP can strongly differ from the original one. Thus, this approach can result in poor controller performance or even lead to infeasibility or instability.
In order to mitigate linearisation errors, the system can be linearised around predicted trajectories resulting in efficiently solvable QP problems [48, 187]. One method using iterative linearisation around trajectories in an MPC setup is the so-called real-time iteration approach [48]. The procedure assumes that at every time instant the shifted solution from the preceding time instant represents a good initial guess close to the actual solution. Thus, a full Newton-step is taken, omitting underlying line-search routines to reduce computation time. Besides using a shifted version of the preceding solution as initial guess, a tangential predictor can be used to prepare the OP for the next time instant [49, 61]. An overview and comparison with other real-time capable nonlinear MPC approaches is provided in [71].
Combining the system linearisation around predicted system trajectories with constraints softening via additional penalty objective terms enables the formulation of an unconstrained, convex OP [147]. The resulting unconstrained MPC problem can be solved in real-time using a sequential linear quadratic solver to compute the optimal controller gains.
In order to improve robustness, approximation errors in predictions can be limited by tubes whose cross-section is simultaneously optimised [26]. The presented MPC scheme results in an efficiently solvable SOCP problem. Furthermore, the approach guarantees recursive feasibility and asymptotic stability making it suitable for real-time applications. In [27], the authors have suggested using probabilistic tubes to consider non-conservative estimations of linearisation errors and stochastic model uncertainties while solving the problem via stochastic MPC.
Another iterative linearisation scheme has been presented in [133]. Starting from an initial trajectory, the nonlinear system equations are successively linearised around the solution computed in the preceding iteration. Adding virtual control inputs eliminates the artificial infeasibility introduced by the linearisation. Furthermore, including trust regions ensures that the solution does not deviate too much from the preceding succession and thus the linearisation represents a valid approximation. This avoids unboundedness of the linearised OP. Based on the ratio between actual objective change and linearised objective change, the algorithm adjusts the trust regions and terminates if the objectives coincide.
A linearise-and-project (LnP) approach has been used in [134] to resolve the nonconvexity due to nonlinear system equations and nonconvex constraints. The approach presumes convex functions for the right-hand side of the system differential equations. The satisfaction of the system dynamics is ensured by relaxing the corresponding equality constraints into inequality constraints and using exact penalty functions. In an iterative optimisation routine, the computed solution is projected onto the constraints to gain adequate linearisation points. The projection is accomplished by solving a subordinate convex programming problem.
Instead of linearising the system equations around trajectories, a linear input-to-state behaviour can be enforced via feedback-linearisation [98, p.505]. An optimal control method can then be used to optimally control the linearised system in an outer control

loop [45]. Mostly, a double-integrator behaviour is imprinted by the lower control loop but different linear behaviours can be prescribed as well. However, the enforced linear system behaviour is not necessarily optimal and can thus result in an suboptimal overall solution as the optimiser could potentially find a better solution considering the real system dynamics. Furthermore, the feedback controller must be designed under consideration of stable internal dynamics, which requires additional effort regarding rather complicated stability proofs. Moreover, it is possible that the nonlinear mapping of the feedback linearisation transforms originally convex objective and constraint functions into nonconvex ones [191]. Then, the exact satisfaction of these constraints requires an NLP strategy or an iterative scheme, which both eliminate the computational benefits of the feedback-linearisation [111]. One possible solution to this problem is approximating these constraints by estimating future inputs based on the preceding time instant within an MPC scheme [111, 180]. The authors in [191] have proposed an alternative MPC approach that replaces such nonlinear constraints via dynamically generated local inner polytopic approximations rendering the OP convex.
Fuzzy MPC with models of Takagi-Sugeno (TS) type has been successfully used for nonlinear MPC. The TS modelling procedure enables an accurate approximation of nonlinear systems by using data combined with model knowledge [200]. The modelling approach decomposes nonlinear models into several local approximations that are blended together via fuzzy rules. However, this requires the system matrices to be stored, thus occupies more memory. Although TS fuzzy models have been successfully used in nonlinear MPC, computation time can be strongly reduced by linearising the TS models around predicted trajectories [143, 144]. The resulting convex QP problem can be solved efficiently while guaranteeing convergence via a line search mechanism.

Differential Dynamic Programming and Iterative LQR Differential dynamic programming (DDP) initially linearises the system equations around a nominal trajectory. Starting from the terminal state, a backward pass identifies optimal LQR gains for each time step using the linearised system. Subsequently, a forward pass computes new nominal trajectories via numerical integration using the controller gains previously computed in the backward pass. This procedure is repeated until convergence. Iterative linear quadratic regulators are a variant of DDP [141]. The main difference is that only first-order derivatives are used for the approximation of the system dynamics via Taylor expansion instead of second-order ones, which reduces computation time [201]. Constrained versions of this optimisation technique have been presented in [33, 202, 229].

Lifting Procedures Lifting-based MPC methods represent another class of linearisation approaches used for solving OCPs with nonlinear systems. By lifting the nonlinear dynamics to a space of higher dimension, the evolution of the system can be represented in a bilinear or linear fashion via Carleman linearisation or the Koopman operator framework, respectively. Representing the nonlinear system equations via bilinear terms enables the analytical computation of sensitivity functions [5] and of solutions [60], which speeds up computation time. The Koopman MPC approach employs a linear predictor of higher

order, which is identified via data sets, to approximate the nonlinear system. The linearly evolving substitute system is then used to solve a larger OCP via linear MPC. By condensing the OCP, the state decision variables are eliminated leaving only the input decision variables. Thus, the computational complexity is comparable to an OCP with linear system dynamics of the same size [104]. Hence, this method enables a fast solution of nonlinear OCPs. However, identifying the linear Koopman system involves some effort requiring an adequate selection of basis functions, non-recurrent data sets and the solution of convex OPs [36, 104]. As illustrated for a vehicular OCP in [37], the Koopman MPC approach does not always outperform the standard MPC method using local linearisations. Depending on the dimension of the lifted states, both approaches can require the storage of a great number of offline computed matrices.

Methods for Nonconvex QCQPs Various methods for solving nonconvex quadratically constrained quadratic programming (QCQP) problems are summarised in [42] and [153] and subsequently outlined.
A two-phase coordinate descent method first reduces the maximum constraint violation trying to find a feasible point. The second phase is restricted to feasible points only and searches in each iteration for a feasible point with a better objective function value.
The convex-concave procedure (CCP) is a method for finding a local optimum for difference-of-convex programming problems. Since any quadratic function can be rewritten as a difference of convex expressions, this method is also suitable for QCQP. The nonconvex part of the constraints is linearised rendering the constraints convex. In order to deal with infeasible initial guesses, penalty CCP relaxes the linearised constraints via slack variables and introduces a gradually increasing penalty objective for constraint violations.
The alternating directions method of multipliers (ADMM) is a variant of the augmented Lagrangian method. It forms an equivalent OP by using auxiliary decision variables that must satisfy a consensus constraint. The objective function is augmented using switching indicator functions that penalise individual constraint violations. The augmented objective function terms are separated and reformulated using two groups of decision variables. Instead of solving the proximal augmented Lagrangian function for both groups of decision variables simultaneously, the solution is computed using an alternating approach. First, the first group of decision variables is fixed and the problem is solved for the second group. Then, the solution of the second group is fixed and the problem is solved for the first one. The results are used for the dual variable update and the procedure is repeated. This alternating approach requires the solution of simplified QCQP problems enhancing computation time compared to solving for both groups of decision variables simultaneously.

Lagrangian Hessian Modifications Convexification measures for sequential quadratic programming (SQP) approaches on a more algorithmic level have been presented in [219]. Many QP solvers do not support an indefinite Hessian of the Lagrangian, since then a descent direction is not guaranteed. The author has presented a structure-preserving

convexification procedure for indefinite Hessians. The convexified Hessian can be fed to any structure-exploiting QP solver [219, p.39]. Furthermore, sequential convex quadratic programming is proposed, which uses second-order derivatives of convex objective functions and convex constraint functions to construct positive definite Hessians enabling the sequential solution of convex QP problems [219, p.70].

Global Optimisation Although global optimisation (GO) methods have greatly improved in computational effort lately, they generally do not aim at real-time capable implementations but rather focus on computing the global optimum. They can also be employed for solving OPs with discrete decision variables such as integer programmings (IPs) and MIP problems. GO techniques make use of convexification and have similarities with the SSC approach presented in the second part of this thesis. Thus, this paragraph reviews selected GO methods however a thorough summary on the state of the art is discarded. The interested reader is referred to [100, 110, 114, 115, 204].
A commonly used GO algorithm is the branch-and-bound (BnB) approach [110, 204]. By replacing the integer variables via continuous, bounded variables, this method transforms IP and MIP OPs into continuous ones. Solving such relaxed problems provides a lower bound on the objective function. If the gap between this lower bound and an estimated or computed upper bound is larger than a predefined tolerance, the domain of the discrete variables is split into subdomains. Again, the integrality constraints are relaxed on each subdomain and corresponding lower bounds are computed. A subdivision into increasingly smaller segments is repeated until the gap between the bounds is smaller than a predefined tolerance. Then, the solution of the relaxed problem represents the optimal solution of the corresponding subdomain. The iterative division into increasingly smaller subregions ensures progressively tighter convex relaxations assuring convergence to the local optimum on the subdomain. Comparing the solutions on the individual subdomains provides the global solution. Individual subdomains can be rejected on the fly from further consideration based on the solution of the current branching step. This enables an exhaustive exploration of the search space in a rather efficient manner and provides the global optimum.
Spatial BnB applies this notion to compute the global optimum of continuous, nonconvex OPs [114, 174, 204]. The search space is iteratively divided into finer subdomains and convex relaxations of nonconvex functions are applied on each subdomain. The efficiently solvable relaxed problem provides a lower bound of the objective function for the corresponding subdomain. Using local minimisation techniques, an upper bound for the objective function can be computed for each segment. If the gap between the upper and lower bound on a subdomain is bigger than a predefined tolerance, the subdomain is split again into subregions. This approach can be used to solve mixed-integer nonlinear programming (MINLP) problems by branching integer variables and continuous variables: Nonconvex terms are replaced by their convex envelopes yielding a convex OP whose solution represents the lower bound. Various methods have been developed for the generation of adequate convex relaxations. For instance, by introducing adequate auxiliary decision variables with corresponding constraints, nonconvex functions

can be decomposed into sums and products of terms for which convex relaxations are available [193][204, p.125].
In summary, the BnB technique searches over a tree whose nodes correspond to relaxed problems that consider different subdomains. Tree nodes are excluded on the fly based on evaluating the lower and upper bounds of the individual nodes eliminating the exploration of the entire domain. This procedure reduces the combinatorial complexity of solving the problem for each possible combination of integer variables and provides the global optimum.

1.3 Outline and Contributions

The preceding sections clarified the importance of optimal control methods as well as numerical optimisation and reviewed literature on this matter. The following paragraphs outline the structure and content of this thesis, which deals with optimal control methods for the offline solution of OPs. It is assumed that the used models are accurate and possible uncertainties are negligible. Since the considered problems include the entire time horizon, infeasibility and stability issues are not examined. Transferring these methods to real-time capable controllers via MPC requires the investigation of these properties. In case of model uncertainties, further precautions can be necessary for robustness.
The main contributions of this thesis are summarised in following list:

- Implementation of a framework capable of solving minimum-time OCPs for a large class of highly nonlinear models with difficult constraints but continuous variables.
- Development of a novel track preprocessing method for the generation of smooth reference trajectories for OCPs with path constraints.
- Experimental validation of a two-track vehicle model for racetrack applications.
- Derivation of simple substitute models for the consideration of electrical overloading in OCPs.
- Development of a novel successive convexification method for the solution of a specific class of nonconvex OCPs with continuous variables.

Chapter 2 recapitulates the basics of optimal control. Starting with the mathematical definition of OPs, conditions for optimality are presented and the link to dynamic OPs, which occur in the context of optimal control, is established. Furthermore, general solution methods for OCPs are compared. Since this thesis employs a numerical solution of OCPs, the majorly used numerical solution methods are presented. The individual OP categories are presented and advantages of individual classes are elaborated.
Chapter 3 presents the first focus of this thesis. Aiming at solving minimum-time OCPs for highly nonlinear systems with nonconvex constraints but continuous variables, a general approach is presented that is applicable to a large class of systems. Direct Hermite-Simpson collocation is used to transcribe the dynamic OP into a static one, which is solved using NLP. In order to enhance solvability of such OPs, various preliminaries are proposed. Using a model transformation, the OP is reformulated in

spatial coordinates, which fixes the final value of the independent variable. A scaling procedure is implemented to equalise the domains of decision variables as well as constraints and get a numerically suitable range for the objective function. Since the employed interior-point method (IPM) solver uses derivative information, the OP must be sufficiently differentiable. Recipes for the smoothing of discontinuities are presented and a novel procedure for the generation of smooth reference trajectories is introduced.
Chapter 4 presents a challenging application for nonconvex optimal control. The NLP algorithm presented in Chapter 3 is used to solve the nonconvex OP that arises when searching for the minimum lap time trajectories of a highly nonlinear vehicle model on a racetrack. A nonlinear two-track vehicle model is validated using measurement data of a test vehicle. The model is augmented by electro-chemical relations to depict the behaviour of a BEV. The chapter shows how the overloading of electrical components can be considered in an OCP while keeping good convergence properties. Furthermore, an initialisation routine producing good initial solutions is proposed to improve convergence and reduce computation time. Employing a concurrent optimisation, lap time-optimal control inputs as well as optimal vehicle parameters are identified for a specific racetrack. The results are discussed in terms of plausibility to verify the performance of the algorithm. The solution computed by the presented framework provides information about strategies that can be used by racing teams to improve lap times. Furthermore, the optimal trajectories can serve as reference when developing real-time capable controllers. Additionally, identifying optimal passive vehicle designs is an important process for racing cars but for industrial cars as well. This framework can be used to shorten development times of vehicles and compare individual vehicle setups with different actuators in regards to various objectives.
Chapter 5 deals with the second major topic of this thesis: the convexification of simpler but still nonconvex OCPs with continuous variables aiming at reducing computation time. The novel SSC approach is presented: It computes a local solution of nonconvex OPs by iteratively solving convex QP substitute problems. This procedure greatly reduces computation time compared to the NLP approach. The method is capable of considering certain classes of zonally convex set (ZCS) as well as equality constraints with possibly multiple univariate nonlinearities. The basic idea behind SSC is the decomposition of nonlinearities into affine segments that are interconnected via suitable constraints. Using the theory of exact penalty functions, the convergence to a local solution of a piecewise linear approximation of the original problem is confirmed. Although this local method does not guarantee to compute the global optimum, the SSC algorithm can provide good solutions even for moderate initial guesses.
Chapter 6 showcases applications for the SSC algorithm introduced in Chapter 5. Nonconvex OCPs for single-mass oscillators (SMOs) are solved and compared to the solution of the NLP algorithm. The influence of the initial guess on the outcome of the optimisation is analysed. The experimental results confirm the robust convergence of the method. In order to provide an example in the context of vehicular control, the SSC algorithm is also employed to compute optimal trajectories for a drag race application.
Chapter 7 closes the thesis with summarising remarks on the covered topics and provides an outlook for future work.

2 Optimal Control

This thesis focuses on variational methods. In order to classify the utilised approaches, a thorough theoretic review is given in this chapter. Basics regarding static OPs are given in Section 2.1. This forms the foundation for dynamic OPs, which arise when optimising dynamical systems. The corresponding mathematical background for optimal control is provided in Section 2.2. Differences in the major procedures for solving OCPs are reviewed in Section 2.3. Since the solution of complicated OPs generally requires numerical approaches, the major algorithms for the numerical solution of OPs are outlined in Section 2.4.

2.1 Static Optimisation

OPs can be differentiated into static OPs, for which the decision variables are elements of the Euclidean space, and dynamic OPs, for which the decision variables are elements of the Hilbert space [152, p.6]. Simply stated, static OPs have parameters as decision variables, whereas dynamic OPs determine functions of an independent variable. Deriving optimality conditions for dynamic OPs is similar to the procedure for static OPs. Furthermore, dynamic OPs can be approximated by static ones. Thus, the mathematical theory on static optimisation presented in this section lays the fundament for later considerations.

2.1.1 Static Optimisation Problems

The task of a constrained SOP is finding the optimal point $\mathbf{w}^\star$ for the vector of decision variables $\mathbf{w} \in \mathbb{R}^{n_w}$ that minimises an objective function $J : \mathbb{R}^{n_w} \to \mathbb{R}$ while satisfying equality constraints $\mathbf{c}(\mathbf{w}) = \mathbf{0}$ with $\mathbf{c} : \mathbb{R}^{n_w} \to \mathbb{R}^{n_c}$ and inequality constraints $\mathbf{h}(\mathbf{w}) \leq \mathbf{0}$ with $\mathbf{h} : \mathbb{R}^{n_w} \to \mathbb{R}^{n_h}$. The constrained SOP can be formulated as

$$\min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad J(\mathbf{w}) \tag{2.1a}$$

$$\text{s.t.} \quad c_i(\mathbf{w}) = 0 \quad \forall i \in \mathcal{E} \coloneqq \{1, \dots, n_c\}, \tag{2.1b}$$

$$h_j(\mathbf{w}) \leq 0 \quad \forall j \in \mathcal{I} \coloneqq \{1, \dots, n_h\}. \tag{2.1c}$$

At any feasible point $\mathbf{w}$, an inequality constraint is called active if $h_j(\mathbf{w}) = 0$ and inactive if $h_j(\mathbf{w}) < 0$ for some integer $j \in \mathcal{I}$. The following index sets are defined for the inequality constraints:

$$\mathcal{I}_0(\mathbf{w}) \coloneqq \{j \in \mathcal{I} | h_j(\mathbf{w}) = 0\} \tag{2.2a}$$

$$\mathcal{I}_-(\mathbf{w}) \coloneqq \{j \in \mathcal{I} | h_j(\mathbf{w}) < 0\}. \tag{2.2b}$$

Therein $\mathcal{I}_0$ and $\mathcal{I}_-$ represent the index set for the active and inactive inequality constraints, respectively. The index set for all active constraints is defined as

$$\mathcal{A}(\mathbf{w}) \coloneqq \mathcal{E} \cup \mathcal{I}_0 \tag{2.3}$$

containing the indices of the equality constraints and active inequality constraints.
A candidate point satisfying the equality constraints (2.1b) and inequality constraints (2.1c) is called feasible. The set comprised of all feasible points is called feasible set and is defined as

$$\Omega_f \coloneqq \{\mathbf{w} \in \mathbb{R}^{n_w} | \mathbf{c}(\mathbf{w}) = \mathbf{0}, \mathbf{h}(\mathbf{w}) \leq \mathbf{0}\}. \tag{2.4}$$

Thus, for unconstrained problems $\Omega_f = \mathbb{R}^{n_w}$ holds. An OP is called infeasible if there is no solution satisfying the constraints (2.1b) and (2.1c) yielding $\Omega_f = \emptyset$.
A major concept of mathematical optimisation is convexity, which is discussed more thoroughly in Section 2.4.1. At this point it is anticipated that strictly convex OPs only possess one solution, which is therefore the global solution. However, a problem can have multiple solutions, which can be divided into local and global ones. A point $\mathbf{w}^\star \in \Omega_f$ is called global solution if following condition holds:

$$J(\mathbf{w}^\star) \leq J(\mathbf{w}) \quad \forall \mathbf{w} \in \Omega_f. \tag{2.5}$$

If (2.5) only holds in a neighbourhood of $\mathbf{w}^\star$, the point is a local solution of the problem. Mathematically, local solutions can be expressed as

$$J(\mathbf{w}^\star) \leq J(\mathbf{w}) \quad \forall \mathbf{w} \in \Omega_f \cap \mathcal{B}_r \tag{2.6}$$

using an Euclidean ball $\mathcal{B}_r \coloneqq \{\mathbf{w} \in \mathbb{R}^{n_w} | \|\mathbf{w} - \mathbf{w}^\star\|_2 \leq r\}$ around $\mathbf{w}^\star$ with some radius r.

2.1.2 Optimality Conditions of Static Optimisation Problems

Conditions on optimality play a major role in optimisation: They provide mathematical guarantees and are used in algorithms to solve the OP. In order to ensure optimality, first-order and second-order conditions exist, which use first-order and second-order derivatives, respectively.
Provided certain regularity conditions are satisfied, the optimum of the static, constrained OP (2.1) can be identified using the the method of Lagrange multipliers, which employs the Lagrangian function

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \coloneqq J(\mathbf{w}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{w}) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{w}). \tag{2.7}$$

Therein each equality constraint and inequality constraint is associated with a Lagrange multiplier, which is stored in the vector $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, respectively. Computing the stationary points of the Lagrangian function (2.7) provides the optimum of the constrained OP if certain constraint qualifications apply. These constraint qualifications are required to ensure that the local optimum satisfies the optimality conditions making them necessary

for an optimal solution[1]. The most often used constraint qualification is the linear independence constraint qualification (LICQ) [148, p.320]. It ensures that the gradients of the equality constraints and the gradients of the active inequality constraints are linearly independent at the local solution $\mathbf{w}^\star$. This condition holds at some point $\hat{\mathbf{w}}$ if the set of active constraint gradients

$$\mathcal{S}_{\text{grad}} \coloneqq \{\nabla_{\mathbf{w}} c_i(\hat{\mathbf{w}}) | i \in \mathcal{E}\} \cup \{\nabla_{\mathbf{w}} h_j(\hat{\mathbf{w}}) | j \in \mathcal{I}_0\} \tag{2.8}$$

is linearly independent.
Assume that the functions J, $\mathbf{c}$ and $\mathbf{h}$ are continuously differentiable[2] and the LICQ holds at the local solution $\mathbf{w}^\star$. Then, a Karush-Kuhn-Tucker (KKT) triple $\mathbf{p}^\star \coloneqq (\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ exists such that the following first-order necessary conditions for optimality, known as KKT conditions, hold:

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \nabla_{\mathbf{w}} J(\mathbf{w}^\star) + \mathbf{c}_{\mathbf{w}}(\mathbf{w}^\star)^T \boldsymbol{\lambda}^\star + \mathbf{h}_{\mathbf{w}}(\mathbf{w}^\star)^T \boldsymbol{\mu}^\star = \mathbf{0} \tag{2.9a}$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \mathbf{c}(\mathbf{w}^\star) = \mathbf{0} \tag{2.9b}$$

$$\mathbf{h}(\mathbf{w}^\star) \leq \mathbf{0} \tag{2.9c}$$

$$\boldsymbol{\mu}^\star \geq \mathbf{0} \tag{2.9d}$$

$$\mathbf{h}(\mathbf{w}^\star)^T \boldsymbol{\mu}^\star = \mathbf{0} \quad \text{or} \quad h_j(\mathbf{w}^\star)\mu_j^\star = 0 \; \forall j \in \mathcal{I}. \tag{2.9e}$$

Proof for the KKT equations (2.9) is given in [148, p.323]. Equations (2.9c)-(2.9e) are called complementarity conditions whereas following distinction between inactive, currently active and strictly active inequality constraints is made:

$$\text{inactive:} \qquad h_j(\mathbf{w}^\star) < 0 \quad \text{and} \quad \mu_j^\star = 0 \tag{2.10}$$

$$\text{currently active:} \; h_j(\mathbf{w}^\star) = 0 \quad \text{and} \quad \mu_j^\star = 0 \tag{2.11}$$

$$\text{strictly active:} \quad h_j(\mathbf{w}^\star) = 0 \quad \text{and} \quad \mu_j^\star > 0. \tag{2.12}$$

A graphical representation of the constraint activity is given in Fig. 2.1.
The KKT equations (2.9) can be satisfied by minimisers but also by other stationary points like maximisers or saddle points. Thus, these equations only provide a candidate for a minimum. Additional information using second-order derivatives is necessary to ensure that the solution is a local minimum. This requires defining the critical cone:

$$\begin{aligned} \mathcal{C}(\mathbf{w}^\star, \boldsymbol{\mu}^\star) \coloneqq \{\boldsymbol{\delta}\mathbf{w} \in \mathbb{R}^{n_w} | &\nabla_{\mathbf{w}} c_i(\mathbf{w}^\star)^T \boldsymbol{\delta}\mathbf{w} = 0 \quad \forall i \in \mathcal{E}, \\ &\nabla_{\mathbf{w}} h_j(\mathbf{w}^\star)^T \boldsymbol{\delta}\mathbf{w} \leq 0 \quad \forall j \in \mathcal{I}_0 \text{ with } \mu_j^\star = 0, \\ &\nabla_{\mathbf{w}} h_j(\mathbf{w}^\star)^T \boldsymbol{\delta}\mathbf{w} = 0 \quad \forall j \in \mathcal{I}_0 \text{ with } \mu_j^\star > 0\}. \end{aligned} \tag{2.13}$$

This set contains the critical directions $\boldsymbol{\delta}\mathbf{w}$ for which the first-order derivative information does not suffice to determine the local behaviour of J [148, pp.330ff][152, p.92]. Assuming that the LICQ holds and the functions f, $\mathbf{c}$ and $\mathbf{h}$ are twice continuously

[1] The optimality conditions will thus miss non-regular solutions.
[2] Further remarks on continuity and smoothness are given in Appendix B.2.

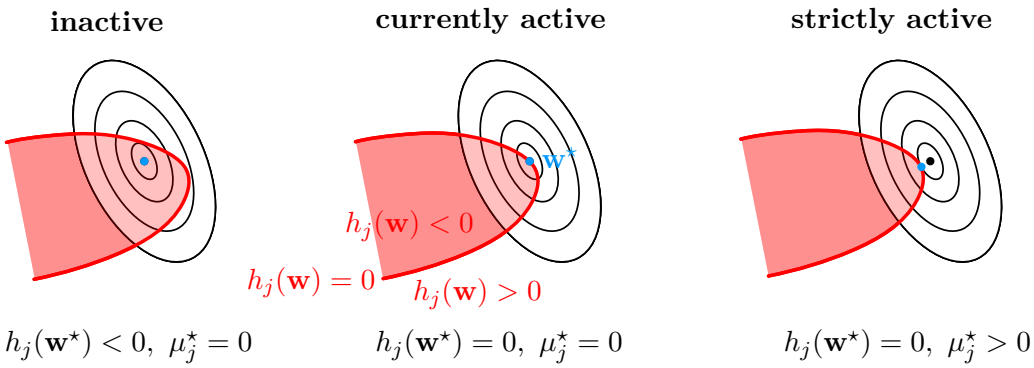


Figure 2.1: Classification of inequality constraints.

differentiable, the second-order necessary condition for a KKT triple $\mathbf{p}^\star := (\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$, which satisfies the KKT condition (2.9), is given by

$$\boldsymbol{\delta}\mathbf{w}^T \nabla_{\mathbf{ww}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) \boldsymbol{\delta}\mathbf{w} \geq 0 \quad \forall \boldsymbol{\delta}\mathbf{w} \in \mathcal{C}(\mathbf{w}^\star, \boldsymbol{\mu}^\star). \tag{2.14}$$

Thus, if the Hessian of the Lagrangian $\nabla_{\mathbf{ww}} \mathcal{L}$ is positive semi-definite on the critical cone $\boldsymbol{\delta}\mathbf{w} \in \mathcal{C}$, the solution $\mathbf{w}^\star$ is a local minimum. Similarly, the second-order sufficient condition declaring a strict local minimum is given by

$$\boldsymbol{\delta}\mathbf{w}^T \nabla_{\mathbf{ww}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) \boldsymbol{\delta}\mathbf{w} > 0 \quad \forall \boldsymbol{\delta}\mathbf{w} \in \mathcal{C}(\mathbf{w}^\star, \boldsymbol{\mu}^\star),\ \boldsymbol{\delta}\mathbf{w} \neq \mathbf{0}. \tag{2.15}$$

After listing the requirements for local minimisers of static OPs, the next section establishes the connection with dynamic OPs.

2.2 Dynamic Optimisation

When dealing with OPs for dynamic systems, the incorporation of the differential equations describing the system behaviour results in a dynamic OP. The basic principles of dynamic optimisation and the link to static optimisation are presented in the following sections.
Real-life systems are time-varying as well as generally nonlinear and stochastic to some extent. This thesis neglects hybrid systems with integer states and is limited to idealised systems. The mathematical background in this chapter is formulated considering continuous, input nonaffine, time-variant systems. These systems can be described by an initial value problem using an ordinary differential equation system with a predefined initial value $\mathbf{x}_0$ and a vector-valued function $\mathbf{f}$:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \qquad \mathbf{x}(0) = \mathbf{x}_0. \tag{2.16}$$

Therein the state vector is represented by the function $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^{n_x}$. The system behaviour can be influenced by applying controls or inputs defined as functions of

time $\mathbf{u}(t) : \mathbb{R} \to \mathbb{R}^{n_u}$. Although neglecting hybrid systems, many physical systems can be represented via the system class (2.16). In order to extend the applicability to systems with nonsmooth right-hand sides $\mathbf{f}$, smooth approximations will be given within this thesis.

2.2.1 Dynamic Optimisation Problems

Opposed to static OPs, the decision variables of dynamic OPs are elements of the Hilbert space. Thus, dynamic OPs deal with decision variables that are functions and no longer static parameters. Mostly, the decision variables are a function of time or a state variable like the arc-length of a reference line. The objective for dynamic OPs becomes a functional: a function of one or more functions. Calculus of variations enables the derivation of optimality conditions for such OPs.
Optimal control searches for the trajectory or sequence of system inputs that yields an optimal predefined objective. Dynamic systems are generally subject to input constraints and often further constraints that possibly depend on the inputs and states. Hence, the general dynamic OP for a system described by (2.16) is given by

$$\min_{\mathbf{u}(t)} \quad J(\mathbf{x}(t), \mathbf{u}(t), t_f) = \vartheta(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \phi(\mathbf{x}(t), \mathbf{u}(t), t) dt \tag{2.17a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} \quad \forall t \in [t_0, t_f]\,, \tag{2.17b}$$

$$\mathbf{c}(\mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} \quad \forall t \in [t_0, t_f]\,, \tag{2.17c}$$

$$\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{0} \quad \forall t \in [t_0, t_f]\,, \tag{2.17d}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.17e}$$

$$\mathbf{g}(\mathbf{x}(t_f), t_f) = \mathbf{0}. \tag{2.17f}$$

The cost functional J in (2.17a) to be minimised is comprised of the terminal cost function ϑ and the intermediate cost term ϕ. The equality constraints (2.17b) ensure that the system behaves according to the system equations (2.16). General equality constraints $\mathbf{c} \in \mathbb{R}^{n_c}$ and general inequality constraints $\mathbf{h} \in \mathbb{R}^{n_h}$ are included via (2.17c) and (2.17d), respectively. Furthermore, it is presumed that the constraint functions $\mathbf{c}$ and $\mathbf{h}$ are of class $\mathcal{C}^2$. The initial condition $\mathbf{x}_0$ and a general form of the final condition with $\mathbf{g} \in \mathbb{R}^{n_g}$ are represented by (2.17e) and (2.17f), respectively. For the subsequent optimality conditions, the functions ϑ, ϕ, $\mathbf{f}$, $\mathbf{h}$, $\mathbf{c}$ and $\mathbf{g}$ are assumed to be twice continuously differentiable.

2.2.2 Optimality Conditions of Dynamic Optimisation Problems

Similar to the Lagrangian function for static optimisation (2.7), the Hamiltonian function $\mathcal{H}$ and extended Hamiltonian function $\tilde{\mathcal{H}}$

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) = \phi(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}(t)^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{2.18a}$$

$$\tilde{\mathcal{H}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, t) = \mathcal{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) + \boldsymbol{\mu}(t)^T \mathbf{h}(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\Lambda}(t)^T \mathbf{c}(\mathbf{x}, \mathbf{u}, t) \tag{2.18b}$$

are used for deriving the optimality conditions of the dynamic OP. The Lagrange multipliers $\boldsymbol{\lambda}(t) \in \mathbb{R}^{n_x}$, $\boldsymbol{\mu}(t) \in \mathbb{R}^{n_h}$ and $\boldsymbol{\Lambda}(t) \in \mathbb{R}^{n_c}$ are also called adjoint variables in this context. Pontryagin's Minimum Principle [160] yields necessary conditions for optimality, which are listed below without the star-exponent for the sake of clarity [152, p.254]. The Hamiltonian dynamical system to be solved is described by

$$\dot{\mathbf{x}} = \nabla_{\boldsymbol{\lambda}} \tilde{\mathcal{H}} = \mathbf{f} \tag{2.19a}$$

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\mathbf{x}} \tilde{\mathcal{H}} = -\nabla_{\mathbf{x}} \phi - \mathbf{f}_{\mathbf{x}}^T \, \boldsymbol{\lambda} - \mathbf{h}_{\mathbf{x}}^T \, \boldsymbol{\mu} - \mathbf{c}_{\mathbf{x}}^T \, \boldsymbol{\Lambda} \tag{2.19b}$$

$$\nabla_{\mathbf{u}} \tilde{\mathcal{H}} = \nabla_{\mathbf{u}} \phi + \mathbf{f}_{\mathbf{u}}^T \, \boldsymbol{\lambda} + \mathbf{h}_{\mathbf{u}}^T \, \boldsymbol{\mu} + \mathbf{c}_{\mathbf{u}}^T \, \boldsymbol{\Lambda} = \mathbf{0} \tag{2.19c}$$

and must hold for $t \in [t_0, t_f]$. Additionally, the equality constraints as well as complementarity conditions must hold for $t \in [t_0, t_f]$:

$$\mathbf{c}\big(\mathbf{x}(t), \mathbf{u}(t), t\big) = \mathbf{0} \tag{2.19d}$$

$$\mu_i \, h_i = 0, \quad i = 1, \ldots, n_h \tag{2.19e}$$

$$\boldsymbol{\mu} \geq \mathbf{0} \tag{2.19f}$$

The second-order necessary condition is given by the Legendre condition, which requires an at least positive semi-definite Hessian of the extended Hamiltonian function

$$\nabla_{\mathbf{uu}}^2 \tilde{\mathcal{H}} \geq \mathbf{0} \quad \forall t \in [t_0, t_f] \quad \text{s.t.} \quad \mathcal{Y} = \{\delta \mathbf{u} | \mathbf{h}_{\mathbf{u}}^a \, \delta \mathbf{u} = \mathbf{0}; \mathbf{c}_{\mathbf{u}} \, \delta \mathbf{u} = \mathbf{0}\} \tag{2.19g}$$

with $\mathbf{h}^a$ representing the vector of active inequality constraints. Moreover, assuming the set of admissible controls

$$\mathcal{U}(\mathbf{x}, t) = \{\mathbf{u} | \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \leq \mathbf{0}; \mathbf{c}(\mathbf{x}, \mathbf{u}, t) = \mathbf{0}\}, \tag{2.19h}$$

the following equation is required to hold:

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) = \min_{\mathbf{u} \in \mathcal{U}(\mathbf{x}, t)} \mathcal{H}(\mathbf{x}, \mathbf{U}, \boldsymbol{\lambda}, t) \quad \forall t \in [t_0, t_f] \, . \tag{2.19i}$$

Besides the boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.19j}$$

$$\mathbf{g}\big(\mathbf{x}(t_f), t_f\big) = \mathbf{0}, \tag{2.19k}$$

conditions for the final values are dictated by the transversality conditions

$$\Big(\nabla_{\mathbf{x}(t_f)} \vartheta + \mathbf{g}_{\mathbf{x}(t_f)}^T \, \boldsymbol{\nu} - \boldsymbol{\lambda}\Big)^T \Big|_{t_f} \, \delta \mathbf{x}_f = 0 \tag{2.19l}$$

$$\Big(\mathcal{H}\Big(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t\Big) + \vartheta_{t_f} + \mathbf{g}_{t_f}^T \, \boldsymbol{\nu}\Big)\Big|_{t_f} \, \delta t_f = 0. \tag{2.19m}$$

The resulting equations (2.19) represent a two-point BVP. In practice, only simple BVPs can be solved analytically and the majority requires numerical approaches.
When trying to solve the above equations, it is possible that the input cannot be computed via (2.19c). This singular case can occur for instance when $\tilde{\mathcal{H}}$ is linear in $\mathbf{u}$. The solution to this problem is a special treatment [152, p.261] or the augmentation of the objective with a small regularisation term that is quadratic in $\mathbf{u}$ [97, 116, 137].

2.3 Solution Methods for Optimal Control Problems

Numerical methods for solving OCPs can be divided into three main groups: DP, IMs and DMs. A qualitative comparison of the general methods is given in Table 2.1. The individual methods are discussed in more detail below.

2.3.1 Dynamic Programming, Indirect Methods and Direct Methods

Dynamic programming is based on Bellman's principle of optimality [7], which states that the remaining part of an optimal trajectory from any intermediate point to the final point of the optimal trajectory represents an optimal trajectory itself [12]. This is exploited by dividing the OCP into many small OCPs that are solved by a recursive procedure moving backwards from the final point. Continuous DP requires the solution of a partial differential equation, called Hamilton-Jacobi-Bellman equation. However, a solution to this equation can be very rarely computed analytically. Thus, mainly discrete DP is applied, which yields an optimal control policy in dependence of the discrete state grid points. The resulting solution represents the global optimum, even if the OCP is nonconvex. However, this approach suffers from the well-known curse of dimensionality: Computational complexity and memory requirement grow exponentially with increasing number of states and inputs. Due to this limitation, classical DP is generally only applicable to systems of small order. Approximate dynamic programming (ADP) alleviates the dimensionality problem by introducing simplifications, like approximations of the value function or the control policies, yielding a suboptimal solution [161]. For systems that cannot be represented by an analytic model, Markov-Chains and probabilistic characteristics are used in stochastic DP. Another remedy is given by DDP, which computes a local solution using a quadratic Taylor expansion around a nominal trajectory [141]. A backward pass identifies optimal, state-dependent controller gains that are used in a subsequent forward pass to compute a new nominal trajectory via numerical integration. This procedure is repeated until convergence. For unconstrained OCPs, the solution can be computed analytically yielding a fast optimisation process in the backward pass. However, various methods have been introduced to enable the consideration of constraints [33, 202, 229]. Due to the dimensionality problem, classical DP is not

Table 2.1: Comparison of solution methods for optimal control problems.

	Dynamic Programming	**Indirect Methods**	**Direct Methods**
General idea	backwards recursion	BVP solution[1]	SOP transformation[2]
Optimality	global	local	local
Initialisation	easy	hard	easy/medium
Accuracy	medium/high	high	medium
Computing time	high	low	low

[1] boundary value problem. [2] static optimisation problem.

suitable for large-scale OCPs. ADP generally requires taking advantage of the problem structure, which supposes to have a good understanding of the underlying OCP [161, 162]. Another drawback is that its success depends on a careful parameter tuning. DDP often requires good initial nominal trajectories due to the successive approximation procedure and often only approximates constraints, e.g. by barrier methods. Thus, large-scale OCPs have been mainly solved using indirect and direct methods when aiming at accurate solutions and reasonable computation times [13, 29, 97].

Indirect methods require the analytical derivation of the necessary optimality conditions presented in Section 2.2.2. Mostly, the resulting BVP (2.19) cannot be solved analytically and requires a numerical solution of the discretised BVP-equations. In general, the analytical expressions for the optimality conditions yield a higher accuracy and more reliable error estimates than DMs provide [97]. However, initialising the adjoint variables can be a challenging process since they have no physical interpretation [13] and become more difficult with growing model complexity [29]. Moreover, the numerical solution of the adjoint equations (2.19b) can be difficult due to their ill-conditioning [13, p.129][154, p.72]. This results in a smaller domain of convergence requiring more accurate initialisations compared to DMs. Additionally, the presence of path inequality constraints requires an a priori estimate of the constrained-arc sequence [13, p.129].

Direct methods directly solve the OCP by transforming the infinite dimensional problem of finding optimal trajectories into a finite dimensional OP. This process called transcription is achieved by discretising the input and, if applicable, also state trajectories using decision variables at individual grid points of the independent variable. The resulting SOP can be solved with an adequate NLP solver. Since NLP solvers are designed to converge with poor initial guesses [166], DMs are very robust regarding the initialisation of the OCP. Furthermore, NLP solvers take advantage of the sparse structure of the OP, which reduces computational effort and memory usage [166].

Hybrid methods of DM and IM have been introduced to overcome the disadvantages of both approaches [154, 196]. A DM can be used to compute an initial solution since it is more robust regarding the initial guess. Then, the accuracy of this solution can be improved by feeding it as initial guess into an IM. However, this still requires the derivation of an initial guess for the adjoint variables.

2.3.2 Shooting and Collocation

Depending on the method for the satisfaction of differential equations, IMs and DMs are each further divided into three classes: single shooting (SS), multiple shooting (MS) or collocation [13, 166]. The properties of the individual approaches are discussed in the following paragraphs and are compared in Table 2.2.

Single shooting methods discretise the input trajectories by selecting the inputs at discrete points of the independent variable as decision variables of the OP. Values in between these points are approximated using polynomial interpolation. The interpolated input trajectories are used to numerically integrate the ordinary differential equations (ODEs) describing the system dynamics, which yields exact state trajectories in every time step. However, the numerical integration generates couplings between early control

Table 2.2: Comparison of shooting and collocation methods.

	Shooting[1]	**Collocation**
Inputs	discretised polynomials	discretised polynomials
States	via numerical integration	discretised polynomials
ODE-Accuracy[2]	exact on entire interval	exact at collocation points
OP-Size[3]	SS: low, MS: medium	high
Sparsity	SS: low, MS: medium	high
Robustness	SS: low, MS: medium	high

[1] SS: single shooting, MS: multiple shooting. [2] ordinary differential equation.
[3] optimisation problem.

inputs and later parts of the state trajectories, which can cause sensitivity problems: A small change in early controls can have a great impact on the entire solution [29]. In order to demonstrate this issue, let the behaviour of a single-state dynamical system with state x and input u be given by the initial value problem

$$\dot{x}(t) = f(x(t), u(t)) \quad \text{with} \quad x(t_0) = x_0. \tag{2.20}$$

The coupling becomes apparent when exemplarily considering numerical integration via the forward-euler method with fixed time step $t_\Delta \coloneqq t_{k+1} - t_k$, which is given by

$$x_{k+1} = x_k + t_\Delta \, f\left(x_k, u_k\right) \tag{2.21}$$

with index k indicating the corresponding time step, e.g. $x_k = x(t_k)$. Consider the first three time instants:

$$x_1 = x_0 + t_\Delta \, f\left(x_0, u_0\right) \tag{2.22a}$$

$$\begin{aligned} x_2 &= x_1 + t_\Delta \, f\left(x_1, u_1\right) \\ &= x_0 + t_\Delta \, f\left(x_0, u_0\right) + t_\Delta \, f\left(x_0 + t_\Delta \, f\left(x_0, u_0\right), u_1\right) \end{aligned} \tag{2.22b}$$

$$\begin{aligned} x_3 &= x_2 + t_\Delta \, f\left(x_2, u_2\right) \\ &= x_0 + t_\Delta \, f\left(x_0, u_0\right) + t_\Delta \, f\left(x_0 + t_\Delta \, f\left(x_0, u_0\right), u_1\right) + \\ &\quad t_\Delta \, f\left(x_0 + t_\Delta \, f\left(x_0, u_0\right) + t_\Delta \, f\left(x_0 + t_\Delta \, f\left(x_0, u_0\right), u_1\right), u_2\right). \end{aligned} \tag{2.22c}$$

Since the inputs at the individual time instants u_k represent decision variables, it becomes evident that the interconnection between the decision variables in the states increases over time. For nonlinear right-hand side functions f, this effect is particularly disadvantageous. These potentially highly nonlinear equations with strong couplings enter the OP via constraints and greatly deteriorate its solvability. The situation is even worse if nonlinear constraints have to be considered. Moreover, for unstable or ill-conditioned system dynamics, the numerical integration can diverge to enormous values especially for long integration intervals. Although there are numerical quadrature methods with improved stability properties, these methods generally require longer computation times.

Multiple shooting methods divide the time interval into subintervals on which the numerical integration is performed. The states at the interval margins represent additional decision variables and continuity constraints are introduced to ensure that the boundary points of the individual integration segments coincide. The shorter intervals improve the robustness of the numerical integration and reduce the couplings between inputs and states introduced by the numerical integration (2.22). Although the OP grows in size, the solvability of the OP can be enhanced due to increased sparsity. The numerical integration on the individual intervals can be performed in parallel to reduce computing time.
Collocation methods apply the discretisation and polynomial approximation for both the input and the state trajectories. The satisfaction of the system dynamics is achieved by introducing equality constraints at certain knot points, called collocation points. Thus, the problem increases in size and the exact fulfilment of the system dynamics is only guaranteed at these generally roughly spaced points. However, this procedure results in minimal couplings between input and state decision variables, leading to a far more sparse structure [29]. Considering example (2.22), the individual states x_k now also represent decision variables. Hence, the interconnection is reduced to decision variables that are locally together in terms of the current and preceding time instant. NLP solvers, like *IPOPT* [222], take advantage of this sparsity, reducing computational effort and memory usage [166]. Collocation methods differ in the order and type of approximating polynomials. Generally, high order polynomials exhibit a higher accuracy but usually lead to less sparsity [118]. Furthermore, the selection of the polynomial order as well as the number and position of the collocation points can be controlled by an adaptive scheme to further improve accuracy and computation time [155].

2.3.3 Model Predictive Control

The preceding sections presented various optimisation methods for the solution of OCPs. The main drawback of optimal control is the long computation time compared to other control strategies. The computational effort of OPs increases with the number of decision variables. If shooting methods are used, a shorter time horizon also yields simpler constraints with less couplings between the decision variables. This increased sparsity improves solvability of the OP. Furthermore, numerical integration is more stable for short integration intervals.
Using optimal control in real-time applications requires a sufficiently fast solution of OPs, which motivates the concept of MPC. MPC reduces computing time by considering a limited time horizon for the OP and successively shifting this time window. Mostly, receding horizon control is used, which looks a few time steps into the future but not in the past [61]. The horizon must be sufficiently long for an adequate system reaction and as short as possible to reduce computation time. Model uncertainties, environmental changes and disturbances cause the real system behaviour to differ from the modelled one. Closing the control loop can provide some reactivity so that these methods can be used in real-life applications. This is achieved by feeding back a measured or estimated state, which is used as initial value condition for the OCP at the current time instant.

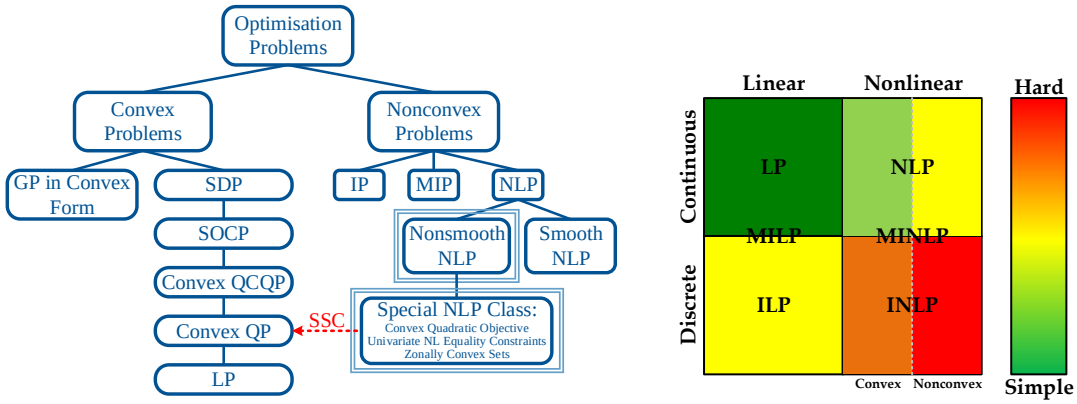
Generally, only the current value of the computed control trajectory is applied to the system and the remaining values are dismissed. When the next state measurement or estimate arrives, a new OCP is solved based on the updated value. However, considering only limited time horizons in the presence of constraints requires precautions to ensure recursive feasibility: The controller must avoid driving the state to a region where the OP does not have a solution [122]. Furthermore, the stability of the closed loop system must be ensured when applying MPC. Finite horizon MPC generally requires additional constraints and objective terms to ensure these features [106, 139, 165]. Most techniques make use of Lyapunov functions to ensure stability of the closed loop system. In contrast, passivity based MPC, which is also applicable to non-passive systems [58, 164], guarantees stability by introducing additional constraints ensuring passivity of the system to a fictitious output.
As mentioned, the computational complexity and thus solution time of OPs scales with the number of decision variables. However, the type of scaling depends on the utilised solution method as well as the class and structure of the OP [167]. A popular method that is often used to reduce the size of OCPs is condensing: Eliminating the state decision variables yields a dense OP of smaller size [17]. However, the growth of computational complexity with the horizon length is far slower for sparse problems than for dense problems. Thus, partial condensing can be used to find a trade-off between problem size and sparsity enhancing computation time [6].
Considering a moving, limited horizon can also be suitable for offline optimisation tasks when aiming at reducing computation time. In a racetrack setting, the solution is however only suboptimal since the decisions are based on a limited preview. Additionally, MPC requires an appropriate design of the horizon length and further precautions to avoid infeasibility and guarantee stability. Thus, this thesis does not consider MPC.

2.4 Numerical Optimisation

Numerical optimisation is a field in applied mathematics that uses numerical methods to solve mathematical OPs. A classification of OPs is given in Section 2.4.1 and popular approaches for the numerical solution of OPs are outlined in Section 2.4.2.

2.4.1 Classification of Optimisation Problems

OPs can be classified by various properties: convexity, linearity or type of nonlinearity, continuity of decision variables and smoothness. Fig. 2.2a provides an overview for the classification of OPs. Convexity represents arguably the most important property. If the OP is convex, any locally optimal solution of the problem is the global one. This reduces the probability of iterative methods getting stuck at suboptimal solutions. There are efficient and robust solvers for convex OPs capable of computing the global optimum with high accuracy and generally short computation times. Linearity represents a second decisive property of OPs. Generally, the more linear an OP is, the better it can be solved in terms of robustness and convergence speed. The type of nonlinearity determines whether the problem is convex and which type of convex OP is given, as depicted



(a) OP classes. Framed classes are considered in this thesis. **(b)** Computational complexity.

Figure 2.2: Classification of optimisation problems.

in Table 2.3. Nonconvex OPs can be further classified into NLP problems, IP problems and MIP problems. NLP problems contain only continuous decision variables and IP problems only discrete ones. However, MIP problems with continuous and discrete decision variables are more often encountered than OPs with solely discrete variables. In this thesis, NLP problems are also distinguished regarding their smoothness since there are many solvers that use derivative information to deduce appropriate descent directions.
The computational complexity of the individual OP classes is qualitatively assessed in Fig. 2.2b. Linear programming (LP) problems are inherently convex and represent one of the best solvable classes of OPs. Efficient algorithms have been developed to solve these problems in polynomial time [91]. Traversing into the domain of continuous but nonlinear OPs increases complexity. However, efficient solvers exist for convex NLPs. For instance, convex QP problems have been proven to be solvable in polynomial time [107]. Although nonconvex NLP problems are generally harder to solve, quite robust solvers have been developed that are capable of reliably solving these problems, albeit they generally require longer computation times. Although OPs with discrete variables are inherently nonconvex, they are referred to as convex, if dropping the integrality constraints results in a convex problem. Generally, the solution to integer linear programming (ILP) problems is identified by solving a sequence of efficiently solvable LP problems resulting in a moderate computational complexity. MINLP problems are generally hard to solve however there has been significant progress in this field enhancing the solvability of these problems, especially for convex MINLP problems [101, 110].
Due to its importance in operations research and since this thesis also deals with the convexification of OPs, convexity is discussed in more detail below. A set $\mathcal{S} \in \mathbb{R}^n$ is convex if a straight line connecting any two points within the set $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{S}$ with $\mathbf{w}_1 \neq \mathbf{w}_2$ is contained in the set [19, p.21]:

$$\alpha \mathbf{w}_1 + (1-\alpha)\mathbf{w}_2 \in \mathcal{S} \quad \forall \alpha \in [0,1]\,. \tag{2.23}$$

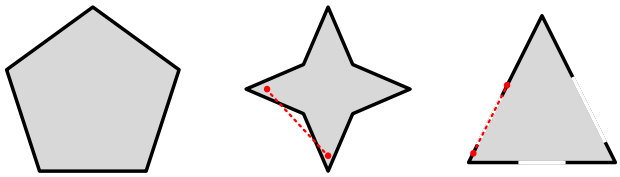


Figure 2.3: Examples for convex and nonconvex sets. The pentagon is like all polyhedra, which are defined by a set of affine equality and inequality constraints, convex [19, p.31]. The star shaped set is nonconvex since the depicted line segment between two admissible points is not contained in the set. The depicted triangle is nonconvex since some boundary points are not included in the set, thus the illustrated line leaves the set.

Examples of convex and nonconvex sets are depicted in Fig. 2.3. Similarly, a function $f : \mathcal{D} \to \mathbb{R}$ is convex if its domain $\mathcal{D}$ is a convex set and Jensen's inequality is satisfied for any two points $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{D}$ [19, p.77]:

$$f(\alpha \mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) \leq \alpha f(\mathbf{w}_1) + (1-\alpha) f(\mathbf{w}_2) \quad \forall \alpha \in [0,1]\,. \tag{2.24}$$

Furthermore, if f is twice differentiable, it is convex if its domain $\mathcal{D}$ is a convex set and its Hessian is positive (semi-)definite [19, p.71]:

$$\nabla^2_{\mathbf{ww}} f(\mathbf{w}) \geq 0 \quad \forall \mathbf{w} \in \mathcal{D}. \tag{2.25}$$

For a function with $\mathcal{D} = \mathbb{R}$, (2.25) reduces to $f''(w) \geq 0$ representing the requirement of a non-negative curvature. Convex sets and convex functions are linked via the epigraph: A function is only convex if its epigraph, which represents the set above the function, is a convex set [19, p.75]. The standard form of a convex OP [19] with decision variables $\mathbf{w} \in \mathbb{R}^{n_w}$ is given by

$$\min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad J(\mathbf{w}) \tag{2.26a}$$

$$\text{s.t.} \quad c_i(\mathbf{w}) \coloneqq \mathbf{a}_i^T \mathbf{w} + b_i = 0 \quad \forall i = 1, \ldots, n_c, \tag{2.26b}$$

$$h_j(\mathbf{w}) \leq 0 \quad \forall j = 1, \ldots, n_h \tag{2.26c}$$

with $\mathbf{a}_i \in \mathbb{R}^{n_w}$ and $b_i \in \mathbb{R}$. The n_c equality constraints (2.26b) are affine[3] thus linear in the decision variables. Furthermore, the cost function $J(\mathbf{w})$ as well as the n_h functions $h_j(\mathbf{w})$ defining the inequality constraints (2.26c) are convex in $\mathbf{w}$. The fact that only linear equality constraints are convex can be demonstrated as follows. An equality constraint $c(\mathbf{w}) = 0$ can be rewritten in terms of inequalities:

$$c(\mathbf{w}) \leq 0 \tag{2.27a}$$

$$c(\mathbf{w}) \geq 0. \tag{2.27b}$$

[3] Affine constraints are linear in the decision variables and can have additional constant terms.

Table 2.3: Classification of convex optimisation problems.

	objective	**equality constraints**	**inequality constraints**
LP	linear	linear	linear
cQP	convex quadratic	linear	linear
cQCQP	convex quadratic	linear	convex quadratic
SOCP	linear	linear	second-order cone
SDP	linear	linear	semi-definite cone
ncGP[1]	posynomial	monomial LHS[2]	posynomial LHS[2]

[1] nonconvex form of geometric program, which can be transformed into convex form.
[2] left-hand side of equation.

The equality constraint holds if both inequality constraints (2.27a) and (2.27b) are satisfied. If $c(\mathbf{w})$ is a convex function, (2.27a) represents a convex inequality constraint but (2.27b) is nonconvex. For a concave function $c(\mathbf{w})$ it is vice versa. Both inequality constraints can only be convex at the same time, if $c(\mathbf{w})$ is an affine function.
The type of objective and inequality constraints define the subclass of convex OP as depicted in Table 2.3. Following hierarchical structure is given for the subclasses of convex OPs:

$$\text{LP} \subset \text{cQP} \subset \text{cQCQP} \subset \text{SOCP} \subset \text{SDP}. \tag{2.28}$$

An LP problem is comprised of an affine objective function as well as affine equality and inequality constraints

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad & \mathbf{q}_0^T \mathbf{w} + r_0 && \text{(2.29a)} \\ \text{s.t.} \quad & \mathbf{A}\,\mathbf{w} + \mathbf{b} = \mathbf{0}, && \text{(2.29b)} \\ & \mathbf{G}\,\mathbf{w} + \mathbf{k} \leq \mathbf{0} && \text{(2.29c)} \end{aligned}$$

with $\mathbf{q}_0 \in \mathbb{R}^{n_w}$, $r_0 \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{n_c \times n_w}$, $\mathbf{b} \in \mathbb{R}^{n_c}$, $\mathbf{G} \in \mathbb{R}^{n_h \times n_w}$ and $\mathbf{k} \in \mathbb{R}^{n_h}$ [19, p.146]. A convex QP problem is given in case of affine constraints and a convex quadratic objective function

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad & \frac{1}{2}\mathbf{w}^T \mathbf{P}_0\,\mathbf{w} + \mathbf{q}_0^T \mathbf{w} + r_0 && \text{(2.30a)} \\ \text{s.t.} \quad & \mathbf{A}\,\mathbf{w} + \mathbf{b} = \mathbf{0}, && \text{(2.30b)} \\ & \mathbf{G}\,\mathbf{w} + \mathbf{k} \leq \mathbf{0} && \text{(2.30c)} \end{aligned}$$

with matrix $\mathbf{P}_0 \in \mathbb{R}^{n_w \times n_w}$ being positive (semi-)definite $\mathbf{P}_0 \geq 0$. A convex QCQP problem arises if also convex quadratic inequality constraints are present

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad & \frac{1}{2}\mathbf{w}^T \mathbf{P}_0\,\mathbf{w} + \mathbf{q}_0^T \mathbf{w} + r_0 && \text{(2.31a)} \\ \text{s.t.} \quad & \mathbf{A}\,\mathbf{w} + \mathbf{b} = \mathbf{0}, && \text{(2.31b)} \\ & \frac{1}{2}\mathbf{w}^T \mathbf{P}_j\,\mathbf{w} + \mathbf{q}_j^T \mathbf{w} + r_j \leq 0 \quad \forall j = 1, \ldots, n_h && \text{(2.31c)} \end{aligned}$$

with $r_j \in \mathbb{R}$, $\mathbf{q}_j \in \mathbb{R}^{n_w}$ and positive (semi-)definite matrices $0 \leq \mathbf{P}_0, \mathbf{P}_j \in \mathbb{R}^{n_w \times n_w}$ [19, p.152]. SOCP problems represent a further generalisation with the inequality constraints representing a second-order cone

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad & \mathbf{q}_0^T \mathbf{w} && \text{(2.32a)} \\ \text{s.t.} \quad & \mathbf{A}\,\mathbf{w} + \mathbf{b} = \mathbf{0}, && \text{(2.32b)} \\ & \|\mathbf{G}_j \mathbf{w} + \mathbf{k}_j\|_2 \leq \mathbf{d}_j^T \mathbf{w} + l_j \quad \forall j = 1, \ldots, n_h && \text{(2.32c)} \end{aligned}$$

with $\mathbf{G}_j \in \mathbb{R}^{n_h \times n_w}$, $\mathbf{k}_j \in \mathbb{R}^{n_h}$, $\mathbf{d}_j \in \mathbb{R}^{n_w}$ and $l_j \in \mathbb{R}$ [19, p.156]. More general positive semi-definite cone inequality constraints yield a semi-definite programming (SDP) problem, which is defined as

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad & \mathbf{q}_0^T \mathbf{w} && \text{(2.33a)} \\ \text{s.t.} \quad & \mathbf{A}\,\mathbf{w} + \mathbf{b} = \mathbf{0}, && \text{(2.33b)} \\ & \mathbf{F}(\mathbf{w}) \coloneqq \mathbf{F}_0 + \sum_{j=1}^{n_w} w_j \mathbf{F}_j \geq \mathbf{0} && \text{(2.33c)} \end{aligned}$$

with symmetric matrices $\mathbf{F}_0, \mathbf{F}_j \in \mathbb{R}^{n_h \times n_h}$ and positive semi-definite matrix $\mathbf{F}(\mathbf{w}) \geq 0$ [216]. Geometric programming (GP) problems represent a special class of nonconvex OPs defined as

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}_{>0}} \quad & p_0(\mathbf{w}) && \text{(2.34a)} \\ \text{s.t.} \quad & m_i(\mathbf{w}) = 1 \quad \forall i = 1, \ldots, n_c, && \text{(2.34b)} \\ & p_j(\mathbf{w}) \leq 1 \quad \forall j = 1, \ldots, n_h && \text{(2.34c)} \end{aligned}$$

with m_i in equality constraints (2.34b) being monomials and p_0 in the objective (2.34a) as well as p_j in the inequality constraints (2.34c) being posynomials [19, p.161]. The decision variables are limited to positive, real values. With $\mathbf{w} \in \mathbb{R}^{n_w}_{>0}$, a monomial and posynomial is given by

$$m(\mathbf{w}) = g \cdot \left(w_1^{e_1} \cdot \ldots \cdot w_{n_w}^{e_{n_w}}\right) \text{ with } g > 0,\ \mathbf{e} = \begin{bmatrix} e_1 & \ldots & e_{n_w} \end{bmatrix} \in \mathbb{R}^{n_w} \quad \text{and} \tag{2.35}$$

$$p(\mathbf{w}) = \sum_{k=1}^{K} g_k \cdot \left(w_1^{e_{k,1}} \cdot \ldots \cdot w_{n_w}^{e_{k,n_w}}\right) \text{ with } g_k > 0,\ \mathbf{e}_k = \begin{bmatrix} e_{k,1} & \ldots & e_{k,n_w} \end{bmatrix} \in \mathbb{R}^{n_w}, \tag{2.36}$$

respectively. Thus, a posynomial represents a sum of monomials. Although (2.34) is nonconvex, it can be cast into its convex form by considering the transformation $\tilde{w}_l = \log(w_l)\ \forall l = 1, \ldots, n_w$. The resulting convex OP is given in [19, p.162].
This thesis deals with the solution of nonsmooth NLP problems. Many engineering problems require nonconvex constraints or a nonlinear, potentially nonsmooth, differential equation system to describe the system behaviour. An optimal control approach for solving nonsmooth, nonconvex NLP problems is presented in Chapter 3. This procedure enables the computation of optimal solutions for many engineering problems. The

approach aims at solving such problems with high accuracy but requires longer computation times. An alternative approach that aims at low computation times is presented in Chapter 5. The SSC method greatly reduces computation time by iteratively solving convex QP subproblems of the original nonconvex OP. The applicability is however limited to a certain class of nonconvex problems: nonconvex OPs that possess a convex quadratic objective, equality constraints that are affine or exhibit only univariate nonlinearities and inequality constraints depicting convex sets or ZCSs.

2.4.2 Numerical Solution of Optimisation Problems

This thesis employs numerical solvers using derivative information to solve OPs. Starting with an initial guess, derivative information is used to compute a descent direction for iterative improvement of the solution. Focussing on Newton-type optimisation methods, solutions to the KKT equations (2.9) are computed by iterating on the triple $\mathbf{p}^\star = (\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ until reaching a solution of sufficient accuracy. There are two major Newton-type approaches, which differ in the treatment of the complementarity conditions (2.9c)-(2.9e): SQP and IPMs. Both methods are presented and compared in this section.

Newton's Method Newton's method can be used to solve nonlinear equation systems or minimisation problems. In order to numerically compute the roots of a differentiable nonlinear, vector-valued equation system $\mathbf{F}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$, a first-order Taylor expansion (B.3c) is used to approximate the equation system at a point $\mathbf{x}_k$ close to the actual zero $\mathbf{x}^\star$ and the estimated function is set to zero:

$$\mathbf{F}(\mathbf{x}) \approx \mathbf{F}(\mathbf{x}_k) + \mathbf{F}_\mathbf{x}(\mathbf{x}_k)\,\Delta\mathbf{x}_k = \mathbf{0}. \tag{2.37}$$

Solving for the direction $\Delta\mathbf{x}_k = \mathbf{x} - \mathbf{x}_k$ enables formulating a policy for successively computing better approximations of the root:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{F}_\mathbf{x}(\mathbf{x}_k)^{-1}\,\mathbf{F}(\mathbf{x}_k). \tag{2.38}$$

Newton's method can also be employed to compute stationary points of a scalar-valued objective function $J(\mathbf{w})$ used in OPs. Inserting $\mathbf{F}(\mathbf{x}) = \nabla_\mathbf{w} J(\mathbf{w})$ into (2.38) yields an equation for iteratively computing the roots of the cost function gradient

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \nabla^2_{\mathbf{ww}} J(\mathbf{w}_k)^{-1}\,\nabla_\mathbf{w} J(\mathbf{w}_k), \tag{2.39}$$

that corresponds to applying a second-order Taylor expansion of J around $\mathbf{w}_k$ [103, p.87]. The prerequisite for using equation (2.39) is a positive definite Hessian $\nabla^2_{\mathbf{ww}} J(\mathbf{w}_k)$, which has to be computed in every iteration step. Due to the quadratic convergence in a close neighbourhood of the root, Newton's method tends to converge faster compared to first-order approaches like the gradient descent method, which exhibits only linear convergence. Since Newton's method suffers from a small area of convergence around the local optimum, it can be combined with a line-search algorithm [148, p.56] or a

trust-region approach [148, p.92] to yield global convergence under certain conditions. Furthermore, Quasi-Newton methods reduce the computational effort by using a positive definite approximation of the Hessian to compute the search direction for the next iterate [148, p.46]. Moreover, there are various algorithms for the modification of an indefinite Hessian yielding a positive definite approximation [148, p.48].

Sequential Quadratic Programming In this paragraph, active-set SQP methods are considered. An equality-constrained QP approach is assumed, which converts the inequality-constrained OP (2.1) into equality-constrained subproblems [148, p.533]. By iteratively guessing which inequality constraint will hold with equality at the optimal solution, a subset called working set is formed in every iteration. Only the active inequalities, denoted in the following by $\mathbf{h}^a$, are considered as equality constraints in the modified OP

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{n_w}} \quad & J(\mathbf{w}) && \text{(2.40a)} \\ \text{s.t.} \quad & \mathbf{c}(\mathbf{w}) = \mathbf{0}, && \text{(2.40b)} \\ & \mathbf{h}^a(\mathbf{w}) = \mathbf{0}. && \text{(2.40c)} \end{aligned}$$

In case of an incorrect guess, the working set is updated by removing or adding inequality constraints until the optimal solution is identified. Thus, the constrained OP is solved by wandering along the border of the admissible set until the optimum is reached [152, p.117]. The Lagrangian function belonging to the modified OP (2.40) is given by

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J(\mathbf{w}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{w}) + \boldsymbol{\mu}^T \mathbf{h}^a(\mathbf{w}) \tag{2.41}$$

with the corresponding KKT equations

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) &= \nabla_{\mathbf{w}} J(\mathbf{w}^\star) + \mathbf{c}_{\mathbf{w}}(\mathbf{w}^\star)^T \boldsymbol{\lambda}^\star + \mathbf{h}^a_{\mathbf{w}}(\mathbf{w}^\star)^T \boldsymbol{\mu}^\star = \mathbf{0} && \text{(2.42a)} \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) &= \mathbf{c}(\mathbf{w}^\star) = \mathbf{0} && \text{(2.42b)} \\ \nabla_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) &= \mathbf{h}^a(\mathbf{w}^\star) = \mathbf{0}. && \text{(2.42c)} \end{aligned}$$

The nonlinear equation system (2.42) can be solved via Newton's method using the linearised equation system

$$\underbrace{\begin{bmatrix} \nabla_{\mathbf{w}} J|_{\mathbf{w}_k} + \mathbf{c}_{\mathbf{w}}^T|_{\mathbf{w}_k} \boldsymbol{\lambda}_k + \mathbf{h}_{\mathbf{w}}^{a^T}|_{\mathbf{w}_k} \boldsymbol{\mu}_k \\ \mathbf{c}|_{\mathbf{w}_k} \\ \mathbf{h}^a|_{\mathbf{w}_k} \end{bmatrix}}_{=\mathbf{F}(\mathbf{p}_k)} + \underbrace{\begin{bmatrix} \nabla^2_{\mathbf{ww}} \mathcal{L}|_{\mathbf{p}_k} & \mathbf{c}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{h}^a_{\mathbf{w}}|_{\mathbf{w}_k} \\ \mathbf{c}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{0} & \mathbf{0} \\ \mathbf{h}^a_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{=\mathbf{F}_{\mathbf{p}}(\mathbf{p}_k)} \underbrace{\begin{bmatrix} \Delta\mathbf{w}_k \\ \Delta\boldsymbol{\lambda}_k \\ \Delta\boldsymbol{\mu}_k \end{bmatrix}}_{=\Delta\mathbf{p}_k} = \mathbf{0} \tag{2.43}$$

at the current iterate $\mathbf{p}_k = (\mathbf{w}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$. Solving equation (2.43) using the update policy (2.38) provides the search direction $\Delta\mathbf{p}_k$ for the computation of the subsequent iterate

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \Delta\mathbf{w}_k, \quad \boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \Delta\boldsymbol{\lambda}_k, \quad \boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \Delta\boldsymbol{\mu}_k \tag{2.44}$$

which enables the reformulation of (2.43) resulting in

$$\begin{bmatrix} \nabla_{\mathbf{w}} J|_{\mathbf{w}_k} \\ \mathbf{c}|_{\mathbf{w}_k} \\ \mathbf{h}^a|_{\mathbf{w}_k} \end{bmatrix} + \begin{bmatrix} \nabla^2_{\mathbf{ww}} \mathcal{L}|_{\mathbf{p}_k} & \mathbf{c}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{h}^a_{\mathbf{w}}|_{\mathbf{w}_k} \\ \mathbf{c}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{0} & \mathbf{0} \\ \mathbf{h}^a_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{w}_k \\ \boldsymbol{\lambda}_{k+1} \\ \boldsymbol{\mu}_{k+1} \end{bmatrix} = \mathbf{0}. \tag{2.45}$$

Equation (2.45) can be interpreted as first-order optimality condition of the following QP problem [148, p.531][152, p.114]:

$$\min_{\Delta \mathbf{w}_k \in \mathbb{R}^{n_w}} \quad \frac{1}{2} \Delta \mathbf{w}_k^T \nabla^2_{\mathbf{ww}} \mathcal{L}(\mathbf{w}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k) \, \Delta \mathbf{w}_k + \nabla_{\mathbf{w}} J(\mathbf{w}_k)^T \, \Delta \mathbf{w}_k \tag{2.46a}$$

$$\text{s.t.} \quad \nabla_{\mathbf{w}} \mathbf{c}(\mathbf{w}_k)^T \, \Delta \mathbf{w}_k + \mathbf{c}(\mathbf{w}_k) = \mathbf{0}, \tag{2.46b}$$

$$\nabla_{\mathbf{w}} \mathbf{h}^a(\mathbf{w}_k)^T \, \Delta \mathbf{w}_k + \mathbf{h}^a(\mathbf{w}_k) = \mathbf{0}. \tag{2.46c}$$

Thus, the application of SQP methods corresponds to a successive approximation of the problem (2.40) around the current iterate $\mathbf{p}_k = (\mathbf{w}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$ using Taylor series expansion. The objective is quadratically approximated and the constraints linearly yielding a QP problem.

Interior-Point Method Opposed to active-set methods, IPMs reach the optimum from within the admissible set by strictly satisfying inequality constraints using a barrier parameter to control the distance to the border of the admissible set [152, p.117]. This procedure avoids the combinatorial complexity of active-set methods, which is especially advantageous for large-scale OPs [152, p.118]. Converting inequality constraints into equality constraints using slack variables $\mathbf{s} \in \mathbb{R}^{n_h}$ yields the OP

$$\min_{\mathbf{w} \in \mathbb{R}^{n_w}, \mathbf{s} \in \mathbb{R}^{n_h}} \quad J(\mathbf{w}) \tag{2.47a}$$

$$\text{s.t.} \quad \mathbf{c}(\mathbf{w}) = \mathbf{0}, \tag{2.47b}$$

$$\mathbf{h}(\mathbf{w}) + \mathbf{s} = \mathbf{0}, \tag{2.47c}$$

$$\mathbf{s} \geq \mathbf{0} \tag{2.47d}$$

with the Lagrangian function

$$\mathcal{L}(\mathbf{w}, \mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J(\mathbf{w}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{w}) + \boldsymbol{\mu}^T \left(\mathbf{h}(\mathbf{w}) + \mathbf{s} \right) - \boldsymbol{\nu}^T \mathbf{s}. \tag{2.48}$$

Relaxing the complementarity conditions of the corresponding KKT equations in (2.9) via a barrier parameter[4] τ results in

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^\star, \mathbf{s}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \nabla_{\mathbf{w}} J(\mathbf{w}^\star) + \mathbf{c}_{\mathbf{w}}(\mathbf{w}^\star)^T \boldsymbol{\lambda}^\star + \mathbf{h}_{\mathbf{w}}(\mathbf{w}^\star)^T \boldsymbol{\mu}^\star = \mathbf{0} \tag{2.49a}$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{w}^\star, \mathbf{s}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \mathbf{c}(\mathbf{w}^\star) = \mathbf{0} \tag{2.49b}$$

$$\nabla_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{w}^\star, \mathbf{s}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \mathbf{h}(\mathbf{w}^\star) + \mathbf{s}^\star = \mathbf{0} \tag{2.49c}$$

$$\mu_j^\star s_j^\star = \tau \quad \forall j = 1, \ldots, n_h \tag{2.49d}$$

$$\mathbf{s}^\star \geq 0, \ \boldsymbol{\mu}^\star \geq 0 \tag{2.49e}$$

[4] The conditions in (2.49) can also be derived using the concept of barrier methods with the barrier function $\Phi(\mathbf{w}, \tau) = J(\mathbf{w}) - \tau \sum_{j=1}^{n_h} \ln(-h_j(\mathbf{w}))$, hence the term barrier parameter [152, p.106, 118].

with $\nabla_{\mathbf{s}}\mathcal{L} = \mathbf{0}$ dictating $\boldsymbol{\nu} = \boldsymbol{\mu}$. Using $\mathbf{S} = \operatorname{diag}(s_1, \dots, s_{n_h})$, $\mathbf{M} = \operatorname{diag}(\mu_1, \dots, \mu_{n_h})$ and $\mathbf{e} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^{n_h}$, the nonlinear equation system given by (2.49) is solved via Newton's method using the so-called primal-dual system

$$\underbrace{\begin{bmatrix} \nabla_{\mathbf{w}} J|_{\mathbf{w}_k} + \mathbf{c}_{\mathbf{w}}^T|_{\mathbf{w}_k} \boldsymbol{\lambda}_k + \mathbf{h}_{\mathbf{w}}^T|_{\mathbf{w}_k} \boldsymbol{\mu}_k \\ \mathbf{S}|_{\mathbf{s}_k} \boldsymbol{\mu}_k - \tau \, \mathbf{e} \\ \mathbf{c}|_{\mathbf{w}_k} \\ \mathbf{h}|_{\mathbf{w}_k} + \mathbf{s}_k \end{bmatrix}}_{=\mathbf{F}(\mathbf{p}_k)} + \underbrace{\begin{bmatrix} \nabla_{\mathbf{ww}}^2 \mathcal{L}|_{\mathbf{p}_k} & \mathbf{0} & \mathbf{c}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{h}_{\mathbf{w}}|_{\mathbf{w}_k} \\ \mathbf{0} & \mathbf{M}|_{\boldsymbol{\mu}_k} & \mathbf{0} & \mathbf{S}|_{\mathbf{s}_k} \\ \mathbf{c}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{h}_{\mathbf{w}}|_{\mathbf{w}_k} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{=\mathbf{F}_{\mathbf{p}}(\mathbf{p}_k)} \underbrace{\begin{bmatrix} \Delta \mathbf{w}_k \\ \Delta \mathbf{s}_k \\ \Delta \boldsymbol{\lambda}_k \\ \Delta \boldsymbol{\mu}_k \end{bmatrix}}_{=\Delta \mathbf{p}_k} = \mathbf{0} \tag{2.50}$$

at the current iterate $\mathbf{p}_k = (\mathbf{w}_k, \mathbf{s}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$. Solving equation system (2.50) via the update policy (2.38) yields the search directions $\Delta \mathbf{p}_k$ at the current iterate for the primal variables $\mathbf{w}, \mathbf{s}$ and dual variables $\boldsymbol{\lambda}, \boldsymbol{\mu}$. The solution is used to compute the subsequent iterate according to

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_{p,k} \, \Delta \mathbf{w}_k, \quad \mathbf{s}_{k+1} = \mathbf{s}_k + \alpha_{p,k} \, \Delta \mathbf{s}_k \tag{2.51a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_{d,k} \, \Delta \boldsymbol{\lambda}_k, \quad \boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \alpha_{d,k} \, \Delta \boldsymbol{\mu}_k \tag{2.51b}$$

whereas the current step widths $\alpha_{p,k}$ and $\alpha_{d,k}$ are chosen such that the boundary of inequalities (2.49e) is not reached too fast [152, p.117]. For vanishing barrier parameter $\tau \to 0$, the solution of the relaxed equation system (2.49) converges to the minimiser of the original problem. Thus, the barrier parameter τ is successively reduced. The algorithm stops when a termination condition of the form

$$\max\left\{ \left\| \nabla_{\mathbf{w}} J|_{\mathbf{w}_k} + \mathbf{c}_{\mathbf{w}}^T|_{\mathbf{w}_k} \boldsymbol{\lambda}_k + \mathbf{h}_{\mathbf{w}}^T|_{\mathbf{w}_k} \boldsymbol{\mu}_k \right\|, \left\| \mathbf{S}|_{\mathbf{s}_k} \boldsymbol{\mu}_k - \tau \, \mathbf{e} \right\|, \left\| \mathbf{c}|_{\mathbf{w}_k} \right\|, \left\| \mathbf{h}|_{\mathbf{w}_k} + \mathbf{s}_k \right\| \right\} \leq \varepsilon \tag{2.52}$$

is satisfied. The convergence properties of IPMs greatly depend on the adaptation strategy for the barrier parameter τ with several methods existing in literature [152, p.120]. For a deeper analysis of individual IPMs, the reader is referred to [63].

Comparison of Newton Methods and Selected Solvers IPMs reach the optimum from within the admissible set by using slack variables to describe the distance to the set boundary. Thus, opposed to the active-set strategy of SQP methods, there is no need figuring out the active inequality constraints, which represents a problem of high combinatorial complexity. Due to their constant computational load when solving instances of related problems, IPM are often preferred for solving QP problems [61]. Another benefit are polynomial runtime guarantees for various IPMs [9, 146]. As a result, IPMs tend to have a faster convergence rate and are well suited for solving large-scale, sparse problems [148, p.564]. Opposed to IPM [86], SQP methods greatly benefit from warmstarting: Reusing the solution from the previous time instant may greatly reduce computational load [61, 167]. For the real-time iteration scheme for nonlinear MPC [48] introduced in Section 1.2, SQP methods provide a better tangential predictor than IPMs: The predictor delivered by SQP methods is also accurate across active-set changes [49, 61]. Thus, SQP methods have been used very successfully for real-time capable MPC.

In this thesis, the entire horizon of the OPs is considered. Thus, the beneficial warmstarting properties of SQP methods, which are especially useful within an MPC framework, do not come into play. In the first part of this thesis, *IPOPT* is employed for the solution of large-scale, sparse NLP problems. This solver uses a primal-dual IPM employing a filter line-search, heuristics and further correction measures for performance improvement and robustness [222]. The second part of the thesis deals with convex programming problems, which are solved via *Gurobi* [74]. By default, this optimiser uses an IPM to solve convex QP problems and a concurrent optimiser combining an IPM with a simplex method for solving LP problems. Although initially developed for LP problems, it is possible to apply the simplex method also to convex QP problems [228]. Detailed information on the simplex algorithm is given in [148, pp.355-392].

3 Nonlinear Programming for Nonconvex Minimum-Time Optimal Control

The first goal of this thesis is solving complicated OCPs focussing on accuracy rather than computation time. Thus, this chapter presents a framework using nonlinear programming to solve a broad class of nonconvex OCPs. Direct Hermite-Simpson collocation is used to convert the dynamic OP into a static one. The framework is implemented using the programming language *Julia* [14] with the embedded modelling language *JuMP* [52] for mathematical optimisation. The necessary derivative information is computed symbolically in advance by a forward mode automatic differentiation algorithm [169] included in *JuMP*. Thus, the gradients and Hessians are available in machine precision hence high accuracy. *IPOPT* is selected for the solution of the nonconvex OP. This chapter is based on our publications [185] and [183].
The class of OCPs for which the framework is applicable is presented in Section 3.1. In order to simplify the OCP and improve convergence rate, several preliminaries are implemented. Approximations for a smooth OP are presented in Section 3.2. The differential equations are reformulated in Section 3.3 to employ spatial information as independent variable. A scaling procedure normalising the OP is presented in Section 3.4. The reformulated dynamic OP is then transcribed into a static one. For this purpose, Section 3.5 elaborates the transcription process using Hermite-Simpson collocation. The resulting SOP is presented in Section 3.6. Section 3.7 concludes the chapter by comparing the approach with other existing methods, concentrating on lap time optimisation.

3.1 Specification of Optimal Control Problem

Aiming at solving a large class of nonconvex OCPs, the framework considers systems that can be characterised by a continuous, input nonaffine, time-invariant differential equation system

$$\dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}\big) \tag{3.1a}$$

with state vector $\mathbf{x} \in \mathbb{R}^{n_x}$, input vector $\mathbf{u} \in \mathbb{R}^{n_u}$ and the adjustable but time-invariant model parameters concatenated in the vector $\mathbf{p} \in \mathbb{R}^{n_p}$. For simplicity, static model parameters are omitted in (3.1a). Since collocation will be employed, the trajectories $\mathbf{x}(t)$ and $\mathbf{u}(t)$ will be represented by decision variables. Additional time-invariant decision variables or optimisation parameters will be used for $\mathbf{p}$ to determine optimal model parameters. For the moment, the system equations are not required to be continuously differentiable[5], thus $\mathbf{f} \in \mathcal{C}^0$ is allowed. It is assumed that the states, inputs and

[5] Further remarks on continuity and smoothness are given in Appendix B.2.

parameters are bounded by box constraints according to

$$\mathbf{x}(t) \in \mathcal{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}], \tag{3.1b}$$
$$\mathbf{u}(t) \in \mathcal{U} = [\underline{\mathbf{u}}, \overline{\mathbf{u}}] \quad \text{and} \tag{3.1c}$$
$$\mathbf{p} \in \mathcal{P} = \left[\underline{\mathbf{p}}, \overline{\mathbf{p}}\right]. \tag{3.1d}$$

Furthermore, the OCP can be subject to general equality and inequality constraints

$$\mathbf{c}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) = \mathbf{0} \quad \text{and} \tag{3.1e}$$
$$\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) \leq \mathbf{0}, \tag{3.1f}$$

respectively. Both constraint functions $\mathbf{c}$ and $\mathbf{h}$ can be nonsmooth thus of class $\mathcal{C}^0$. The goal of the OCP is minimising the objective

$$J(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}), \tag{3.1g}$$

which will be twice continuously differentiable due to the contemplated problems within this thesis. As previously mentioned, solvers using derivative information require the OP to be sufficiently continuously differentiable. Thus, the next section discusses how discontinuities in $\mathbf{f}$, $\mathbf{c}$ and $\mathbf{h}$ can be approximated, enabling the use of such algorithms.

3.2 Smoothing of Optimal Control Problem

The applied solver *IPOPT* uses an elaborate IPM to compute the optimal solution. Considering the corresponding equation system (2.50) that is solved for IPMs, discontinuities need to be smoothed in order to guarantee twice continuous differentiability of the functions $\mathbf{f}$, $\mathbf{h}$, $\mathbf{c}$ and J. Assuming $0 < \varepsilon \ll 1$ and $c \gg 1$, following smooth approximation is adopted for absolute value functions:

$$\operatorname{abs}(x) = |x| \approx \sqrt{x^2 + \varepsilon} =: \operatorname{abs}_\varepsilon(x). \tag{3.2}$$

Applying (3.2) to the maximum and minimum function results in

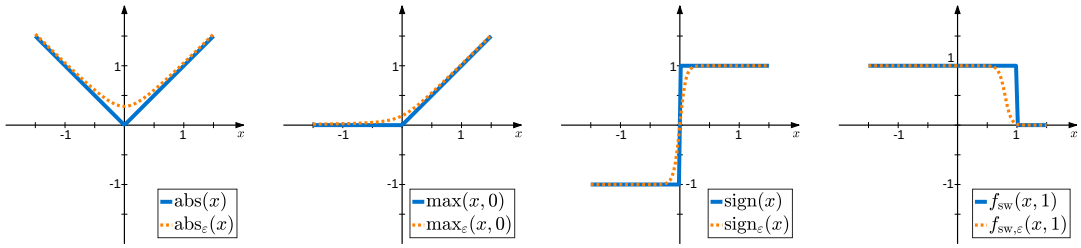
$$\max(x, y) = \frac{1}{2}\Big[(x + y) + |x - y|\Big] \approx \frac{1}{2}\left[(x + y) + \sqrt{(x - y)^2 + \varepsilon}\right] =: \max_\varepsilon(x, y) \tag{3.3a}$$
$$\min(x, y) = \frac{1}{2}\Big[(x + y) - |x - y|\Big] \approx \frac{1}{2}\left[(x + y) - \sqrt{(x - y)^2 + \varepsilon}\right] =: \min_\varepsilon(x, y), \tag{3.3b}$$

respectively [158]. Furthermore, the sign-function can be approximated by

$$\operatorname{sign}(x) \approx \tanh(c\,x) = \tanh\left(\frac{x}{\varepsilon}\right) =: \operatorname{sign}_\varepsilon(x). \tag{3.4}$$

This can be used to represent a smooth switching function with switching point x_{sw}:

$$\begin{aligned} f_{\mathrm{sw}}(x, x_{\mathrm{sw}}) &= -\frac{1}{2}\Big[\operatorname{sign}(x - x_{\mathrm{sw}}) - 1\Big] \\ &\approx -\frac{1}{2}\Big[\tanh\Big(c\,(x - x_{\mathrm{sw}} + \varepsilon_x)\Big) - 1\Big] =: f_{\mathrm{sw},\varepsilon}(x, x_{\mathrm{sw}}). \end{aligned} \tag{3.5}$$



(a) Absolute value function. **(b)** Maximum function. **(c)** Sign function. **(d)** Switching function.

Figure 3.1: Smooth versions of discontinuous functions.

Therein a small shift variable $0 < \varepsilon_x \ll 1$ is introduced to avoid violating the switching point and approximate the boundary from below. The original functions with their corresponding smooth approximations are illustrated in Fig. 3.1. After implementing these smoothing measures, the smoothed differential equations will be reformulated in the subsequent Section 3.3 in order to transition to spatial information as independent variable. This transformation is especially useful for minimum-time problems with spatially dependent path constraints.

3.3 Reformulation of Differential Equations

The system behaviour is described by the differential equation system (3.1a) with time as independent variable. For minimum-time optimal control applications, it can be advantageous to reformulate the system equations to depend on spatial information rather than time. When the final position is fixed, this approach transforms the OCP with free final time to a problem with fixed final value for the independent variable [158, 173]. The time information is then decoupled from the path geometry and inputs and states are explicitly connected to the spatial variable. Furthermore, some geometric path limitations, such as track boundaries and obstacles, can then be formulated via simple box constraints. For industrial robots, this procedure is illustrated in [219, p.104]. The approach is adopted for planar point-to-point motions of vehicular systems, which is subsequently presented.

3.3.1 Curvilinear Coordinates

The system motion can be defined in dependence of a spatial variable like the arc length $s_\mathcal{R}$ of a predefined reference curve $\mathcal{R}$, as illustrated in Fig. 3.2. This reference line can be characterised by its curvature $\kappa_\mathcal{R}$ over its arc length. Two coordinate systems are introduced for further considerations. The inertial frame with coordinates (x^I, y^I, z^I) represents a static world coordinate system. The body frame with coordinates (x^B, y^B, z^B) is linked to the COG of the moving object capturing the position and orientation of the object. The yaw angle ψ represents the rotation of the body coordinate frame around

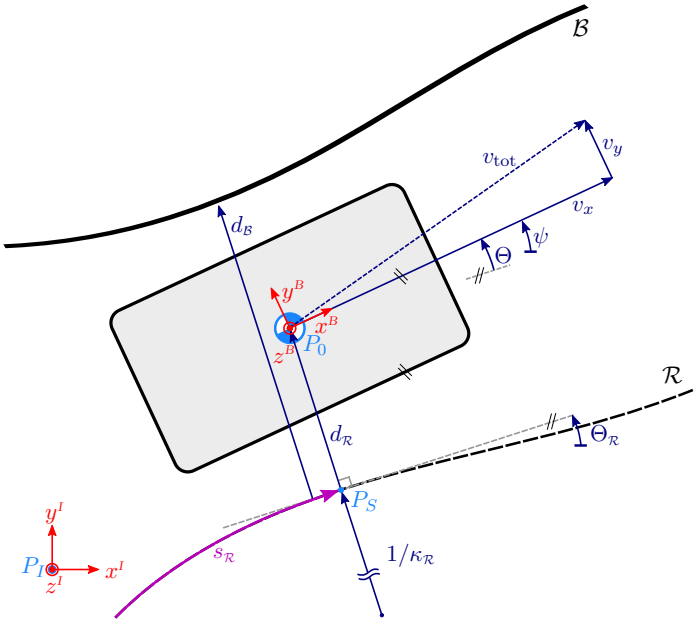


Figure 3.2: Moving object in relation to reference line $\mathcal{R}$ and path boundary $\mathcal{B}$.

the z^I-axis of the inertial coordinate frame. Furthermore, the longitudinal and lateral velocity of the moving object given in the body coordinate frame is represented by v_x and v_y, respectively. These speed components compose the total velocity of the object v_{tot}. For describing the system behaviour relative to the reference curve, the position of the object has to be associated with a point on the reference line. Considering a current point P_S at $s_{\mathcal{R}}$ on the reference curve, $\Theta_{\mathcal{R}}$ depicts the angle around the z^I-axis to the tangent of the reference curve in this point. The deviation between this angle and the vehicle orientation defines the tangent error angle $\Theta = \psi - \Theta_{\mathcal{R}}$. The normal distance between the tangent and the COG of the object is represented by the lateral deviation $d_{\mathcal{R}}$. The differential equations describing these curvilinear coordinates are given by [189]

$$\dot{s}_{\mathcal{R}} = \frac{v_x \cos(\Theta) - v_y \sin(\Theta)}{1 - d_{\mathcal{R}}\kappa_{\mathcal{R}}}, \tag{3.6a}$$

$$\dot{d}_{\mathcal{R}} = v_x \sin(\Theta) + v_y \cos(\Theta) \quad \text{and} \tag{3.6b}$$

$$\dot{\Theta} = \dot{\psi} - \kappa_{\mathcal{R}} \dot{s}_{\mathcal{R}}. \tag{3.6c}$$

The derivation of (3.6) is demonstrated in Appendix A.3. Introducing these coordinates enables describing path boundaries $\mathcal{B}$ using simple box constraints with a possibly arc length-dependent distance variable $d_{\mathcal{B}}$, which represents the half-width $d_{\mathcal{B}} = d_{hw}$ when $\mathcal{R}$ is the centre line. An extension to three-dimensional track profiles is presented in [157]. The system description depending on the arc length requires a transformation of the differential equation system (3.1a) according to

$$\mathbf{x}'(s_{\mathcal{R}}) \coloneqq \frac{\partial \mathbf{x}}{\partial s_{\mathcal{R}}} = \frac{d\mathbf{x}}{dt}\frac{\partial t}{\partial s_{\mathcal{R}}} = \dot{\mathbf{x}}\frac{1}{\dot{s}_{\mathcal{R}}} = \frac{1}{\dot{s}_{\mathcal{R}}}\mathbf{f}. \tag{3.7}$$

Thus, a drawback of transformation (3.7) is the introduction of additional nonlinearities into the differential equations. Time information can be retrieved by considering the differential equation

$$t'(s_\mathcal{R}) = \frac{\partial t}{\partial s_\mathcal{R}} = \frac{1}{\dot{s}_\mathcal{R}} \quad \text{with} \quad t(s_{\mathcal{R},0}) = 0. \tag{3.8}$$

For a full description of the system in relation to the reference curve, the state vector is augmented according to

$$\tilde{\mathbf{x}}(s_\mathcal{R}) = \begin{bmatrix} \mathbf{x}(s_\mathcal{R}) \\ \mathbf{x}_{\text{ref}}(s_\mathcal{R}) \end{bmatrix} \in \mathbb{R}^{n_{\tilde{x}}} \quad \text{with} \quad \mathbf{x}_{\text{ref}}(s_\mathcal{R}) = \begin{bmatrix} t(s_\mathcal{R}) & d_r(s_\mathcal{R}) & \Theta(s_\mathcal{R}) \end{bmatrix}^T. \tag{3.9}$$

Using (3.6), the augmented differential equation system is given by

$$\mathbf{x}'(s_\mathcal{R}) = \tilde{\mathbf{f}}(\mathbf{x}(s_\mathcal{R}), \mathbf{u}(s_\mathcal{R}), \mathbf{p}) \quad \text{with} \quad \tilde{\mathbf{f}} = \frac{1}{\dot{s}_\mathcal{R}} \begin{bmatrix} \mathbf{f}^T & 1 & \dot{d}_\mathcal{R} & \dot{\Theta}_\mathcal{R} \end{bmatrix}^T. \tag{3.10}$$

Due to the spatial reformulation, the differential equations in (3.10) depend on the curvature $\kappa_\mathcal{R}(s_\mathcal{R})$ of the reference trajectory. Since the employed numerical solver utilises derivative information, a nonsmooth curvature trajectory can result in poor solver performance. Thus, the following section presents an elaborate smoothing approach for measured position data of reference trajectories.

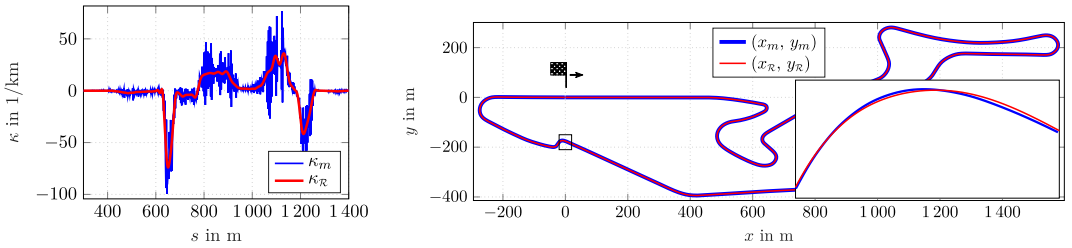
3.3.2 Reference Path Preprocessing

The curvature of the reference curve enters the model equations via the Frenet equations (3.6a) and (3.6c) as well as via the transformation to spatial dependency (3.10). For good convergence properties of derivative-based numerical solvers, the curvature trajectory $\kappa_\mathcal{R}(s_\mathcal{R})$ is required to be sufficiently smooth. For real-world applications, it is desirable to generate appropriate reference trajectories from measured coordinates. This section showcases how a smooth curvature trajectory can be derived from 2D-position data. The approach has been first presented in [185] for racetrack applications however it is applicable to other use cases as well.
The starting point are measured position coordinates $x_m(s_m)$ and $y_m(s_m)$ of the reference curve available in dependence of the arc length $s_m \in [s_{m,0}, s_{m,f}]$. The specification in dependence of the arc length can be achieved using pre-written software like [40]. The geometric relations between position and curvature are given by

$$x_m^{'} = \frac{\partial x_m}{\partial s_m} = \cos(\Theta_m) \qquad y_m^{'} = \frac{\partial y_m}{\partial s_m} = \sin(\Theta_m) \tag{3.11a}$$

$$\Theta_m = \arctan\left(\frac{y_m^{'}}{x_m^{'}}\right) \tag{3.11b}$$

$$\kappa_m = \Theta_m' = \frac{\partial \Theta_m}{\partial s_m} = \frac{y_m^{''} x_m^{'} - y_m^{'} x_m^{''}}{{x_m^{'}}^2 + {y_m^{'}}^2}. \tag{3.11c}$$



(a) Curvature of racetrack centre line.

(b) Coordinates of racetrack centre line.

Figure 3.3: Track data preprocessing.

Therein the tangent angle and curvature are denoted by Θ_m and κ_m, respectively. The crux of the matter is that the curvature is computed via (3.11c) using derivatives of measured data, which amplifies the noise. This results in a highly oscillating curvature trajectory as displayed in Fig. 3.3a. However, smoothing the curvature data manipulates the position data: Computing the x-y-coordinates corresponding to a smoothed curvature trajectory requires a double integration process, which accumulates errors and can hence lead to an unclosed circuit or a strong deviation from the original x-y-trajectory. Thus, the following paragraph presents an elaborate smoothing approach yielding a smooth curvature trajectory that depicts the original x-y-coordinates.
Approximating the curvature trajectory with piecewise cubic polynomials inherently guarantees smoothness since these polynomials are of class $\mathcal{C}^2$. Furthermore, the analytical derivative and integral of the piecewise polynomial form is straightforward. Oscillations are reduced by appropriate positioning of the knot points used for the spline fitting. The optimal positions of $n_\mathcal{R}$ knot points accumulated in the vector $\mathbf{s}_\mathcal{R} = [s_{\mathcal{R},0}, \dots, s_{\mathcal{R},n_\mathcal{R}-1}] \in \mathbb{R}^{n_\mathcal{R}}$ are identified via the multi-objective OP

$$\min_{\mathbf{s}_\mathcal{R}} \quad \sum_{i=0}^{n_e-1} \underbrace{\alpha_\kappa \kappa_\mathcal{R}'(s_{e,i})^2}_{=:J_{\text{smooth},i}} + \underbrace{(x_m(s_{e,i}) - x_\mathcal{R}(s_{e,i}))^2 + (y_m(s_{e,i}) - y_\mathcal{R}(s_{e,i}))^2}_{=:J_{\text{fit},i}} \tag{3.12a}$$

$$\text{s.t.} \quad s_{\mathcal{R},0} = s_{m,0}, \tag{3.12b}$$

$$s_{m,0} < s_{\mathcal{R},j} \leq s_{m,f}, \ \forall j = 1, \dots, n_\mathcal{R} - 1, \tag{3.12c}$$

$$s_{\mathcal{R},j} < s_{\mathcal{R},j+1}, \ \forall j = 0, \dots, n_\mathcal{R} - 2, \tag{3.12d}$$

$$\Theta_{\mathcal{R},n_e} = \Theta_{\mathcal{R},0} - 2\pi, \tag{3.12e}$$

$$\Delta d_{\mathcal{R},n_e} = -\sin(\Theta_{\mathcal{R},0})(x_{\mathcal{R},0} - x_{\mathcal{R},n_e}) + \cos(\Theta_{\mathcal{R},0})(y_{\mathcal{R},0} - y_{\mathcal{R},n_e}) = 0. \tag{3.12f}$$

The cost function in (3.12a) is comprised of two objectives. A smoothing objective $J_{\text{smooth},i}$ that minimises the derivative of the curvature generating a preference for a smooth curvature trajectory. The fitting objective $J_{\text{fit},i}$ penalises errors between the original and fitted position trajectories. The individual objectives are weighed relatively to each other using the scaling parameter α_κ, trading curvature oscillation for position error. The number of spline knot points $n_\mathcal{R}$ is kept rather low to reduce optimisation time. However, the curvature and position trajectories are evaluated on a fine and uni-

form grid of n_e evaluation points $\mathbf{s}_e \in [s_{\mathcal{R},0}, s_{\mathcal{R},n_\mathcal{R}-1}] \in \mathbb{R}^{n_e}$ via spline interpolation. The finer grid $n_e > n_\mathcal{R}$ is chosen to avoid errors between the original and fitted position trajectories in between knot points. The first knot point is fixed using the equality constraint (3.12b). Inequality constraint (3.12c) ensures that the knot points lie within the maximum arc length of the measured trajectory. A strictly increasing arc length is enforced via (3.12d), which is implemented as linear matrix inequality constraint. Considering racetrack applications require a closed circuit, equality constraints for the tangent angle (3.12e) and the lateral distance (3.12f) are introduced for the final knot point.

Although the curvature can be directly computed via (3.11c), it is advisable to introduce an intermediate spline fitting according to Algorithm 1 to avoid using second-order derivatives of measured data and thus a highly oscillating reference trajectory. Highly oscillating objectives have a large number of local minima increasing the difficulty of finding the optimal solution especially for derivative-based optimisation schemes. The detour via an intermediate fitting of Θ in line 8 only requires first-order derivatives of the measurement data yielding a smoother reference trajectory and accelerating the optimisation. The tangent angle trajectory is computed via (3.11b) and interpolated using

Algorithm 1 Cost function for track data smoothing algorithm.

1: **INPUTS**:
   - $\mathbf{x}_m(s_m), \mathbf{y}_m(s_m)$: vectors containing arc length dependent position points
   - $\mathbf{s}_\mathcal{R}, \mathbf{s}_e$: arc length at knot points and evaluation points
2: **OUTPUT**: J: cost function value
3: **MAIN FUNCTION**:
4: $x_m(s) \leftarrow \text{pwCS}(\mathbf{s}_m, \mathbf{x}_m)$ and $y_m(s) \leftarrow \text{pwCS}(\mathbf{s}_m, \mathbf{y}_m)$[a]
5: $x'_m(s) \leftarrow \text{DpwCS}(\mathbf{s}_\mathcal{R}, x_m(s))$ and $y'_m(s) \leftarrow \text{DpwCS}(\mathbf{s}_\mathcal{R}, y_m(s))$[b]
6: $\mathbf{x}'_{m,\text{ref}} \leftarrow x'_m(\mathbf{s}_\mathcal{R})$ and $\mathbf{y}'_{m,\text{ref}} \leftarrow y'_m(\mathbf{s}_\mathcal{R})$
7: $\boldsymbol{\Theta}_{m,\text{ref}} \leftarrow \arctan(\mathbf{y}'_{m,\text{ref}}/\mathbf{x}'_{m,\text{ref}})$ (3.11b)
8: $\Theta_{\mathcal{R},\text{temp}}(s) \leftarrow \text{pwCS}(\mathbf{s}_\mathcal{R}, \boldsymbol{\Theta}_{m,\text{ref}})$
9: $\kappa_{\mathcal{R},\text{temp}}(s) \leftarrow \text{DpwCS}(\mathbf{s}_\mathcal{R}, \Theta_{\mathcal{R},\text{temp}}(s))$
10: $\boldsymbol{\kappa}_{\mathcal{R},\text{ref}} \leftarrow \kappa_{\mathcal{R},\text{temp}}(\mathbf{s}_\mathcal{R})$
11: $\kappa_\mathcal{R}(s) \leftarrow \text{pwCS}(\mathbf{s}_\mathcal{R}, \boldsymbol{\kappa}_{\mathcal{R},\text{ref}})$
12: $\Theta_\mathcal{R}(s) \leftarrow \text{IpwCS}(\mathbf{s}_\mathcal{R}, \kappa_\mathcal{R}(s))$[c]
13: $\kappa'_\mathcal{R}(s) \leftarrow \text{DpwCS}(\mathbf{s}_\mathcal{R}, \kappa_\mathcal{R}(s))$
14: $\boldsymbol{\Theta}_{\mathcal{R},\text{ref}} \leftarrow \Theta_\mathcal{R}(\mathbf{s}_\mathcal{R})$
15: $\mathbf{x}'_{\mathcal{R},\text{ref}} \leftarrow \cos(\boldsymbol{\Theta}_{\mathcal{R},\text{ref}})$ and $\mathbf{y}'_{\mathcal{R},\text{ref}} \leftarrow \sin(\boldsymbol{\Theta}_{\mathcal{R},\text{ref}})$ (3.11a)
16: $x'_\mathcal{R}(s) \leftarrow \text{pwCS}(\mathbf{s}_\mathcal{R}, \mathbf{x}'_{\mathcal{R},\text{ref}})$ and $y'_\mathcal{R}(s) \leftarrow \text{pwCS}(\mathbf{s}_\mathcal{R}, \mathbf{y}'_{\mathcal{R},\text{ref}})$
17: $x_\mathcal{R}(s) \leftarrow \text{IpwCS}(\mathbf{s}_\mathcal{R}, x'_\mathcal{R}(s))$ and $y_\mathcal{R}(s) \leftarrow \text{IpwCS}(\mathbf{s}_\mathcal{R}, y'_\mathcal{R}(s))$
18: $J \leftarrow \sum_{i=0}^{n_e-1} J_{\text{smooth},i}(\kappa'_\mathcal{R}(\mathbf{s}_{e,i})) + J_{\text{fit},i}(x_m(\mathbf{s}_{e,i}),\, y_m(\mathbf{s}_{e,i}),\, x_\mathcal{R}(\mathbf{s}_{e,i}),\, y_\mathcal{R}(\mathbf{s}_{e,i}))$ (3.12a)

[a] pwCS(): generation of piecewise cubic splines via solution of a tridiagonal linear system [18, p.43].
[b] DpwCS(): analytical derivative of piecewise cubic splines.
[c] IpwCS(): analytical integral of piecewise cubic splines.
Bold variables represent data point vectors and piecewise spline functions are depicted with light variables.

piecewise cubic splines. The analytical derivatives of these tangent angle splines yield piecewise quadratic curvature splines. For $\mathcal{C}^2$-smoothness of the curvature trajectory, piecewise cubic splines are subsequently fitted to these quadratic splines in line 11. Since the analytical integral of these splines is known, the corresponding piecewise quadratic tangent angle trajectory is defined as well. The trajectories for the position derivatives are computed using the tangent angle trajectories and (3.11a). A subsequent integration yields the position trajectories. However, the analytical integral is not straightforwardly computable due to the nonlinear relation. Thus, a subordinate spline fitting of the position derivatives is performed in line 16 and the position trajectories are then computed via analytical integration[6]. Although this additional fitting process can potentially introduce a discrepancy between curvature trajectory and position trajectories, a sufficiently high number of knot points yields practically negligible numerical errors. Since racetracks are generally smooth, relatively small knot point numbers already yield a good coincidence of the original and fitted trajectories.
The results of applying Algorithm 1 to measurements of 2D-position data for the Nürburgring Grand-Prix course are displayed in Fig. 3.3. The curvature trajectory in Fig. 3.3a is much smoother than the trajectory computed via (3.11c). The corresponding coordinate trajectories are illustrated in Fig. 3.3b. With a small mean deviance of 2.76 centimetres between the coordinates, the fitted position trajectories match well the original position data. These results confirm the correct functioning of Algorithm 1.

Comparison with Related Methods Subsequently, the presented procedure is compared with other existing methods for trajectory smoothing. The notion of optimising the position of spline knot points within a fitting process has been used before in [21, 69, 87, 197, 215]. However, the method previously presented in this section simultaneously optimises three trajectories, both position trajectories and the curvature trajectory, which are related to each other via integration and nonlinear equations resulting in a different OP. Firstly, the smoothing objective $J_{\text{smooth},i}$ and fitting objective $J_{\text{fit},i}$ in (3.12a) relate to different trajectories yielding a different cost function. Secondly, constraints (3.12e) and (3.12f) are added to guarantee a closed trajectory.
However, there have been multiple algorithms specifically designed for preprocessing of track data using various procedures. Filtering, trajectory stretching and the use of error-compensation terms are displayed in [30]. The approach in [157, 158] formulates the smoothing task as an OCP which is transcribed using orthogonal collocation yielding a SOP similar to (3.12). However, each differential equation is approximated using Lagrange polynomials. Exact correspondence of these polynomials with the differential equations is only guaranteed at the generally roughly spaced collocation points. The application of hp-adaptive mesh refinement strategies, for instance implemented in *GPOPS-II* [155], can reduce approximation errors introduced by such a polynomial approximation.
By using analytical derivatives and integrals of the piecewise polynomials, Algorithm 1 ensures the satisfaction of the differential equations in each point. The approximation

[6] Alternatively, numerical integration can be employed.

errors introduced by the intermediate fittings are generally negligible. Requiring only cubic spline interpolation and static optimisation, an easy implementation is possible with open-source or well-known software frameworks such as *Julia* [14] or *MATLAB* [138]. This is especially advantageous when no software for hp-adaptive collocation methods is available.

3.4 Scaling of Optimal Control Problem

In order to reduce the computation time required by the NLP solver, suitable domains should be enforced for the decision variables, the cost function and the constraints via scaling. As indicated in Section 3.1, the inputs, states and adjustable model parameters represent decision variables. Adopting the scaling procedure presented in [183], which is derived from [3, 13], the OP is scaled such that the decision variables lie within the range $\mathcal{R}_y := [-1, 1]$ and inequality constraints take values within $\mathcal{R}_g := [-1, 0]$. Furthermore, the objective function is multiplied by a scalar to take values within the domain $\mathcal{R}_J := [0, 150]$. Using the vector of optimisation parameters $\mathbf{y} \in \mathbb{R}^{n_y}$ to represent the states $\tilde{\mathbf{x}}$, the inputs $\mathbf{u}$ or the passive optimisation parameters $\mathbf{p}$, the following equations describe the scaling procedure. The relation between the original decision variables $\mathbf{y}$ and the scaled and shifted counterpart $\hat{\mathbf{y}}$ is given by

$$\hat{\mathbf{y}} = \boldsymbol{\varphi}_y(\mathbf{y}) = \mathbf{S}_y \mathbf{y} + \mathbf{k}_y \quad \Leftrightarrow \quad \mathbf{y} = \boldsymbol{\varphi}_y^{-1}(\mathbf{y_s}) = \mathbf{S}_y^{-1} \left(\hat{\mathbf{y}} - \mathbf{k}_y \right). \tag{3.13a}$$

Assuming approximated constant bounds $y_i \in [\underline{y}_i, \overline{y}_i] \; \forall i = 1, \ldots, n_y$, the scaling matrix $\mathbf{S}_y \in \mathbb{R}^{n_y \times n_y}$ and the shifting vector $\mathbf{k}_y \in \mathbb{R}^{n_y}$ are selected according to

$$\mathbf{S}_y = \operatorname{diag}\left(s_{y,1}, \ldots, s_{y,n_y} \right) \quad \text{with} \quad s_{y,i} = \frac{2}{\overline{y}_i - \underline{y}_i} \quad \forall i = 1, \ldots, n_y \tag{3.13b}$$

$$\mathbf{k}_y = \begin{bmatrix} k_{y,1} & \ldots & k_{y,n_y} \end{bmatrix}^T \quad \text{with} \quad k_{y,i} = 1 - \frac{2\overline{y}_i}{\overline{y}_i - \underline{y}_i} \quad \forall i = 1, \ldots, n_y \tag{3.13c}$$

yielding decision variables ranging within the domain $\mathcal{R}_y$ [3]. The application of scaling (3.13a) requires the differential equation system (3.10) to be adjusted:

$$\begin{aligned} \hat{\mathbf{f}}(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\mathbf{p}}) = \hat{\mathbf{x}}' = \frac{\partial \boldsymbol{\varphi}_{\tilde{x}}(\tilde{\mathbf{x}})}{\partial s_{\mathcal{R}}} = \frac{\partial \boldsymbol{\varphi}_{\tilde{x}}(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}} \frac{\mathrm{d}\tilde{\mathbf{x}}}{\mathrm{d}s_{\mathcal{R}}} = \mathbf{S}_{\tilde{x}} \tilde{\mathbf{x}}' = \mathbf{S}_{\tilde{x}} \tilde{\mathbf{f}}(\tilde{\mathbf{x}}, \mathbf{u}, \mathbf{p}) = \\ = \mathbf{S}_{\tilde{x}} \tilde{\mathbf{f}}\Big(\mathbf{S}_{\tilde{x}}^{-1} \left(\hat{\mathbf{x}} - \mathbf{k}_{\tilde{x}} \right), \mathbf{S}_u^{-1} \left(\hat{\mathbf{u}} - \mathbf{k}_u \right), \mathbf{S}_p^{-1} \left(\hat{\mathbf{p}} - \mathbf{k}_p \right) \Big). \end{aligned} \tag{3.13d}$$

For the scaling of vector $\mathbf{g}(\tilde{\mathbf{x}}, \mathbf{u}, \mathbf{p}) \in \mathbb{R}^{n_g}$, which represents a concatenation of all inequality constraints, a constant lower bound $\underline{g}_i \; \forall \; i = 1, \ldots, n_g$ is assumed to be known for each constraint. In order to yield the constraint range $\mathcal{R}_g$, following scaling is applied:

$$\hat{\mathbf{g}}(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\mathbf{p}}) = \mathbf{S}_g \mathbf{g}(\tilde{\mathbf{x}}, \mathbf{u}, \mathbf{p}) \quad \text{with} \quad \mathbf{S}_g = \operatorname{diag}\left(\frac{1}{|\underline{g}_1|}, \ldots, \frac{1}{|\underline{g}_{n_g}|} \right). \tag{3.13e}$$

In this thesis, the objective J is presumed to only take positive values. Assuming an approximate cost function value $J^\star$ at the optimum is available, the cost function is scaled following

$$\hat{J}(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\mathbf{p}}) = \frac{150}{J^\star} J(\tilde{\mathbf{x}}, \mathbf{u}, \mathbf{p}) \tag{3.13f}$$

to yield objective values approximately within the region $\mathcal{R}_J$. After scaling the differential equations, constraints and cost function, the collocation approach can be applied to convert the dynamic OP into a static one. This procedure is presented in the following section.

3.5 Hermite-Simpson Collocation

Various solution methods for OCP have been discussed in Section 2.3. In this thesis, direct Hermite-Simpson collocation is employed for the solution of OCPs due to following reasons. A direct method is preferred over an indirect method due to the more robust initialisation properties. Compared to DP, direct methods require significantly shorter computation times. Furthermore, this thesis aims at accurately solving OCPs for systems with complicated differential equations of possibly larger order. This excludes DP due to the dimensionality problem or the required approximations for ADP. Collocation methods discretise the input trajectories as well as the state trajectories. As illustrated in Section 2.3.2, this procedure minimises couplings, which reduces the complexity of equations enhancing solvability of the OP. Hermite-Simpson collocation is based on dividing the integration interval into segments and approximating the trajectories for the controls and the right-hand sides of the differential equations by piecewise quadratic polynomials. The low polynomial order results in only mild couplings between adjacent decision variables while the piecewise quadratic polynomials yield piecewise cubic, hence smooth, state trajectories. Utilising specifically the separated form of Hermite-Simpson collocation, sparsity in the differential equations is fully exploited by also using the mid points of each segment as collocation points [13, p.143]. Thus, the selected approach represents a good compromise between accuracy and sparsity.
Hermite-Simpson collocation employs Simpson's Rule for numerical integration, which is defined as

$$\int_{t_0}^{t_f} w(\tau) d\tau \approx \frac{t_f - t_0}{6} \left[w(t_0) + 4\, w\left(t_0 + \frac{t_f - t_0}{2} \right) + w(t_f) \right]. \tag{3.14}$$

Therein the integrand $w(\tau)$ is replaced by a quadratic polynomial with coinciding values at the start, mid and end point. These three points represent the collocation points for the segment. The discretisation is applied to the transformed and scaled continuous system dynamics (3.13d) using the scaled decision variables. The integration interval for the spatial independent variable $s_\mathcal{R} \in [s_{\mathcal{R},0}, s_{\mathcal{R},f}]$ is divided into n_seg segments with the individual segment width $\Delta_i = s_{\mathcal{R},i+1} - s_{\mathcal{R},i}$ yielding $n_\text{coll} = 2\, n_\text{seg} + 1$ collocation points.

Applying (3.14) on each segment provides the collocation constraints

$$\mathbf{c}_{\text{coll},i} \coloneqq \begin{bmatrix} \hat{\mathbf{x}}_{i+1} - \hat{\mathbf{x}}_i - \frac{1}{6}\Delta_i(\hat{\mathbf{f}}_i + 4\,\hat{\mathbf{f}}_{i+\frac{1}{2}} + \hat{\mathbf{f}}_{i+1}) \\ \hat{\mathbf{x}}_{i+\frac{1}{2}} - \frac{1}{2}(\hat{\mathbf{x}}_i + \hat{\mathbf{x}}_{i+1}) - \frac{1}{8}\Delta_i(\hat{\mathbf{f}}_i - \hat{\mathbf{f}}_{i+1}) \end{bmatrix} = \mathbf{0} \quad \forall i = 0, 1, \dots, n_{\text{seg}} - 1 \tag{3.15}$$

with $\hat{\mathbf{f}}_j = \hat{\mathbf{f}}(\hat{\mathbf{x}}_j, \hat{\mathbf{u}}_j, \hat{\mathbf{p}})$ [13, 97]. The margins of a segment are given by two collocation points represented by $(\cdot)_i$ and $(\cdot)_{i+1}$, whereas the collocation point in the middle between them is marked as $(\cdot)_{i+\frac{1}{2}}$.
The optimisation process computes optimal values for the inputs and states at the collocation points. In a post-processing step, the intermediate values are computed by applying piecewise quadratic and piecewise cubic interpolation for the input and state trajectories, respectively. Thus, the input values are determined by

$$\hat{\mathbf{u}}(s_{\mathcal{R}}) = \frac{2}{\Delta_i^2}\Big(\sigma - \frac{\Delta_i}{2}\Big)\Big(\sigma - \Delta_i\Big)\hat{\mathbf{u}}_i - \frac{4}{\Delta_i^2}\sigma\Big(\sigma - \Delta_i\Big)\hat{\mathbf{u}}_{i+\frac{1}{2}} + \frac{2}{\Delta_i^2}\sigma\Big(\sigma - \frac{\Delta_i}{2}\Big)\hat{\mathbf{u}}_{i+1} \tag{3.16a}$$

for the i^{th} segment with $s_{\mathcal{R}} \in [s_{\mathcal{R},i}, s_{\mathcal{R},i+1}]$ using $\sigma = s_{\mathcal{R}} - s_{\mathcal{R},i}$. The input derivatives are deduced by differentiating (3.16a) resulting in

$$\hat{\mathbf{u}}'(s_{\mathcal{R}}) = \underbrace{\frac{1}{\Delta_i}\Big(-3\hat{\mathbf{u}}_i + 4\hat{\mathbf{u}}_{i+\frac{1}{2}} - \hat{\mathbf{u}}_{i+1}\Big)}_{=:\boldsymbol{\Lambda}_{0,i}} + \underbrace{\frac{2}{\Delta_i^2}\Big(2\hat{\mathbf{u}}_i - 4\hat{\mathbf{u}}_{i+\frac{1}{2}} + 2\hat{\mathbf{u}}_{i+1}\Big)}_{=:\boldsymbol{\Lambda}_{1,i}}\sigma. \tag{3.16b}$$

The continuously differentiable state trajectories are defined by the piecewise cubic polynomials

$$\hat{\mathbf{x}}(s_{\mathcal{R}}) = \hat{\mathbf{x}}_i + \sigma\hat{\mathbf{f}}_i + \frac{\sigma^2}{2\Delta_i}\Big(-3\,\hat{\mathbf{f}}_i + 4\,\hat{\mathbf{f}}_{i+\frac{1}{2}} - \hat{\mathbf{f}}_{i+1}\Big) + \frac{\sigma^3}{3\Delta_i^2}\Big(2\,\hat{\mathbf{f}}_i - 4\,\hat{\mathbf{f}}_{i+\frac{1}{2}} + 2\,\hat{\mathbf{f}}_{i+1}\Big). \tag{3.16c}$$

After applying the discretisation approach, the infinite dimensional OP is transformed into a finite one, which is listed in the following section.

3.6 Transcribed Optimisation Problem

The SOP resulting from applying the separated form of Hermite-Simpson collocation is presented in this section. On each segment, the margin points as well as the mid point represent collocation points yielding n_{coll} decision variables for each component of the state vector $\hat{\mathbf{x}} \in \mathbb{R}^{n_{\tilde{x}}}$ and input vector $\hat{\mathbf{u}} \in \mathbb{R}^{n_u}$. Moreover, all adjustable parameters $\hat{\mathbf{p}} \in \mathbb{R}^{n_p}$ are further decision variables. In total, this results in $n_{\text{opt}} = (n_{\tilde{x}} + n_u)\,n_{\text{coll}} + n_p$ decision variables. Accumulating the state decision variables in $\mathbf{X} = \begin{bmatrix}\hat{\mathbf{x}}_0 & \hat{\mathbf{x}}_{\frac{1}{2}} & \dots & \hat{\mathbf{x}}_{n_{\text{seg}}}\end{bmatrix} \in \mathbb{R}^{n_{\tilde{x}} \times n_{\text{coll}}}$ and the input decision variables in $\mathbf{U} = \begin{bmatrix}\hat{\mathbf{u}}_0 & \hat{\mathbf{u}}_{\frac{1}{2}} & \dots & \hat{\mathbf{u}}_{n_{\text{seg}}}\end{bmatrix} \in \mathbb{R}^{n_u \times n_{\text{coll}}}$, the SOP pursuing a minimum-time objec-

tive is stated in the following form:

$$\begin{aligned}
\min_{\mathbf{X},\mathbf{U},\hat{\mathbf{p}}} \quad & J_t + \varepsilon_{\dot{u}}\, J_{\dot{u}} && \text{(3.17a)}\\
\text{s.t.} \quad & \hat{\mathbf{x}}_k \in \hat{\mathcal{X}}, \quad \hat{\mathbf{u}}_k \in \hat{\mathcal{U}}, \quad \hat{\mathbf{p}} \in \hat{\mathcal{P}} \quad \forall k \in \mathcal{K}, && \text{(3.17b)}\\
& \mathbf{c}_{\text{coll},i}(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i, \hat{\mathbf{x}}_{i+1}, \hat{\mathbf{u}}_{i+1}, \hat{\mathbf{p}}) = \mathbf{0} \quad \forall i \in \mathcal{I}_{\text{coll}}, && \text{(3.17c)}\\
& \mathbf{c}_k(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k, \hat{\mathbf{p}}) = \mathbf{0} \quad \forall k \in \mathcal{K}, && \text{(3.17d)}\\
& \hat{\mathbf{h}}_k(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k, \hat{\mathbf{p}}) \leq \mathbf{0} \quad \forall k \in \mathcal{K}, && \text{(3.17e)}\\
& \boldsymbol{\Phi}(\hat{\mathbf{x}}_0, \hat{\mathbf{u}}_0, \hat{\mathbf{x}}_{n_{\text{seg}}}, \hat{\mathbf{u}}_{n_{\text{seg}}}, \hat{\mathbf{p}}) = \mathbf{0}. && \text{(3.17f)}
\end{aligned}$$

Therein $\mathbf{c}_{\text{coll},i}$ represent the collocation constraints (3.15) for each collocation segment $i \in \mathcal{I}_{\text{coll}}$. Further equality and scaled inequality constraints at the collocation points $k \in \mathcal{K}$ are depicted by $\mathbf{c}_k$ and $\hat{\mathbf{h}}_k$, respectively. The set of collocation segment indices and collocation point indices is given by

$$\mathcal{I}_{\text{coll}} \coloneqq \{0, 1, 2, \ldots, n_{\text{seg}} - 1\} \text{ and} \tag{3.18a}$$

$$\mathcal{K} \coloneqq \{0, \frac{1}{2}, 1, \ldots, n_{\text{seg}}\}, \tag{3.18b}$$

respectively. The boundary conditions are considered via $\boldsymbol{\Phi}$. Furthermore, all states, inputs and parameters are limited by the scaled box constraints in (3.17b) given by the convex sets $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$ and $\hat{\mathcal{P}}$, respectively. Considering the smoothing procedures presented in Section 3.2, all constraints are assumed to be continuously differentiable. Scaling is performed according to Section 3.4. The cost function (3.17a) is comprised of the minimum-time objective

$$\begin{aligned}
J_t &= \int_{t_0}^{t_f} 1 dt = \int_{s_{\mathcal{R},0}}^{s_{\mathcal{R},f}} \frac{dt}{ds_{\mathcal{R}}} ds_{\mathcal{R}} = \int_{s_{\mathcal{R},0}}^{s_{\mathcal{R},f}} \frac{1}{\dot{s}_{\mathcal{R}}} ds_{\mathcal{R}} = t(s_{\mathcal{R},f}) - t(s_{\mathcal{R},0})\\
&= t(s_{\mathcal{R},f}) = t_{n_{\text{seg}}}.
\end{aligned} \tag{3.19}$$

and a regularisation term $J_{\dot{u}}$

$$\begin{aligned}
J_{\dot{u}} &= \sum_{j=1}^{n_u} \int_{s_{\mathcal{R},0}}^{s_{\mathcal{R},f}} \hat{u}_j'^2 ds_{\mathcal{R}} \approx\\
&\approx \sum_{j=1}^{n_u} \sum_{i=0}^{n_{\text{seg}}-1} \frac{\Delta_i}{6} \left(\hat{u}_j'(s_{\mathcal{R},i})^2 + 4\, \hat{u}_j'(s_{\mathcal{R},i} + \frac{1}{2}\Delta_i)^2 + \hat{u}_j'(s_{\mathcal{R},i} + \Delta_i)^2 \right) =\\
&= \sum_{j=1}^{n_u} \sum_{i=0}^{n_{\text{seg}}-1} \frac{\Delta_i}{6} (6\, \Lambda_{0,i,j}^2 + 6\, \Lambda_{0,i,j} \Lambda_{1,i,j} \Delta_i + 2\, \Lambda_{1,i,j}^2 \Delta_i^2)
\end{aligned} \tag{3.20}$$

that penalises input oscillations using the input derivatives (3.16b). From an application standpoint, smooth control trajectories are generally preferable for vehicular applications due to actuator limitations or operating effort, which legitimises (3.20). Furthermore, the additional objective function term (3.20) improves convergence properties. OPs

exhibiting a family of optimal solutions rather than a unique optimal solution can cause convergence problems: The numerical solver searches for the locally optimal solution among equally good solutions oscillating between the individual solutions and failing to converge [97]. Trajectory OPs with non-unique solutions often have singular arcs, which occur when the OCP is not uniquely defined by the optimality conditions. In this case, the Hessian of the extended Hamiltonian matrix (2.18b) with respect to the inputs $\tilde{\mathcal{H}}_{\mathbf{uu}}$ is singular [13, pp.125/212][152, p.261]. An additional objective term that is quadratic in $\mathbf{u}$ can be used to remedy the problem by modifying the Hessian [97, 116, 137]. Although the combined objective (3.17a) is not strictly time-optimal, the cost manipulation by (3.20) can greatly improve convergence and is negligible when choosing a sufficiently small regularisation weight $\varepsilon_{\dot{u}}$.
The SOP (3.17) can be solved using elaborate NLP solvers. This thesis applies the open-source solver *IPOPT*. The approach presented in this chapter can be used to solve a broad class of OPs with different objectives instead of (3.19). In the following section, the approach is compared with other software frameworks for the solution of nonconvex OCPs however focussing on vehicular applications.

3.7 Comparison with Related Methods

Although the approach presented in this chapter is capable of solving problems for a wide range of applications, this section concentrates on lap time optimisation since it represents a focus of this thesis. The corresponding related literature has been presented in Section 1.1.
Graph search and incremental search methods are generally utilised for motion planning. Using rather simple models or being restricted by predefined motion primitives, these simplifications yield suboptimal solutions. However, search methods can be used to mitigate the problem of high computation times that can occur for variational methods. For instance in [4], search methods are employed to generate approximately optimal paths that are tracked via variational methods meeting a compromise between computation time and accuracy or optimality. Geometrical relations and QSS simulations have also been used to generate velocity profiles that can be tracked via variational methods and other controllers, as mentioned in Section 1.1. Although the concept of separating the trajectory OP into trajectory generation and trajectory tracking can reduce computation time, it generally results in suboptimal solutions.
The method presented in this chapter focusses rather on accuracy than on computation speed. The goal is to provide a general framework capable of considering highly nonlinear models and aiming at identifying optimal control strategies as well as optimal model parameters. Thus, variational methods solving the trajectory OP are more suitable: Due to the flexibility in the problem formulation, a broad range of OPs can be solved. Since MPC introduces suboptimal solutions and the issue of infeasibility and stability due to the limited optimisation horizon, the MPC methods from Section 1.1 are not revisited here. Furthermore, since accurate models are generally of high dimension, DP is not suitable due to the curse of dimensionality. This leaves only direct and indi-

rect variational methods. The indirect optimal control software *PINS* [10, 11] has been used successfully for solving vehicular OCPs [15, 16, 124, 203]. Due to their robustness regarding the initial guess and the omitted necessity of deriving optimality conditions, direct methods are often preferred. Especially direct collocation is commonly used for maximum sparsity in order to enhance convergence properties. Often employed collocation frameworks for direct optimal control are *ICLOCS* [59] and *GPOPS-II* [155] using trapezoidal and orthogonal collocation, respectively. Trapezoidal collocation implements piecewise linear trajectories on the collocation segments yielding a nonsmooth solution. Orthogonal collocation deploys Lagrange polynomials as basis functions, whereas the collocation points are commonly obtained from the roots of Chebyshev polynomials or Legendre polynomials [166][155][47, p.184]. Using polynomials of higher order generates smooth solutions. However, an increasing polynomial order increases local density of the OP, which can deteriorate solvability [158]. Elaborate adaptive schemes exist that adjust the polynomial order and the placement of the collocation points [155, 166] in order to improve convergence rate and accuracy. With Hermite-Simpson collocation, a simple but effective approach is selected in this thesis. The quadratic polynomials yield smooth state trajectories while discontinuities are possible in the input trajectories. Furthermore, the small polynomial order yields sparse matrices enhancing solvability. A fixed discretisation mesh of constant width is used but a more elaborate meshing could improve the convergence rate. However, this is a subject for future work. Although some smoothing functions have been introduced in [158], additional smoothing functions are proposed in this thesis extending the applicability to a larger problem class. Furthermore, the method for generating smooth curvature data introduced in Section 3.3.2 represents a simple, alternative procedure for the preprocessing of reference trajectories.

4 Nonconvex Lap Time Optimisation for Vehicles using Nonlinear Programming

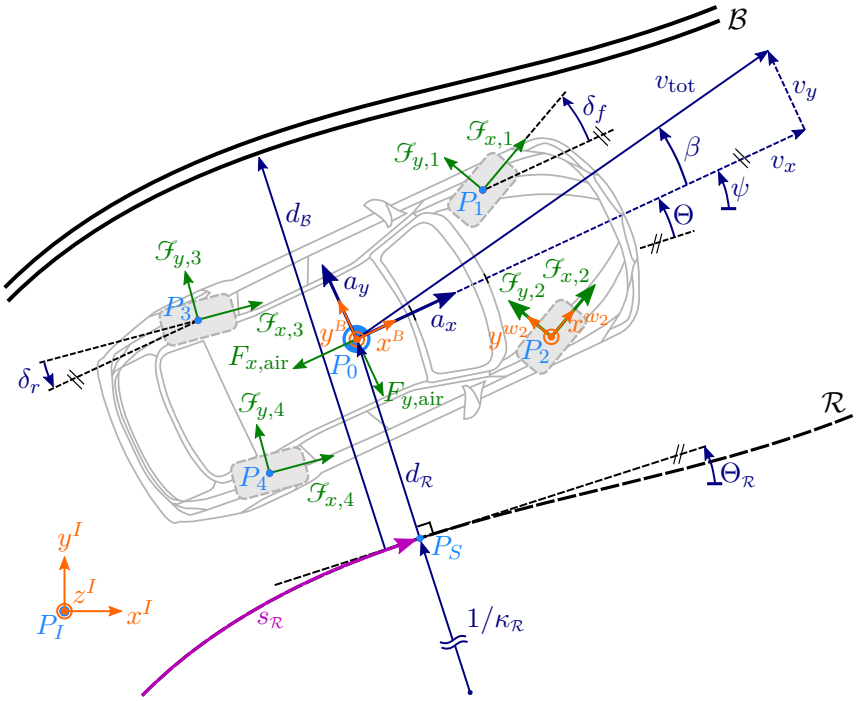
Many OCPs for engineering applications require the solution of nonconvex OPs with highly nonlinear equations. Chapter 3 presented an approach for the solution of such problems. In order to verify that the algorithm is capable of solving complicated optimisation tasks, this section deals with computing minimum-time trajectories for a vehicular racetrack application. This chapter focuses on electrical overloading (EOL) and is based on our publications [183] and [185]. However, the approach has also been used to analyse the benefits of rear-axle steering and transfer cases while considering the concurrent optimisation of selected vehicle parameters. The corresponding results are showcased in our publication [185].
In Section 4.1, a sophisticated vehicle model is derived that depicts the major physical effects of the system. Augmentations for the description of electrical dynamics are listed in Section 4.2. Smooth approximations of discontinuities in the model equations are illustrated in Section 4.3. The OP is discussed in Section 4.4 and the corresponding results are presented in Section 4.5.

4.1 Nonlinear Two-Track Vehicle Model

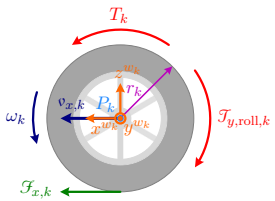
When driving on a racetrack, the vehicle is operated at the limits of handling. A good representation of the vehicle behaviour at this rather demanding conditions requires a sophisticated model. For this purpose, this section presents a nonlinear two-track vehicle model capable of depicting the majorly occurring physical effects in vehicles. Aiming at keeping results tractable and computations feasible, it is important to represent only the essential dynamics and reduce model complexity if possible. Since conditions on racetracks are rather quasi steady-state, the model neglects vertical dynamics and computes the wheel loads assuming quasi steady-state conditions. Furthermore, the current application focusses on comparing vehicles with different powertrain topologies. Aiming at identifying relative differences between the individual vehicle configurations, neglecting the vertical dynamics is acceptable. A tire model of medium complexity is selected for sufficient accuracy but tractable parameter identification. Thus, the implemented model represents a good compromise between accuracy and complexity especially suitable for racetrack applications. The differential equations describing the basic vehicle behaviour are derived in Section 4.1.1 and validated with measured data in Section 4.1.2.

4.1.1 Model Equations

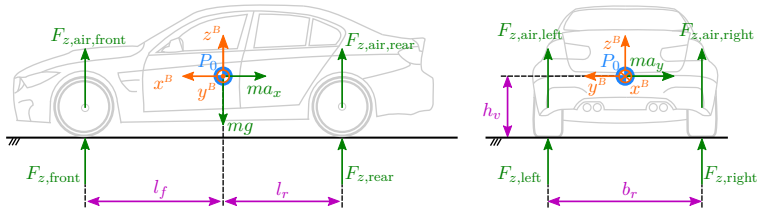
Fig. 4.1 illustrates the physical conditions considered by the model: The topview of the vehicle model on a racetrack is illustrated in Fig. 4.1a with the corresponding free body diagram of the wheels shown in Fig. 4.1b. Introducing various coordinate frames simplifies the derivation of the basic model equations. The world coordinate frame is depicted by the inertial coordinates (x^I, y^I, z^I). Furthermore, the body coordinate frame using the coordinates (x^B, y^B, z^B) moves with the vehicle and has its origin P_0 in the COG of the vehicle. Thus, it rotates with the yaw angle ψ around the z^I-axis, which coincides with the z^B-axis since only planar motions are considered. Coordinates $(x^{w_k}, y^{w_k}, z^{w_k})$



(a) Top view of vehicle model on a racetrack with centre line $\mathcal{R}$ and track border $\mathcal{B}$.



(b) Forces and torques acting on wheel.



(c) Computation of quasi steady-state wheel loads.

Figure 4.1: Vehicle model.

Table 4.1: Parameters for nonlinear two-track vehicle model.

	Variable	Value	Unit	Description
env.	g	9.81	ms^{-2}	gravitational acceleration
	μ	1.0	–	road friction coefficient
	ρ_{air}	1.204	kgm^{-3}	air density
tires	$\bar{F}_a/\bar{F}_b$	2000.0/6000.0	N	reference load a/b
	$\bar{\lambda}_{x,a,1/2}/\bar{\lambda}_{x,b,1/2}$	0.123/0.109	–	front long. slip coeff. at load a/b
	$\bar{\lambda}_{y,a,1/2}/\bar{\lambda}_{y,b,1/2}$	0.129/0.119	–	front lat. slip coeff. at load a/b
	$\bar{D}_{x,a,1/2}/\bar{D}_{x,b,1/2}$	1.512/1.219	–	front long. friction coeff. at load a/b
	$\bar{D}_{y,a,1/2}/\bar{D}_{y,b,1/2}$	1.472/1.427	–	front lat. friction coeff. at load a/b
	$C_{x,1/2}/C_{y,1/2}$	1.949/1.938	–	front long./lat. shape factor
	$\bar{\lambda}_{x,a,3/4}/\bar{\lambda}_{x,b,3/4}$	0.116/0.106	–	rear long. slip coeff. at load a/b
	$\bar{\lambda}_{y,a,3/4}/\bar{\lambda}_{y,b,3/4}$	0.108/0.094	–	rear lat. slip coeff. at load a/b
	$\bar{D}_{x,a,3/4}/\bar{D}_{x,b,3/4}$	1.338/1.185	–	rear long. friction coeff. at load a/b
	$\bar{D}_{y,a,3/4}/\bar{D}_{y,b,3/4}$	1.421/1.271	–	rear lat. friction coeff. at load a/b
	$C_{x,3/4}/C_{y,3/4}$	1.949/1.949	–	rear long./lat. shape factor
vehicle	l_f/l_r	1.45/1.24	m	distance from COG to front/rear axle
	b_f/b_r	1.57/1.59	m	track width of front/rear axle
	b_v	1.85	m	vehicle chassis width
	h_v	0.34	m	COG height
	$r_{1/2}/r_{3/4}$	0.32/0.33	m	dyn. roll. radius of front/rear wheels
	m	2200.0	kg	total vehicle mass
	J_{zz}	3140.0	kgm^2	vehicle inertia around vertical axis
	A_{air}	2.21	m^2	cross-section area of vehicle
	$c_{\mathrm{air},x}, c_{\mathrm{air},y}$	0.36	–	longitudinal/lateral drag coefficient
	$c_{\mathrm{air},z,1/2}/c_{\mathrm{air},z,3/4}$	0.09/0.03	–	aerodyn. lift coeff. for front/rear axle
	$f_{\mathrm{roll},0}$	0.0031	–	rolling resistance coefficient
	τ_{acc}	0.03	s	time constants for acceleration low-pass
	ξ_{roll}	0.7	–	roll moment distribution factor

define the frame that is coupled with the corresponding wheel in its centre point P_k with $k \in \mathbb{K} = \{1, 2, 3, 4\}$. It is assumed that both wheels on an axle are deflected with the same angle: the front wheels with the front steering angle $\delta_f = \delta_1 = \delta_2$ and the rear wheels with the rear steering angle $\delta_r = \delta_3 = \delta_4$. Thus, the wheel coordinate frames are rotated by the corresponding wheel angle around the z^I-axis compared to the body coordinate frame. For a better differentiation, calligraphic symbols are used for the components of a vector in the corresponding wheel coordinate frame. The description of the curvilinear coordinates is given in Section 3.3.1. The parameters for the basic vehicle model are described and listed in Table 4.1.

The tire forces are described in the wheel coordinate frame $\mathbf{F}_k^{w_k} = \begin{bmatrix} \mathcal{F}_{x,k} & \mathcal{F}_{y,k} \end{bmatrix}^T$ and can be transformed into the body reference frame via

$$\mathbf{F}_k^B = \begin{bmatrix} F_{x,k} \\ F_{y,k} \end{bmatrix} = \mathbf{R}_z(-\delta_k)\, \mathbf{F}_k^{w_k} \quad \forall k \in \mathbb{K} \tag{4.1}$$

using the corresponding steering angle δ_k and the rotation matrix

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) \\ -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{4.2}$$

which describes a rotation around the z^I-axis with the angle $\gamma \in \mathbb{R}$. Newton's second law [181] is used to derive the differential equations describing the vehicle motion. These equations utilise following parameters: vehicle mass m, vehicle inertia around the vertical axis J_{zz}, distance from COG to front axle l_f or to rear axle l_r, track width of front axle b_f or rear axle b_r, dynamic rolling radius of the corresponding tire r_k and inertia of the corresponding wheel-unit J_k. The translational movement of the vehicle is described by the temporal development of the longitudinal vehicle velocity v_x and lateral vehicle velocity v_y at the COG:

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \frac{1}{m} \underbrace{\begin{bmatrix} \sum_{k=1}^4 F_{x,k} - F_{x,\mathrm{air}} \\ \sum_{k=1}^4 F_{y,k} - F_{y,\mathrm{air}} \end{bmatrix}}_{\coloneqq \begin{bmatrix} F_x & F_y \end{bmatrix}^T} + \begin{bmatrix} \dot{\psi}\, v_y \\ -\dot{\psi}\, v_x \end{bmatrix}. \tag{4.3a}$$

Therein $F_{x,\mathrm{air}}$ and $F_{y,\mathrm{air}}$ represent the longitudinal and lateral aerodynamic drag force in the body coordinate frame, respectively. The derivation of (4.3a) is demonstrated in Appendix A.1. The yaw motion of the vehicle is given by the differential equation

$$\ddot{\psi} = \frac{1}{J_{zz}} \left[(F_{y,1} + F_{y,2}) l_f - (F_{y,3} + F_{y,4}) l_r + (F_{x,2} - F_{x,1}) \frac{b_f}{2} + (F_{x,4} - F_{x,3}) \frac{b_r}{2} \right] \tag{4.3b}$$

for the yaw rate $\dot{\psi}$. The rotational speed ω_k of the individual wheels follows the differential equation

$$\dot{\omega}_k = \frac{1}{J_k} \left(T_k - \mathcal{F}_{x,k}\, r_k + \mathcal{T}_{y,\mathrm{roll},k} \right) \quad \forall k \in \mathbb{K}. \tag{4.3c}$$

Therein the torque applied to the wheel is represented by T_k and torques resulting from rolling resistance are considered via $\mathcal{T}_{y,\mathrm{roll},k}$. The missing equations for the aerodynamic forces, rolling resistance and tire forces used in the motion equations (4.3) are given in the subsequent paragraphs.
For completeness, the side-slip angle, which often serves as indicator for the stability of the driving situation, is computed via

$$\beta = \arctan\left(\frac{v_y}{v_x} \right) \tag{4.4}$$

whereas large side-slip angles occur when the vehicle is drifting. On racetracks, drifting is generally avoided since it strains the tire and results in fast tire wear.
Due to air resistance, the chassis of the vehicle is exposed to aerodynamic forces that depend on the total velocity of the vehicle

$$v_{\text{tot}} = \sqrt{v_x^2 + v_y^2}. \tag{4.5}$$

For simplicity, this thesis assumes windless conditions and computes aerodynamic drag and lift forces using following parameters: air density ρ_{air}, longitudinal and lateral drag coefficient $c_{\text{air},x}$ and $c_{\text{air},y}$, aerodynamic lift coefficient $c_{\text{air},z,k}$ and cross-section area of the vehicle A_{air}. Aerodynamic drag forces are assumed to apply to the COG of the vehicle and are computed in the longitudinal and lateral direction via

$$F_{x,\text{air}} = \frac{1}{2}\,\rho_{\text{air}}\,c_{\text{air},x}\,A_{\text{air}}\,v_{\text{tot}}\,v_x \quad \text{and} \quad F_{y,\text{air}} = \frac{1}{2}\,\rho_{\text{air}}\,c_{\text{air},y}\,A_{\text{air}}\,v_{\text{tot}}\,v_y, \tag{4.6a}$$

respectively [181]. For simplicity, the same cross-section area is used for the longitudinal and lateral air force. This thesis presumes that the aerodynamic lift forces apply on the centre of the corresponding vehicle axle resulting in the aerodynamic lift force

$$F_{z,\text{air},k} = \frac{1}{4}\,\rho_{\text{air}}\,c_{\text{air},z,k}\,A_{\text{air}}\,v_{\text{tot}}^2 \quad \forall k \in \mathbb{K} \tag{4.6b}$$

for the individual wheels [181].
As previously mentioned, the vehicle is subject to rolling resistance forces that depend on the wheel load of the corresponding wheel $F_{z,k}$ and result in torques opposed to the rolling direction given by

$$\mathcal{T}_{y,\text{roll},k} = -f_{\text{roll},0}\,F_{z,k}\,r_k \quad \forall k \in \mathbb{K} \tag{4.7}$$

with the rolling resistance coefficient $f_{\text{roll},0}$ [181].
In order to simplify the computation of wheel loads, a rigid connection between vehicle chassis and wheels is assumed. Thus, dynamic rolling, pitching and heaving motions are neglected discarding dynamic wheel loads. Since conditions on racetracks are rather QSS, this simplification is legitimate. However, the influence of the neglected suspension dynamics is partly considered in the tire force parameters since these parameters are tuned to fit the overall vehicle model to measurement data. The vertical forces acting on the tires are computed by applying balance equations for forces and momentums considering the accelerations acting on the COG. Using the gravitational acceleration g, the static wheel loads are given by

$$F_{z,s,1} = \frac{1}{2}\,m\,g\,\frac{l_r}{l_f + l_r} = F_{z,s,2} \quad \text{and} \quad F_{z,s,3} = \frac{1}{2}\,m\,g\,\frac{l_f}{l_f + l_r} = F_{z,s,4}. \tag{4.8a}$$

Considering the COG height of the vehicle h_v, wheel load changes due to longitudinal acceleration are included according to

$$F_{z,a_x,1} = -\frac{1}{2}\,m\,a_x\,\frac{h_v}{l_f + l_r} = F_{z,a_x,2} \quad \text{and} \quad F_{z,a_x,3} = \frac{1}{2}\,m\,a_x\,\frac{h_v}{l_f + l_r} = F_{z,a_x,4}. \tag{4.8b}$$

Assuming that the roll momentum $M_x = m\,a_y\,h_v$ is allocated via the roll moment distribution factor ξ_{roll}, the wheel loads on each side need to be modified with the terms

$$F_{z,a_y,1} = -\xi_{\text{roll}}\,\frac{M_x}{b_f} = -F_{z,a_y,2} \quad \text{and} \quad F_{z,a_y,3} = -(1-\xi_{\text{roll}})\,\frac{M_x}{b_r} = -F_{z,a_y,4}. \tag{4.8c}$$

Using equations (4.8a)-(4.8c) and the aerodynamic wheel loads (4.6b), the resulting tire loads are given by

$$F_{z,k} = F_{z,s,k} + F_{z,a_x,k} + F_{z,a_y,k} + F_{z,\text{air},k} \quad \forall k \in \mathbb{K}. \tag{4.8d}$$

The computation of the wheel loads (4.8b)-(4.8c) is based on the accelerations of the vehicle. Using the tire forces to compute these accelerations results in an algebraic loop. In the context of optimal control, this circular dependency yields a set of algebraic equations that needs to be solved within the NLP [158]. A different procedure is used in this thesis to simplify the OP. The algebraic loop is removed by considering delayed accelerations using first-order lag elements with time constant τ_{acc} [16]:

$$\dot{a}_x = \frac{1}{\tau_{\text{acc}}}\left(\frac{1}{m}F_x - a_x\right) \quad \text{and} \quad \dot{a}_y = \frac{1}{\tau_{\text{acc}}}\left(\frac{1}{m}F_y - a_y\right). \tag{4.9}$$

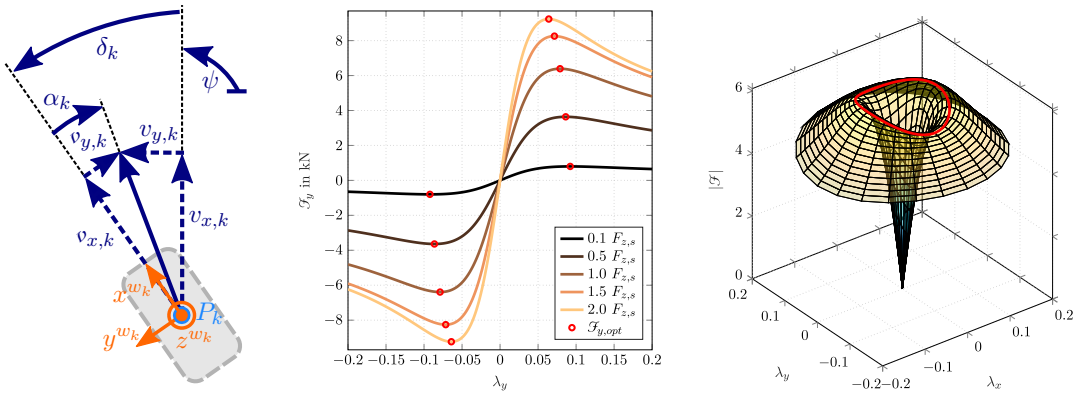
Besides avoiding additional equality constraints, this approach reduces couplings and nonlinearities, which eases the solution process of the OCP. Physically, the equations in (4.9) can be justified by the neglected suspension dynamics [16] and the fact that tire forces build up dynamically, which is commonly depicted via first-order lag elements with variable time constant for each individual tire force [192]. However, this would add eight additional state variables, making the solution of the OCP more challenging.
The computation of the tire forces requires the wheel slip, which characterises the relative speed between wheel and ground. Using Fig. 4.2a as graphical support, the velocities of the wheel centre points are computed below. Let $\mathbf{r}^B_{P_0P_k}$ be the position vector in the body coordinate frame pointing from the COG P_0 to the corresponding wheel centre point P_k. Considering the body coordinate frame rotates with the yaw rate $\dot{\psi}$ around the z^I-axis, the velocities of the wheel centre points given in the body or corresponding wheel coordinate frame are computed via

$$\mathbf{v}^B_{P_k} = \begin{bmatrix} v_{x,k} \\ v_{y,k} \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \mathbf{r}^B_{P_0P_k} \quad \text{and} \quad \mathbf{v}^{w_k}_{P_k} = \begin{bmatrix} v_{x,k} \\ v_{y,k} \end{bmatrix} = \mathbf{R}_z(\delta_k) \begin{bmatrix} v_{x,k} \\ v_{y,k} \end{bmatrix}, \tag{4.10}$$

respectively. The resulting quantities are given in Appendix A.2. Using these velocities, the longitudinal wheel slips are given by

$$\lambda_{x,k} = \frac{\omega_k r_k - v_{x,k}}{\max\left(|\omega_k r_k|, |v_{x,k}|\right)} \overset{\substack{\omega_k \geq 0 \\ v_{x,k} \geq 0}}{=} \frac{\omega_k r_k - v_{x,k}}{\max\left(\omega_k r_k, v_{x,k}\right)}. \tag{4.11a}$$

Thus, a symmetrical longitudinal slip relation is chosen by incorporating the case-distinction in the denominator of (4.11a). This enables using symmetrical tire force



(a) Wheel slip angle relation with positively drawn angles.

(b) Lateral tire force for $\lambda_x = 0$, $\mu = 1$ (dry asphalt) and various wheel loads.

(c) Total tire force for nominal load $F_{z,s}$ and relation to friction ellipse (red).

Figure 4.2: Combined slip tire force model.

curves and improves the tire model quality. Good results in the field of traction control [168] confirm the benefit of this slip equation. Furthermore, the lateral slips $\lambda_{y,k}$ are approximated by the wheel slip angles α_k following

$$\lambda_{y,k} \approx \alpha_k = \delta_k - \arctan\left(\frac{v_{y,k}}{v_{x,k}}\right). \tag{4.11b}$$

A simplified version of the Pacejka tire model for combined slip [149] is employed to compute the tire forces based on the longitudinal and lateral slip. The influence of tire pressure and tire temperature is neglected. For clarity, the following equations only depict the longitudinal tire forces but lateral tire forces are constructed analogously. The tire parameters differ for the front and rear wheels as well as in the longitudinal and lateral direction yielding an anisotropic behaviour. For simplicity, the load-dependent tire slip optimum and the nonlinear wheel load degressivity are approximated using linear relations between two characteristic wheel loads $0 < F_{z,a} < F_{z,b}$ according to

$$\lambda_{x,\max,k} = \frac{\bar{\lambda}_{x,b,k} - \bar{\lambda}_{x,a,k}}{\bar{F}_{z,b} - \bar{F}_{z,a}}\left(F_{z,k} - \bar{F}_{z,a}\right) + \bar{\lambda}_{x,a,k} \quad \text{and} \tag{4.12a}$$

$$D_{x,k} = \frac{\bar{D}_{x,b,k} - \bar{D}_{x,a,k}}{\bar{F}_{z,b} - \bar{F}_{z,a}}\left(F_{z,k} - \bar{F}_{z,a}\right) + \bar{D}_{x,a,k}, \tag{4.12b}$$

respectively [95]. Therein the values $\bar{(\cdot)}$ represent fixed tire parameters. The impact of (4.12a) and (4.12b) is depicted in Fig. 4.2b. Both effects strongly influence the vehicle behaviour particularly during cornering because of the lateral wheel load transfer due to the lateral acceleration (4.8c). Relation (4.12a) and its counterpart for lateral tire forces shift the optimal longitudinal and lateral slips $\lambda^\star_{x,k}$ and $\lambda^\star_{y,k}$ towards smaller values as the load increases. Thus, different optimal slips are present between the left and right wheel

of an axle during cornering. Reaching the optimal slips on both sides simultaneously generally requires lateral torque allocation. By including (4.12b), the tire forces increase degressively with rising wheel load. Thus, the load discrepancy between the left and right wheel during cornering reduces the total wheel force potential of an axle. The normalised slips and the combined slip coefficient are characterised by

$$\lambda_{x,n,k} = \frac{\lambda_{x,k}}{\lambda_{x,\max,k}} \quad \text{and} \quad \lambda_{\text{tot},k} = \sqrt{\lambda_{x,n,k}^2 + \lambda_{y,n,k}^2}, \tag{4.12c}$$

respectively. The tire force shape curve, which remains in the range $[-1, 1]$, is given by

$$\mathcal{F}_{x,\text{shape},k} = \frac{\lambda_{x,n,k}}{\lambda_{\text{tot},k}} \sin\Big(C_{x,k} \arctan(B_{x,k}\, \lambda_{\text{tot},k})\Big) \quad \text{with} \quad B_{x,k} = \frac{\pi}{2 \arctan(C_{x,k})}. \tag{4.12d}$$

Using (4.8d),(4.12b) and (4.12d) together with the road friction coefficient μ yields the longitudinal component of the tire force

$$\mathcal{F}_{x,k} = \mu\, F_{z,k}\, D_{x,k}\, \mathcal{F}_{x,\text{shape},k}. \tag{4.12e}$$

The total tire force value is determined by the longitudinal and lateral tire force component according to $|\mathcal{F}_k| = \sqrt{\mathcal{F}_{x,k}^2 + \mathcal{F}_{y,k}^2}$. The total tire force peaks at the friction ellipse illustrated by the red line in Fig. 4.2c, which can be represented by the elliptical equation

$$\left(\frac{\lambda_{x,k}}{\lambda_{x,k}^\star}\right)^2 + \left(\frac{\lambda_{y,k}}{\lambda_{y,k}^\star}\right)^2 = 1 \quad \text{with} \tag{4.13a}$$

$$\lambda_{x,k}^\star = \underbrace{\frac{1}{B_{x,k}} \tan\left(\frac{\pi}{2C_{x,k}}\right)}_{=:\lambda_{x,n,k}^\star} \lambda_{x,\max,k} \quad \text{and} \quad \lambda_{y,k}^\star = \underbrace{\frac{1}{B_{y,k}} \tan\left(\frac{\pi}{2C_{y,k}}\right)}_{=:\lambda_{y,n,k}^\star} \lambda_{y,\max,k}. \tag{4.13b}$$

It can be shown that the region within the friction ellipse (4.13a) depicts the stable region of the wheels [178, p.32][105, p.16].

4.1.2 Model Validation

The two-track vehicle model presented in the previous section has been experimentally validated in a test vehicle. For this purpose, the test vehicle was equipped with an Automotive Dynamic Motion Analyzer (ADMA) from the company *GeneSys*. This measurement device contains a highly precise inertial measurement unit whose data is combined with differential GPS data via data fusion. This provides high quality measurement data of accelerations and speeds for the three axes of the body coordinate frame as well as the position of the vehicle. Furthermore, the ADMA computes the angles and angular rates around the body frame axes and provides the side-slip angle. The rotational wheel speeds are measured via the standard incremental encoders installed at the wheels. Torque sensors have been used to sense the torque applied to the shaft of the wheels and brake torques have been computed using brake pressure sensors. This

enables the computation of the wheel torques, which serve, together with the measured front steering angle δ_f, as model inputs.
The model is tuned to depict the overall vehicle behaviour on racetracks with the highest possible accuracy. For the parameter fitting, the vehicle motion has been measured on a short racetrack. The tire parameters have been identified in an optimisation aiming at minimising a multi-objective error between model data and measurements of the test vehicle. This procedure partly covers non-modelled effects in the model. The decision variables of the OP are

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_f \\ \mathbf{p}_r \end{bmatrix} \quad \text{with} \tag{4.14a}$$

$$\mathbf{p}_j = \begin{bmatrix} \bar{\lambda}_{x,a,j} & \bar{\lambda}_{x,b,j} & \bar{\lambda}_{y,a,j} & \bar{\lambda}_{y,b,j} & \bar{D}_{x,a,j} & \bar{D}_{x,b,j} & \bar{D}_{y,a,j} & \bar{D}_{y,b,j} & C_{x,j} & C_{y,j} \end{bmatrix} \tag{4.14b}$$

whereas $\mathbf{p}_f$ and $\mathbf{p}_r$ represent the tire parameters for the front and rear wheels, respectively. Subsequently, the measured variables are denoted by an additional index: $\mathbf{y}_m$ represents the n_{meas} measured data points and $\mathbf{y}$ the data points generated by simulating the model via numerical integration. The multi-objective OP used to fit the model data is given by

$$\min_{\mathbf{P}} \quad c_{\text{acc}}\big(J_{\text{fit}}(\mathbf{a}_x, \mathbf{a}_{x,m}) + J_{\text{fit}}(\mathbf{a}_y, \mathbf{a}_{y,m})\big) + c_{\dot{\psi}} J_{\text{fit}}(\dot{\boldsymbol{\psi}}, \dot{\boldsymbol{\psi}}_m) + c_\omega \sum_{j=1}^{4} J_{\text{fit}}(\boldsymbol{\omega}_j, \boldsymbol{\omega}_{j,m}) \tag{4.15a}$$

$$\text{with } J_{\text{fit}}(\mathbf{y}, \mathbf{y}_m) \coloneqq \frac{1}{n_{\text{meas}}} \sum_{k=0}^{n_{\text{meas}}} \left(\frac{y_k - y_{m,k}}{\|\mathbf{y}_m\|_{\max}} \right)^2 \tag{4.15b}$$

$$\text{s.t.} \quad \begin{aligned} \mathbf{P} &\geq \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.4 & 1.4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.4 & 1.4 \end{bmatrix} \\ \mathbf{P} &\leq \begin{bmatrix} 0.5 & 0.5 & 0.524 & 0.524 & 2.0 & 2.0 & 2.0 & 2.0 & 1.95 & 1.95 \\ 0.5 & 0.5 & 0.524 & 0.524 & 2.0 & 2.0 & 2.0 & 2.0 & 1.95 & 1.95 \end{bmatrix} \end{aligned}, \tag{4.15c}$$

$$\begin{aligned} &\overline{\lambda}_{x,b,f} < \overline{\lambda}_{x,a,f},\ \overline{\lambda}_{y,b,f} < \overline{\lambda}_{y,a,f},\ \overline{D}_{x,b,f} < \overline{D}_{x,a,f},\ \overline{D}_{y,b,f} < \overline{D}_{y,a,f} \\ &\overline{\lambda}_{x,b,r} < \overline{\lambda}_{x,a,r},\ \overline{\lambda}_{y,b,r} < \overline{\lambda}_{y,a,r},\ \overline{D}_{x,b,r} < \overline{D}_{x,a,r},\ \overline{D}_{y,b,r} < \overline{D}_{y,a,r} \end{aligned}, \tag{4.15d}$$

$$\begin{aligned} &\left(\lambda^\star_{x,f}\Big|_{F_{z,s,f}} - \lambda^\star_{y,f}\Big|_{F_{z,s,f}} \right)^2 \leq 0.02^2,\ \left(\lambda^\star_{x,r}\Big|_{F_{z,s,r}} - \lambda^\star_{y,r}\Big|_{F_{z,s,r}} \right)^2 \leq 0.02^2 \\ &\left(\lambda^\star_{x,f}\Big|_{F_{z,s,f}} - \lambda^\star_{x,r}\Big|_{F_{z,s,r}} \right)^2 \leq 0.02^2,\ \left(\lambda^\star_{y,f}\Big|_{F_{z,s,f}} - \lambda^\star_{y,r}\Big|_{F_{z,s,r}} \right)^2 \leq 0.02^2 \end{aligned}, \tag{4.15e}$$

$$\begin{aligned} &\left(\mathcal{F}_{x,f}\Big|_{\lambda^\star_{x,f}, F_{z,s,f}} - \mathcal{F}_{y,f}\Big|_{\lambda^\star_{y,f}, F_{z,s,f}} \right)^2 \leq 500^2 \\ &\left(\mathcal{F}_{x,r}\Big|_{\lambda^\star_{x,r}, F_{z,s,r}} - \mathcal{F}_{y,r}\Big|_{\lambda^\star_{y,r}, F_{z,s,r}} \right)^2 \leq 500^2 \end{aligned} \tag{4.15f}$$

using the objective weights $c_{\text{acc}} = 0.4$, $c_{\dot{\psi}} = 0.4$ and $c_\omega = 0.2$. Therein the bounds (4.15c) are used to limit the search space to a plausible range: The empirically selected values avoid tire forces that are excessively large or possess an implausible shape curve. Furthermore, the bounds prevent an exaggerated wheel load degressivity and avoid large

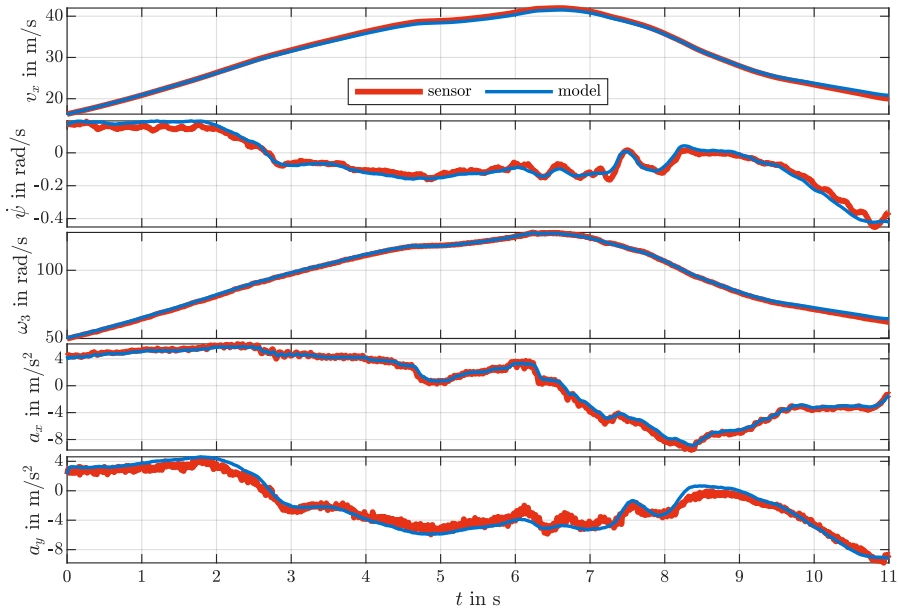


Figure 4.3: Comparison of measured signals and model simulation on a racetrack section.

slip optimum shifts over load. The inequalities in (4.15d) ensure that the dependencies in (4.12a) and (4.12b) are given in the right direction, thus fix the sign of the linear relations. Furthermore, the discrepancies between the individual optimal slip values at nominal loads are limited via (4.15e) to avoid excessive anisotropy and large differences between front and rear tires. Similarly, the discrepancy in maximum tire force at nominal loads between the longitudinal and lateral direction is limited in (4.15f) for both axles. The additional constraints are necessary to enforce plausible solutions since many solutions exist to the OP. Further constraints can be imposed to imprint a tendency. The OP has been solved via *MATLAB* using the solver *fminsearchcon* [41]. Within each iteration, the model data $\mathbf{y}$ is generated by feeding the measured wheel torques and steering angle to the model and applying numerical integration: A Runge-Kutta-4 solver with step time 0.1 seconds is employed. The tire parameters computed by the optimisation are listed in Table 4.1.
Selected state trajectories generated by the model simulation with the identified tire parameters are depicted together with corresponding sensor data in Fig. 4.3. The figure shows a combined slip situation for alternating turns under acceleration and deceleration with a velocity up to around $150\frac{\mathrm{km}}{\mathrm{h}}$. Although representing a highly nonlinear situation, the model depicts the measured trajectories with sufficient accuracy. The model is suitable for racetrack applications but fails to accurately depict situations with large side-slip angles, which occur for instance when the vehicle is drifting. However, a highly accurate representation of the system behaviour is not crucial for the following investigations since the absolute lap time value is not of primary interest: The thesis focusses on relative differences between individual vehicle configurations, which all employ the same two-track base model.

4.2 Model Augmentations for Electric Vehicles

The application considered in this chapter aims at analysing the control strategy for a BEV with four wheel-independent drive units, which is depicted in Fig. 4.4. Each drive unit is comprised of an electrical motor, an inverter and a single-speed reduction gear. The torque and rotational speed of the corresponding electric machine is represented by $T_{m,k}$ and $\omega_{m,k}$, respectively. The electrical motors are capable of generating positive and negative torques, which depict drive torques and motor brake torques, respectively. With $i_{g,k}$ and $\eta_{g,k}$ denoting the gear ratio and the gear efficiency of the corresponding powertrain unit, the wheel-based motor torques are given by

$$T_{w,k} = T_{m,k}\, i_{g,k}\, \eta_{g,k}^{\operatorname{sign}(T_{m,k})} \quad \forall k \in \mathbb{K}. \tag{4.16}$$

A sign function is used in (4.16) to capture the directional property of the mechanical efficiencies. An additionally installed conventional braking system enables supplementary friction brake torques $T_{br,k}$. Thus, the accumulated torque at the wheels

$$T_k = T_{w,k} + T_{br,k} \tag{4.17}$$

enters the differential equation for the wheel dynamics (4.3c). The rotational speeds of the wheels ω_k and the corresponding motor speeds $\omega_{m,k}$ can be converted into each other as follows:

$$\omega_{m,k} = \omega_k\, i_{g,k} \quad \forall k \in \mathbb{K}. \tag{4.18}$$

By differentiating between components at the input and the output of the gear box with $J_{w,\mathrm{in},k}$ and $J_{w,\mathrm{out},k}$, the inertia of the corresponding wheel-unit J_k in (4.3c) considers the gear ratio according to

$$J_k = J_{w,\mathrm{out},k} + J_{w,\mathrm{in},k}\, i_{g,k}^2 \quad \forall k \in \mathbb{K}. \tag{4.19}$$

In order to consider the essential battery pack dynamics, Section 4.2.1 presents the link between the mechanical and electrical parts of the vehicular system. A simplified modelling procedure to depict EOL in OCPs is presented in Section 4.2.2. The required model parameters are given in Table 4.2.

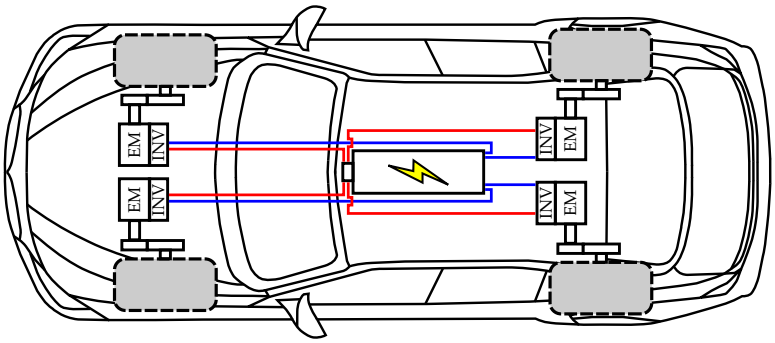


Figure 4.4: Powertrain components of battery electric vehicle with wheel-independent drive units: battery pack, inverters, electric machines and reduction gears.

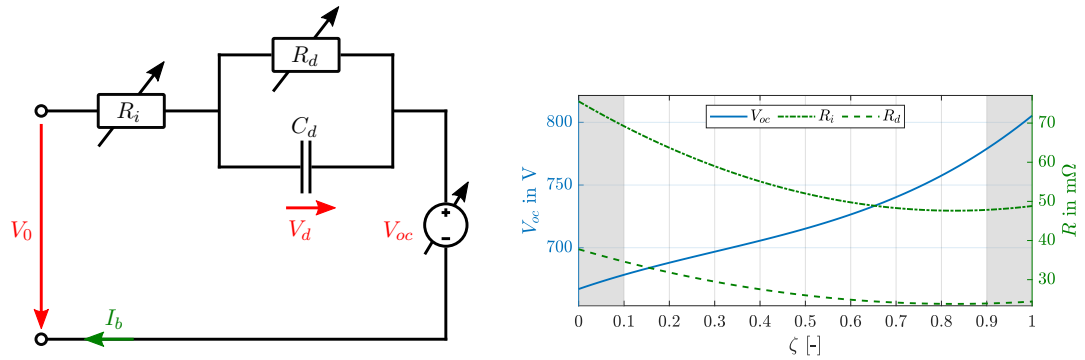
Table 4.2: Model parameters for electrical vehicle components.

Variable	Value	Unit	Description
$\eta_{g,1/2/3/4}$	0.98	–	efficiency factor of transmission
$J_{w,\mathrm{out},1/2}/J_{w,\mathrm{out},3/4}$	1.80/2.25	kgm^2	front/rear wheel inertia on output-side
$J_{w,\mathrm{in},1/2}, J_{w,\mathrm{in},3/4}$	0.08	kgm^2	front/rear wheel inertia on input-side
Q_b	60.0	Ah	capacity of battery pack
C_d	300.0	F	capacitance of battery pack
$p_{R_i,1}$	0.202	–	char. parameter for resistance R_i
$p_{R_i,2}$	-0.167	–	char. parameter for resistance R_i
$p_{R_i,3}$	0.048	–	char. parameter for resistance R_i
$p_{R_d,1}$	0.143	–	char. parameter for resistance R_d
$p_{R_d,2}$	-0.118	–	char. parameter for resistance R_d
$p_{R_d,3}$	0.024	–	char. parameter for resistance R_d
$p_{v,1}$	5.20	–	char. parameter for open circuit voltage
$p_{v,2}$	-1.57	–	char. parameter for open circuit voltage
$p_{v,3}$	696.94	–	char. parameter for open circuit voltage
$p_{v,4}$	16.62	–	char. parameter for open circuit voltage
$c_{m,1}$	$634.86 \cdot 10^{-4}$	[a]	char. parameter for motor power loss
$c_{m,2}$	$0.16 \cdot 10^{-4}$	[a]	char. parameter for motor power loss
$c_{i,1}$	$206.96 \cdot 10^{-4}$	[a]	char. parameter for inverter power loss
$c_{i,2}$	$42.62 \cdot 10^{-4}$	[a]	char. parameter for inverter power loss
$c_{i,3}$	$3.05 \cdot 10^{-4}$	[a]	char. parameter for inverter power loss
$c_{i,4}$	$0.11 \cdot 10^{-4}$	[a]	char. parameter for inverter power loss
$\tau_{\mathrm{bst},b}$	10.0	s	boosting time of battery pack
$\tau_{\mathrm{bst},m}$	5.0	s	boosting time of motors

[a] Parameter units in accordance with (4.20f) and (4.20g) using SI-units for torque and rotational speed.

4.2.1 Electrical Relations and Battery Pack Dynamics

Permanent-magnet synchronous-machines are assumed for the four electrical motors. Describing the behaviour of these machines generally requires differential equations for the stator currents and the rotational motor speeds, as depicted in [159]. However, these electromagnetic dynamics are generally fast compared to mechanical effects [125]. In order to reduce the number of differential equations and thus simplify the OP, the consideration of the motor dynamics is reduced to incorporating the rotor inertias of the motors into the wheel-unit inertias J_k in (4.19).
The battery pack is comprised of n_{par} parallel strings with n_{ser} battery cells interconnected in series. Neglecting long-term dynamic effects over lifetime, the main battery dynamics are captured using a Thevenin battery model [175] with the equivalent electric circuit depicted in Fig. 4.5a. The short-term dynamics of the battery pack are described using an RC-element with the resistance R_d and the capacitance C_d. The



(a) Equivalent electric circuit of battery pack in charging condition.

(b) Shape curves for open circuit voltage and resistances; Recommended working zone: $\zeta \in [0.1, 0.9]$.

Figure 4.5: Thevenin battery model with parameters varying over state of charge.

internal resistance, battery pack current, terminal voltage and open circuit voltage is represented by R_i, I_b, V_0 and V_{oc}, respectively. Both resistances as well as the open circuit voltage depend on the battery pack state of charge (SOC) ζ. Generally, the working zone for the SOC is constrained to a certain range to preserve durability and decelerate degradation of the battery pack. In this thesis, the recommended range for the SOC is assumed to be $\zeta \in [\underline{\zeta}_r, \overline{\zeta}_r] = [0.1, 0.9]$. In order to provide sufficiently smooth equations for the derivative-based NLP solver, the parameter dependencies are described via smooth analytic polynomials depicted in Fig. 4.5b, which have been introduced in our publication [183]. Compared to [34], the order of the polynomial describing the open circuit voltage is reduced. Furthermore, polynomials for the approximation of the resistance relations are also presented. Generally, the parameters of the battery pack model also differ over temperature. However, the vehicle is assumed to be equipped with an efficient thermal management system that is capable of keeping the temperature of the electric parts within a desired operating region [99]. In this region, the parameter changes due to temperature variation are small and thus neglected. Over the course of one lap, this simplifying assumption has been verified to be possible via experiments with a test vehicle. Thus, thermic dependencies of the electrical components are neglected for the sake of simplicity improving solvability of the OP. The table-based implementation of the battery model has been validated in [127].
Assuming for simplicity that the resistance values for charging and discharging are the same, the electrical resistance curves are approximated by

$$R_i = \frac{n_{\text{ser}}}{n_{\text{par}}} R_{i,\text{cell}} = R_{\text{shape}}(\mathbf{p}_{R_i}) \quad \text{and} \quad R_d = \frac{n_{\text{ser}}}{n_{\text{par}}} R_{d,\text{cell}} = R_{\text{shape}}(\mathbf{p}_{R_d}) \tag{4.20a}$$

$$\text{with} \quad R_{\text{shape}}(\mathbf{p}_R) = (p_{R,1}\,\zeta + p_{R,2})^2 + p_{R,3} \quad \text{and} \quad \mathbf{p}_R = \begin{bmatrix} p_{R,1} & p_{R,2} & p_{R,3} \end{bmatrix}. \tag{4.20b}$$

The open circuit voltage is approximated via

$$V_{oc} = n_{\text{ser}} V_{oc,\text{cell}} = (p_{v,1}\,\zeta + p_{v,2})^3 + p_{v,3} + p_{v,4}\,(p_{v,1}\zeta + p_{v,2})\,. \tag{4.20c}$$

Applying Kirchhoff's laws [102] yields the differential equation for the capacitor voltage

$$\dot{V}_d = \frac{1}{C_d}\left(I_b - \frac{V_d}{R_d}\right) \quad \text{with} \quad C_d = \frac{n_{\text{par}}}{n_{\text{ser}}} C_{d,\text{cell}} \tag{4.20d}$$

and enables the computation of the terminal voltage

$$V_0 = R_i I_b + V_d + V_{oc}. \tag{4.20e}$$

The efficiencies of the electric motors and inverters are considered via the power losses $P_{l_{m,k}}$ and $P_{l_{i,k}}$, respectively. These losses can be approximated via power loss maps depending on the motor torque and motor speed, exemplarily depicted for the electrical machines in Fig. 4.6. The combination of various loss terms $T_m^{n_T}\omega_m^{n_\omega}$ of different orders n_T and n_ω enables the depiction of arbitrary power loss maps [128]. Although specific loss terms can be motivated physically for various machine types, this thesis pursues the pragmatic approach of choosing loss terms aiming at minimising the error to measured power loss maps. Increasing the number of the individual loss terms generally improves accuracy [129] however complicates the OP due to additional nonlinearities. The power losses also depend on the voltage applied to the inverter as well as the temperature of the machine components [182]. As previously mentioned, this thesis assumes an elaborate thermic management system and thus neglects temperature dependencies. In order to reduce complicated equations, the loss maps are assumed to be constant over voltage. Power loss maps based on measurements with $V_0 = 740\text{V}$ have been used as reference to identify the coefficients for the loss terms

$$P_{l_{m,k}} = c_{m,1}\, T_{m,k}^2 + c_{m,2}\sqrt{T_{m,k}^2 + \varepsilon_{T_m}}\,\omega_{m,k}^2 \quad \forall k \in \mathbb{K} \quad \text{and} \tag{4.20f}$$

$$P_{l_{i,k}} = c_{i,1}\sqrt{T_{m,k}^2 + \varepsilon_{T_m}}\,\omega_{m,k} + c_{i,2}\, T_{m,k}^2 + c_{i,3}\,\omega_{m,k}^2 + c_{i,4}\, T_{m,k}^2\,\omega_{m,k} \quad \forall k \in \mathbb{K} \tag{4.20g}$$

with $P_{l_{m,k}}, P_{l_{i,k}} \geq 0$ and $\omega_k \geq 0$. This voltage value corresponds approximately to the mean value of the terminal voltage trajectory for the first lap with initial SOC $\zeta = 0.9$.

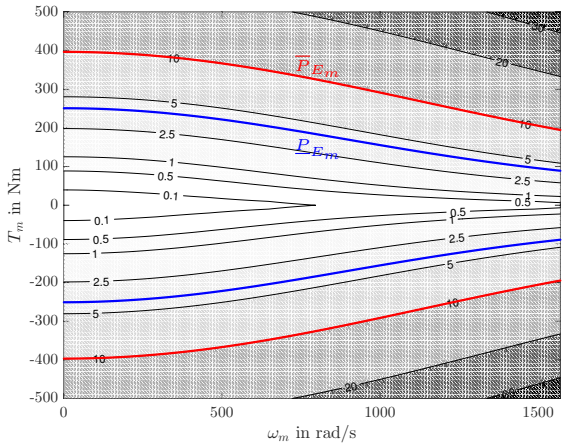


Figure 4.6: Motor power loss $P_{l,m}$ in kW over motor torque T_m and motor speed ω_m. — continuous operation boundary $\underline{P}_{E_m}$. — auxiliary boundary for temporary operation $\overline{P}_{E_m}$.

According to the power balance equation, the total electrical power P_{el} at the battery pack terminal must equal the sum of the power demands of each drive unit P_{el_k}:

$$P_{el} = \sum_{k=1}^{4} P_{el_k} = \sum_{k=1}^{4} -(T_{m,k}\,\omega_{m,k} + P_{l_{m,k}} + P_{l_{i,k}}) \stackrel{!}{=} V_0 I_b. \tag{4.20h}$$

Inserting (4.20e) in (4.20h), applying the quadratic formula and selecting the solution that enables positive currents and provides zero standby currents for $T_{m,k} = 0 = \omega_{m,k} \forall k \in \mathbb{K}$ yields

$$I_b = \frac{1}{2R_i}\Big(-\Sigma_V + \sqrt{\Sigma_V^2 + 4R_i P_{el}}\Big) \quad \text{and} \quad V_0 = \frac{1}{2}\Big(\Sigma_V + \sqrt{\Sigma_V^2 + 4R_i P_{el}}\Big) \tag{4.20i}$$

with $\Sigma_V \coloneqq V_d + V_{oc}$. In order to show that the discriminant $D_b = \Sigma_V^2 + 4R_i P_{el}$ in (4.20i) takes only positive values, the total electrical power

$$P_{el} = V_0\, I_b \stackrel{(4.20e)}{=} R_i I_b^2 + (V_d + V_{oc}) I_b \tag{4.20j}$$

is derived with respect to the battery pack current and set to zero, which identifies the extreme value:

$$\frac{\partial P_{el}}{\partial I_b} = 2R_i I_b + (V_d + V_{oc}) \stackrel{!}{=} 0 \quad \Rightarrow \quad I_{b,\lim} = -\frac{V_d + V_{oc}}{2R_i}. \tag{4.20k}$$

Inserting (4.20k) into (4.20j) yields the limit value for the total electrical power, for which the discriminant in (4.20i) is zero:

$$P_{el,\lim} = -\frac{(V_d + V_{oc})^2}{4R_i} \quad \Rightarrow \quad D_{b,\lim} = \Sigma_V^2 + 4R_i P_{el,\lim} = 0 \quad \Rightarrow \quad D_b \geq 0. \tag{4.20l}$$

Thus, the discriminant D_b stays nonnegative enabling the use of the formulas in (4.20i). Finally, the dynamic behaviour of the SOC is given by the battery pack capacity Q_b using the Coulomb counting method [34]:

$$\dot{\zeta} = \frac{I_b}{Q_b}. \tag{4.20m}$$

Using this sign convention, drive torques $T_{m,k} > 0$ draw current from the battery pack via negative motor currents and motor brake torques $T_{m,k} < 0$ charge the battery pack via positive motor currents. Hence, the battery pack is being charged for positive battery pack currents $I_b > 0$, which will be called "recuperation" in the following sections[7].

[7] In fact, energy is recuperated when any electric machine is used for motor braking. For simplicity, the term is used in this thesis when all machines in total recuperate energy.

4.2.2 Electrical Overloading

In order to provide increased power, electrical components can be temporarily overloaded, however this raises heat generation in the component. Since the electrical components can only withstand certain temperatures, this boosting measure depends on the heat properties of the component. Sophisticated thermic models could be used to control this feature without damaging the component [79]. However, identifying the parameters for thermic models generally requires costly test benches and an appropriate sensor setup. Moreover, adding further, possibly nonlinear and complicated, differential equations increases the size and complexity of the OP deteriorating its solvability. In this thesis, vehicles with four wheel-independent drives are analysed. An exact overload model would require implementing temperature models for each electrical component, which would result in a high number of additional differential equations. Solving the corresponding OCP would take considerably longer. Thus, this thesis only analyses the macroscopic relations regarding overloading, which suffices to identify key differences between several vehicle setups while keeping a reasonable computation time. Neglecting the internal states of the electrical components, the simplified approach presented in our publication [183] is employed. Overloading is limited by the duration and overload extent via a tank model in combination with power loss flows and appropriate constraints. The heat generation is approximated via power losses representing the overload extent. Using the power losses (4.20f) for the electric machines, the main power loss in the battery pack is modelled as Joule heating. Assuming ideal resistors with complete conversion of the power into heat, the battery pack power loss is computed by

$$P_{l_b} = P_{l_{R_i}} + P_{l_{R_d}} = R_i I_b^2 + \frac{V_d^2}{R_d}. \tag{4.21}$$

The threshold temperature up to which electrical components can be operated continuously is represented by a power limit $\underline{P}_E$. Fig. 4.6 exemplarily depicts the corresponding limit $\underline{P}_{E_m}$ for an electrical machine. The tolerable heat energy deposition into the component, which characterises the overload capability, is represented by a predefined energy storage E of capacity $\overline{E}$. The power losses P_l depict the heat power fed into the component and the dissipated heat power is denoted by P_d. Thus, the total power flow is given by

$$P_E = P_l - P_d. \tag{4.22}$$

For simplicity, it is assumed that the dissipated heat power is constant and considered in the corresponding power limits $\underline{P}_E$ yielding $P_d \equiv 0$[8]. When the total power flow exceeds the power limit $P_E > \underline{P}_E$, the energy storage, which represents the boost capacity, is drained. Analogously, the storage is filled for total power flows below the limit $P_E < \underline{P}_E$.

[8] The dissipated heat power can depend on aerodynamic relations, which could be considered via velocity-dependent terms. Furthermore, the heat dissipation can be modelled using temperature differences between the component and its surroundings. For this purpose, simplified thermic models represent an alternative approach [82]. However, depending on the number of temperature nodes, this method can require additional differential equations.

For a unified modelling of these relations for storages of various capacity $\overline{E}$, the normalised flows running into the overload tank for the battery pack $E_b \in [0,1]$ and each motor $E_{m,k} \in [0,1]$ are defined as

$$\Delta P_{E_b} = \frac{\underline{P}_{E_b} - P_{E_b}}{\overline{E}_b} \quad \text{and} \quad \Delta P_{E_{m,k}} = \frac{\underline{P}_{E_{m,k}} - P_{E_{m,k}}}{\overline{E}_{m,k}} \quad \forall k \in \mathbb{K}. \tag{4.23}$$

Thus, the magnitude of the power losses (4.20f) and (4.21) indicate the extent of the overload and affect the flow rate (4.23), which determines the possible overload duration. Within the scope of this thesis, the term "boosting" represents a negative power flow $\Delta P_E < 0$ and "chilling" refers to positive flows $\Delta P_E > 0$. Inflows into the overload storage are only possible until the tank is full. This is implemented using the switching function (3.5):

$$\dot{E}_b = \min(\Delta P_{E_b}, 0) + \max(\Delta P_{E_b}, 0)\, f_{\mathrm{sw}}(E_b, 1) \quad \text{and} \tag{4.24a}$$

$$\dot{E}_{m,k} = \min(\Delta P_{E_{m,k}}, 0) + \max(\Delta P_{E_{m,k}}, 0)\, f_{\mathrm{sw}}(E_{m,k}, 1) \quad \forall k \in \mathbb{K}. \tag{4.24b}$$

Switching off the inflow is necessary to avoid infeasible sets that can occur for instance at the race start: Small initial power losses $P_l = P_E < \underline{P}_E$ result in positive power flows $\Delta P_E > 0$, but the limited energy storage $E \leq 1$ is initially full with $E = 1$. Additional states for inverter overloading are omitted under the assumption that the thermic bounds of the inverters are much higher than the thermic bounds of the motors. Exaggerated EOL reduces the efficiency of the electrical components over lifetime. It is assumed that proper overload limits are chosen that either exclude such long-term effects or these effects become negligible for the desired operation period of the vehicle. In order to model the power loss flows via (4.23), the corresponding power limit $\underline{P}_E$ and energy tank capacity $\overline{E}$ must be available. The following paragraph presents a method to approximate these values if they are unknown. Manufacturers often provide estimated regions for continuous and temporary operation together with a time span τ_{bst} for the operation within this boosted power area. These regions are represented by the lower and upper power limit value $\underline{P}$ and $\overline{P}$, respectively. For the battery pack, these power limits are denoted by $\underline{P}_b$ and $\overline{P}_b$. Using these power limits, the boundary values for the power losses are approximated by the maximum possible power loss, which is identified via the SOP

$$\min_{\zeta,\, V_d} \quad -P_{l_b} = -\left[R_i \left(\frac{P_{el}^\star}{V_0^\star}\right)^2 + \frac{V_d^2}{R_d}\right] \quad \text{with} \tag{4.25a}$$

$$V_0^\star = \frac{1}{2}\left[(V_d + V_{oc}) + \sqrt{(V_d + V_{oc})^2 + 4R_i P_{el}^\star}\right] \tag{4.25b}$$

$$\text{s.t.} \quad 0.1 \leq \zeta \leq 0.9, \tag{4.25c}$$

$$-15\mathrm{V} \leq V_d \leq 15\mathrm{V}. \tag{4.25d}$$

Adequate bounds are set for the SOC ζ and capacitor voltage V_d in (4.25c) and (4.25d) to gain realistic values. The terminal voltage (4.25b) is the result of inserting $I_b = P_{el}^\star / V_0^\star$

in (4.20e) and selecting the solution with $V_0^\star \geq 0$. Solving (4.25) using $P_{el}^\star = \underline{P}_b$ provides a solution $(\zeta^\star, V_d^\star)$. The SOC $\zeta^\star$ can be used to compute the resistances via (4.20a) and the open circuit voltage following (4.20c). Then, the battery pack current is computable by (4.20i) with $V_d^\star$ and $P_{el}^\star = \underline{P}_b$. Finally, the corresponding power loss, which represents the power limit $\underline{P}_{E_b}$, is given by (4.21). Analogously, using $P_{el}^\star = \overline{P}_b$ delivers the auxiliary upper bound $\overline{P}_{E_b}$. It is possible to compute the power limit values for the electric machines similarly: Prescribing a certain power, the maximum possible power loss value for given torque characteristics and power loss maps (4.20f) can be used as limit value. However, it is presumed for now that the power limits $\underline{P}_{E_{m,k}}$ and $\overline{P}_{E_{m,k}}$, which are depicted in Fig. 4.6, are known.
Using the power limits, the overload tank capacities are approximated by

$$\overline{E}_b = (\overline{P}_{E_b} - \underline{P}_{E_b})\frac{\tau_{E_b}}{\mathrm{s}} \quad \text{and} \quad \overline{E}_{m,k} = (\overline{P}_{E_{m,k}} - \underline{P}_{E_{m,k}})\frac{\tau_{E_m}}{\mathrm{s}} \quad \forall k \in \mathbb{K} \tag{4.26}$$

for the battery pack and each motor, respectively.

4.3 Model Smoothing

Before formulating the OP, the model equations listed in Section 4.1 and Section 4.2 must be smoothed according to Section 3.2 to ensure the smoothness requirements of the NLP solver. The discontinuity due to the maximum function included in the longitudinal slips (4.11a) is removed using (3.3a) yielding

$$\lambda_{x,k} \approx \frac{\omega_k r_k - v_{x,k}}{\frac{1}{2}\Big((\omega_k r_k + v_{x,k}) + \sqrt{(\omega_k r_k - v_{x,k})^2 + \varepsilon_{\lambda,x}}\Big)} \quad \forall k \in \mathbb{K}. \tag{4.27a}$$

Using the approximate absolute value function (3.2), the total slips (4.12c) are replaced by

$$\lambda_{\mathrm{tot},k} \approx \sqrt{\lambda_{x,n,k}^2 + \lambda_{y,n,k}^2 + \varepsilon_{\lambda,\mathrm{tot}}} \quad \forall k \in \mathbb{K} \tag{4.27b}$$

avoiding nonsmoothness for $\lambda_{x,n,k} = 0 = \lambda_{y,n,k}$ and a division by zero in (4.12d). Another discontinuity is given for $v_{\mathrm{tot}} = 0$ in (4.5), which is introduced into the system by the aerodynamic drag and lift forces in (4.6a) and (4.6b). However, this discontinuity is avoided by enforcing positive longitudinal speeds via constraints yielding $v_{\mathrm{tot}} > 0$, which enables keeping the equations simple. Since driving backwards on racetracks is not reasonable, this arrangement is legitimate.
Using (3.4), the directionality of the mechanical efficiencies in (4.16) is approximated by

$$T_{w,k} \approx T_{m,k}\, i_{g,k}\, \eta_{g,k}^{\tanh\left(T_{m,k}/\varepsilon_\eta\right)} \quad \forall k \in \mathbb{K}. \tag{4.28}$$

In (4.20f) and (4.20g), the absolute value of the individual motor torques is approximated using (3.2). Since the identification of the corresponding parameters listed in Table 4.2 was performed using the smoothed equations to reduce errors, the smoothed equations have been directly specified in Section 4.2.1.

Table 4.3: Smoothing parameters for vehicular optimal control problem.

Variable	Value	Unit	Description
$\varepsilon_{\lambda,x}$	10^{-6}	–	smoothing factor for longitudinal slips
$\varepsilon_{\lambda,\mathrm{tot}}$	10^{-8}	–	safety factor for singularity in total slips
ε_{η}	1.0	–	smoothing factor for directionality of gear efficiency
ε_{T_m}	10^{2}	–	smoothing factor for power loss maps
ε_{P_l}	10^{-6}	–	smoothing factor for power loss
c_E	10^{2}	–	smoothing factor for switching function
ε_E	$5.0 \cdot 10^{-2}$	–	shift factor for switching function

The differential equations describing the overloading of the battery pack and the electrical machines in (4.24) are smoothed using the approximations (3.3a), (3.3b) and (3.5). This results in the smooth equations

$$\min(\Delta P_E, 0) \approx \frac{1}{2}\left(\Delta P_E - \sqrt{\Delta P_E^2 + \varepsilon_{P_l}}\right), \tag{4.29a}$$

$$\max(\Delta P_E, 0) \approx \frac{1}{2}\left(\Delta P_E + \sqrt{\Delta P_E^2 + \varepsilon_{P_l}}\right) \quad \text{and} \tag{4.29b}$$

$$f_{\mathrm{sw}}(E, 1) \approx -\frac{1}{2}\Big[\tanh\Big(c_E(E - 1 + \varepsilon_E)\Big) - 1\Big] \tag{4.29c}$$

for $\Delta P_E \in \{\Delta P_{E_b}, \Delta P_{E_{m,1}}, \ldots, \Delta P_{E_{m,4}}\}$ and $E \in \{E_b, E_{m,1}, \ldots, E_{m,4}\}$. The selected values for the smoothing parameters are listed in Table 4.3. After applying the smoothing measures in this section, the model equations are ready to be used within an OP that shall be solved by a derivative-based NLP solver. The formulation of adequate OPs for computing minimum-time optimal trajectories is illustrated in the subsequent section.

4.4 Optimisation Problem

Using the model equations presented in the previous sections, this section describes various OPs aiming at completing a lap on the Nürburgring Grand Prix course in minimum time. The vehicle behaviour is described by a continuous, input nonaffine, nonlinear, time-invariant differential equation system of the form (3.1a). The input vector of the system model is given by

$$\mathbf{u} = \Big[T_{m,1} \quad T_{m,2} \quad T_{m,3} \quad T_{m,4} \quad T_{br,1} \quad T_{br,2} \quad T_{br,3} \quad T_{br,4} \quad \delta_f\Big]^T \in \mathbb{R}^9. \tag{4.30}$$

using the motor torques $T_{m,k}$, friction brake torques $T_{br,k}$ and front steering angle δ_f. The state vector is comprised of the individual model parts

$$\mathbf{x} = \Big[\mathbf{x}_{\mathrm{ref}} \quad \mathbf{x}_{\mathrm{veh}} \quad \mathbf{x}_{\mathrm{bat}} \quad \mathbf{x}_{\mathrm{eol}}\Big]^T \in \mathbb{R}^{19}. \tag{4.31}$$

The position of the vehicle relative to the track is determined using the differential equations (3.6) describing the temporal development of the track reference states

$$\mathbf{x}_{\text{ref}} = \begin{bmatrix} s_{\mathcal{R}} & d_{\mathcal{R}} & \Theta \end{bmatrix} \in \mathbb{R}^3. \tag{4.32}$$

The basic vehicle motion is depicted using the states

$$\mathbf{x}_{\text{veh}} = \begin{bmatrix} v_x & v_y & \dot{\psi} & \omega_1 & \omega_2 & \omega_3 & \omega_4 & a_x & a_y \end{bmatrix} \in \mathbb{R}^9 \tag{4.33}$$

following (4.3) and (4.9). The temporal behaviour of the battery pack is illustrated via the SOC and capacitor voltage

$$\mathbf{x}_{\text{bat}} = \begin{bmatrix} \zeta & V_d \end{bmatrix} \in \mathbb{R}^2 \tag{4.34}$$

according to (4.20m) and (4.20d). The EOL is depicted using (4.24) to describe the development of the overload reservoirs over time:

$$\mathbf{x}_{\text{eol}} = \begin{bmatrix} E_b & E_{m,1} & E_{m,2} & E_{m,3} & E_{m,4} \end{bmatrix} \in \mathbb{R}^5. \tag{4.35}$$

Applying the objective (3.17a) composed of the time-optimality term (3.19) and the regularisation term (3.20) favours smooth input solutions, which are better realisable by human drivers. The cost weights are chosen such that the extra time gained by the regularisation term is negligible but smoothness and solvability is greatly enhanced. The minimum-time OP is solved for a vehicle with nominal passive parameters with

Table 4.4: Optimisation parameters for vehicular optimal control problem.

Variable	Value	Unit	Description
$i_{g,\text{nom}}$	6.456	–	nominal gear ratio of transmission
$k_{\text{mot,nom}}$	0.0	–	nominal motor performance shift
$\varepsilon_{\dot{u}}$	$2.0 \cdot 10^{-2}$	–	scaling factor for regularisation-term
$\varepsilon_{\dot{s}}$	1.0	ms^{-1}	safety margin for arc length constraint
$\overline{\delta}_f$	$40.0 \cdot \pi/180.0$	rad	maximum steering angle for front wheels
$\underline{T}_{br,1/2}/\underline{T}_{br,3/4}$	-7025.0/-4030.0	Nm	maximum brake torque for front/rear axle
$\overline{\omega}_m$	1570.0	rad/s	maximum motor speed
$\underline{v}_x$	0.9	ms^{-1}	minimum longitudinal speed
$\underline{I}_b/\overline{I}_b$	-1500.0/480.0	A	minimum/maximum current of battery
$\underline{V}_0/\overline{V}_0$	520.0/806.0	V	min./max. terminal voltage of battery
$\underline{P}_{E_b}/\overline{P}_{E_b}$	18.5/51.0	kW	limit values for battery pack power losses
$\underline{P}_{E_m}/\overline{P}_{E_m}$	4.0/10.0	kW	limit values for motor power losses
$\overline{T}_m$	470.0	Nm	maximum motor torque
$\underline{\zeta}/\overline{\zeta}$	0.724/1.0	–	minimum/maximum state of charge
$\underline{i}_g/\overline{i}_g$	1.0/15.0	–	minimum/maximum gear ratio
$\overline{k}_{\text{mot}}$	0.8	–	maximum performance shift

and without EOL. Furthermore, an optimal passive vehicle setup is determined for a vehicle with EOL by concurrently optimising selected model parameters as described in Section 4.4.1. Adequate initial values and necessary constraints are listed in Section 4.4.2 and 4.4.3, respectively. An elaborate procedure for the generation of initial trajectories is presented in Section 4.4.4. The necessary parameters for posing the OP are listed in Table 4.4.

4.4.1 Passive Optimisation Parameters

For improved vehicle performance, selected model parameters are optimised simultaneously with the system inputs and states. Firstly, the optimal front and rear gear ratios $i_{g,f} = i_{g,1/2}$ and $i_{g,r} = i_{g,3/4}$ are identified. Secondly, a predefined total motor performance is optimally distributed in longitudinal direction using the shift variable k_{mot} according to

$$k_{\mathrm{mot},f} = k_{\mathrm{mot},1/2} = 1 + k_{\mathrm{mot}} \quad \text{and} \quad k_{\mathrm{mot},r} = k_{\mathrm{mot},3/4} = 1 - k_{\mathrm{mot}}. \tag{4.36}$$

The passive optimisation parameters are accumulated in following vector

$$\mathbf{p} = \begin{bmatrix} i_{g,f} & i_{g,r} & k_{\mathrm{mot}} \end{bmatrix}^T. \tag{4.37}$$

In general, with increasing motor power rises the maximum torque and the heat tolerance represented by power losses. However, the maximum motor speed lowers due to the increased rotor inertia. For simplicity, these tendencies are represented by a linear scaling of the maximum motor torque, the motor power limits and the maximum motor speed according to

$$\overline{T}_{m,k} \coloneqq k_{\mathrm{mot},k}\, \overline{T}_m \quad \forall k \in \mathbb{K}, \tag{4.38a}$$

$$\underline{P}_{E_{m,k}} \coloneqq k_{\mathrm{mot},k}\, \underline{P}_{E_m} \quad \forall k \in \mathbb{K} \quad \text{and} \quad \overline{P}_{E_{m,k}} \coloneqq k_{\mathrm{mot},k}\, \overline{P}_{E_m} \quad \forall k \in \mathbb{K}, \tag{4.38b}$$

$$\overline{\omega}_{m,k} \coloneqq (1 - k_{\mathrm{mot}})\, \overline{\omega}_m \;\forall k \in \{1,2\} \quad \text{and} \quad \overline{\omega}_{m,k} \coloneqq (1 + k_{\mathrm{mot}})\, \overline{\omega}_m \;\forall k \in \{3,4\}. \tag{4.38c}$$

Values for $\overline{T}_m$, $\overline{\omega}_m$, $\underline{P}_{E_m}$ and $\overline{P}_{E_m}$ are given in Table 4.4. With (4.38), the passive optimisation parameters (4.37) affect the wheel inertias in (4.19), the maximum wheel-based motor torque in (4.47) and the maximum wheel speed in (4.48). Additionally, the capacities of the motor overload tanks in (4.26) are scaled by the corresponding shift variable $k_{\mathrm{mot},k}$ due to the adaptation of both power limits in (4.38b). For simplicity, it is assumed that the changes in motorisation do not affect the weight distribution of the vehicle and the power loss maps (4.20f) and (4.20g).

4.4.2 Initial Condition

Assuming a rather small battery pack capacity Q_b, the task of completing $n_{\mathrm{lap}} = 5$ laps is intended. The optimisation examines only the first lap in order to reduce the problem size. Hence, the vehicle starts with a SOC of $\zeta = \overline{\zeta}_r$ and full overload tanks with $E_b = 1 = E_{m,k}\ \forall k \in \mathbb{K}$. The initial track segment is straight. Thus, the steering

angle, lateral speed and yaw rate are initially set to zero. Furthermore, the vehicle starts in the centre of the racetrack in alignment with the track reference curve yielding $d_{\mathcal{R}} = \Theta = \psi = 0$. Imposing a small initial longitudinal speed $v_x = \mathrm{v}_{x,k} = 1\frac{m}{s}$ avoids the singularity at vanishing speed in (4.11b). Since steady-state conditions are assumed at the race start, accelerations are initially zero resulting in only static and aerodynamic wheel loads. For simplicity, the longitudinal wind force is assumed to be fully imposed on the rear tire forces. The friction brake torques are initially set to zero. These assumptions enable the computation of the front and rear tire forces as well as the wheel-based motor torques via (4.3a) and (4.3c) resulting in

$$\mathcal{F}_{x,1} = -f_{\mathrm{roll},0}\, F_{z,1} = \mathcal{F}_{x,2}, \quad \mathcal{F}_{x,3} = \frac{1}{2}(-F_{x,\mathrm{air}} - \mathcal{F}_{x,1} - \mathcal{F}_{x,2}) = \mathcal{F}_{x,4} \quad \text{and} \tag{4.39a}$$

$$T_k = \mathcal{F}_{x,k}\, r_k - \mathcal{T}_{y,\mathrm{roll},k} \quad \forall k \in \mathbb{K}. \tag{4.39b}$$

After identifying the initial tire forces (4.39a), the longitudinal slip and the rotational wheel speeds are computed via (4.12e) and (4.11a). The initial motor torques are determined by inserting (4.39b) in (4.16). Motor speeds are computed using the previously calculated wheel speeds and (4.18). Presuming steady-state conditions, (4.20d) results in $V_d = R_d I_b$. Using this relation together with (4.20i) as well as the initial SOC, motor speeds and motor torques, provides the initial capacitor voltage V_d.

4.4.3 Constraints

As stated in Section 3.5, the integration interval is divided into n_{seg} segments yielding n_{coll} collocation points. Each state and input at each collocation point represents an optimisation parameter. Applying the collocation constraints (3.15) to each collocation segment ensures the satisfaction of the system dynamics. However, further constraints are necessary to consider non-modelled physical effects, enable simplifications and avoid numerical problems. Although the OP is formulated using the scaled optimisation variables $\hat{\mathbf{x}}$, $\hat{\mathbf{u}}$ and $\hat{\mathbf{p}}$, the remaining constraints are depicted using the original variables to improve comprehensibility. All subsequent constraints are imposed at each collocation point, however the indices are omitted to enhance readability.
In order to prevent the vehicle from leaving the racetrack, the lateral deviation of the vehicle from the track centre line is limited via

$$-d_{hw} + \frac{b_v}{2} \leq d_{\mathcal{R}} \leq d_{hw} - \frac{b_v}{2} \tag{4.40}$$

using the chassis width b_v and the arc length-dependent track half-width d_{hw}. Driving backwards on racetracks is not reasonable and is hence avoided by enforcing positive longitudinal speeds via a lower limit

$$0 < \underline{v}_x \leq v_x. \tag{4.41}$$

When driving on racetracks, (4.41) generally results in $v_{x,k}, \mathrm{v}_{x,k} > 0$ omitting the singularity in (4.11b). Furthermore, (4.41) enables the simplification made in (4.11a).

In order to stay within the stable region of the tires, the wheel slips (4.11a) and (4.11b) are enforced to stay within the load-dependent friction ellipse

$$g_{s,k} \coloneqq \left(\frac{\lambda_{x,k}}{\lambda^{\star}_{x,k}}\right)^2 + \left(\frac{\lambda_{y,k}}{\lambda^{\star}_{y,k}}\right)^2 - 1 \leq 0 \quad \forall k \in \mathbb{K} \tag{4.42}$$

defined by the optimal slips (4.13b). Guaranteeing tire forces within the region of adhesion, (4.42) results in an efficient tire usage and reduced tire wear. Since the tire force potential reduces with tire wear, constraining the slips via (4.42) is eligible.
Tire forces can only be transmitted as long as the wheels have ground contact. In order to avoid additional discontinuities as in [158], positive wheel loads are demanded:

$$F_{z,k} > 0 \quad \forall k \in \mathbb{K}. \tag{4.43}$$

Moreover, dividing by zero when implementing the spatial reformulation (3.7) is avoided by imposing

$$\dot{s}_{\mathcal{R}} \geq \varepsilon_{\dot{s}} > 0, \tag{4.44}$$

which is legitimate since backwards driving is not a reasonable scenario on racetracks.
Generally, the operating range of actuators is limited, which is captured by box constraints for the front steering angle

$$-\overline{\delta}_f \leq \delta_f \leq \overline{\delta}_f \tag{4.45}$$

and for the friction brake torques

$$\underline{T}_{br,k} \leq T_{br,k} \leq 0 \quad \forall\, k \in \mathbb{K}. \tag{4.46}$$

Further actuator constraints are given for the electrical machines. The motor torques are limited by the maximum torque (4.38a) yielding

$$-\overline{T}_{m,k} \leq T_{m,k} \leq \overline{T}_{m,k} \quad \forall\, k \in \mathbb{K}. \tag{4.47}$$

Since we exclude driving backwards on racetracks, adequate constraints are set to enforce positive motor speeds. Considering the maximum motor speed (4.38c), the wheel speeds are limited according to

$$0 \leq \omega_k \leq \frac{\overline{\omega}_{m,k}}{i_{g,k}} \quad \forall\, k \in \mathbb{K}. \tag{4.48}$$

Furthermore, a safe operation of the battery pack requires strict limits on the current of the battery pack

$$\underline{I}_b \leq I_b \leq \overline{I}_b \tag{4.49}$$

as well as on its terminal voltage

$$\underline{V}_0 \leq V_0 \leq \overline{V}_0. \tag{4.50}$$

The charging current is chemically limited by $\overline{I}_b$, whereas the maximum discharging current $\underline{I}_b$ generally depends on thermodynamic conditions resulting in $|\underline{I}_b| \gg \overline{I}_b$.

As mentioned in Section 4.4.2, only the first of $n_{\text{lap}} = 5$ laps is considered by the optimisation. The first lap generally requires more energy due to the initial acceleration phase. Thus, a slightly higher portion

$$p_{\text{lap}} = \frac{1}{n_{\text{lap}}} + 2\% = 0.22 \tag{4.51}$$

is granted for the battery pack capacity. Using (4.51), the initial SOC and the recommended working zone $\zeta_r \in \left[\underline{\zeta}_r, \overline{\zeta}_r\right]$, the SOC boundaries are set according to

$$\underline{\zeta} \coloneqq \zeta(s_{\mathcal{R},0}) - p_{\text{lap}}(\overline{\zeta}_r - \underline{\zeta}_r) \leq \zeta \leq 1 = \overline{\zeta}. \tag{4.52}$$

Therein the true maximum upper bound is admitted enabling a fully charged battery pack. Furthermore, the overload reservoirs are constrained from below via

$$0 \leq E_b, E_{m,k} \quad \forall k \in \mathbb{K}. \tag{4.53}$$

Due to the saturation in (4.24), the overload tanks do not need to be limited from above. Finally,the passive optimisation parameters are restricted according to

$$\underline{i}_g \leq i_{g,f/r} \leq \overline{i}_g \quad \text{and} \quad -\overline{k}_{\text{mot}} \leq k_{\text{mot}} \leq \overline{k}_{\text{mot}} \tag{4.54}$$

to only allow plausible gear ratios and motor powers.

4.4.4 Initialisation Routine

Direct collocation methods are generally robust enough to cope with initialising the decision variables with zeros or linearly interpolating between the initial and final value. Yet, providing a good initial solution for the OP can greatly reduce computation time. Furthermore, derivative-based optimisation methods only ensure convergence to local optima and can potentially get stuck in suboptimal solutions.
There are various possibilities to deduce better initial solutions. If available, measurements can be used to generate initial trajectories. In case the OP is complicated, a

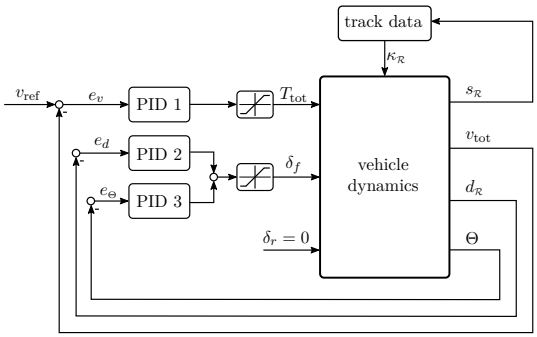


Figure 4.7: Triple PID controller structure for generation of initial trajectories for the OCP.

simplified version of the problem can be solved first using its solution as initialisation for the actual problem. Another method is generating initial trajectories using tracking controllers to follow a predefined trajectory. For racetrack applications, the very first trajectories for the inputs and states can be created via the simple controller structure depicted in Fig. 4.7 that tracks the racetrack centre line [185]. Three separate proportional-integral-derivative (PID) controllers are used to reduce the lateral deviation $e_d = -d_{\mathcal{R}}$, the tangent angle error $e_\Theta = -\Theta$ and the speed error $e_v = v_{\mathrm{ref}} - v_{\mathrm{tot}}$. A constant, small reference speed v_{ref} is chosen whereas the torques are allocated equally according to $T_k = \frac{1}{4} T_{\mathrm{tot}}$.
Regardless of which initial solution is chosen, the number of collocation segments should be increased gradually starting with a rather coarse mesh to simplify the computation of the very first optimal solution. Thus, the quality of the subsequent optimisations successively increases while preventing that the solver fails entirely to find a solution.

4.5 Results

The OP presented in the previous section is solved using 1000 equally spaced collocation points yielding a mesh interval of about 2.5 metres. An OP is solved for each of the vehicle configurations C0-C2 listed in Table 4.5. The solution for nominal configuration C0 represents the baseline solution. This setup represents a vehicle with nominal parameter values and without EOL. Hence, the power losses of battery pack and electric machines must stay below the corresponding limit for continuous operation: $P_{l_b} \leq \underline{P}_{E_b}$ and $P_{l_{m,k}} \leq \underline{P}_{E_{m,k}}$ $\forall k \in \mathbb{K}$. The influence of EOL is analysed via configuration C1. In order to identify the optimal passive setup, the passive parameters (4.37) are simultaneously optimised for configuration C2 while also considering EOL. The individual setups are compared in regards to various features. Considering (4.36), the performance percentage of the rear motors is given by

$$p_{\mathrm{mot},r} \coloneqq \frac{1}{2} k_{\mathrm{mot},r} = \frac{1}{2} \left(1 - k_{\mathrm{mot}}\right). \tag{4.55}$$

The recuperation percentage

$$k_{\mathrm{rec}} \coloneqq \frac{\int_{t_0}^{t_f} \max(V_0 I_b, 0) dt}{\int_{t_0}^{t_f} \left|\min(V_0 I_b, 0)\right| dt}. \tag{4.56}$$

classifies the ratio between recuperated energy and consumed energy. The time advantage progression compared to the nominal configuration is defined as

$$\Delta t_{Ck}(s_{\mathcal{R}}) = t_{Ck}(s_{\mathcal{R}}) - t_{C0}(s_{\mathcal{R}}) \quad \text{with} \quad k \in \{1, 2\}. \tag{4.57}$$

The time advantage progression and track curvature depicted in Fig. 4.8 show that EOL mainly reduces lap time at the race start and after exiting corners. The causes of these lap time progressions are explained below by examining the optimal trajectories of the individual vehicle setups.

Table 4.5: Optimisation results for individual vehicle configurations.

	EOL	CO	parameter values			lap time			$k_{\text{rec}}[\%]$
			$i_{g,f}[-]$	$i_{g,r}[-]$	$p_{\text{mot},r}[\%]$	$t_{\text{lap}}[\text{s}]$	$\Delta t_{\text{lap}}[\text{s}]$	$\Delta t_{\text{lap}}[\%]$	
C0			6.456	6.456	50.0	141.58	-	-	31.42
C1	✓		6.456	6.456	50.0	135.60	-5.98	-4.22	31.95
C2	✓	✓	4.987	6.275	59.3	135.45	-6.13	-4.33	31.85

EOL: electrical overloading. CO: concurrent optimisation of passive parameters (4.37). $i_{g,f}/i_{g,r}$: front/rear gear ratio. $p_{\text{mot},r}$: performance percentage of rear motors (4.55). t_{lap}: lap time. Δt_{lap}: lap time advantage over configuration C0. k_{rec}: recuperation percentage (4.56).

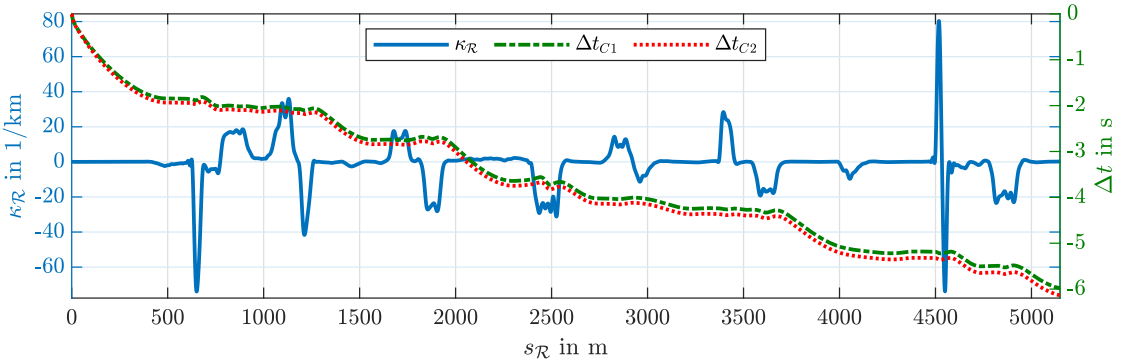


Figure 4.8: Racetrack curvature $\kappa_{\mathcal{R}}$ and course of time advantage Δt_{Ck} compared to setup C0.

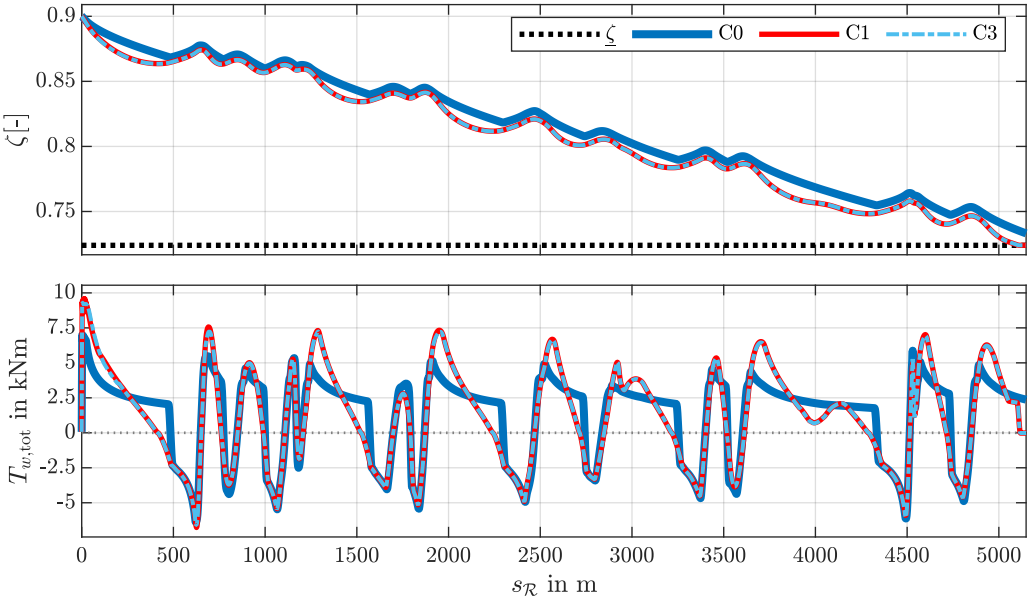


Figure 4.9: State of charge ζ and total motor torque $T_{w,\text{tot}}$ for individual vehicle setups.

Fig. 4.9 compares the SOC ζ and total wheel-based motor torques $T_{w,\mathrm{tot}} = \sum_{k=1}^{4} T_{w,k}$ to provide general energetic discrepancies between the individual configurations. The figure illustrates that setups C1 and C2 show higher total motor torques due to the possibility to overload the battery pack and motors. Thus, the full amount of energy allowed by (4.52) is consumed so that the SOC is at its lower limit at the end of the lap. Considering the performance-oriented cost function, this indicates that the results are correct. For C1 and C2, the gradient in the torque trajectory is substantially steeper switching faster from drive torques to motor brake torques. Thus, the SOC trajectories for C1 and C2 diverge from the trajectories for C0 on acceleration phases and approach each other again when decelerating. The recuperation percentages in Table 4.5 show that the EOL also increased efficiency by recovering 0.53% and 0.43% more energy for setup C1 and C2, respectively.

Optimal Electrical Overloading In order to go into depth regarding EOL, selected trajectories are subsequently shown for setup C1. The heat power flows of the battery pack ΔP_{E_b}, the front right motor $\Delta P_{E_{m,2}}$ and the rear left motor $\Delta P_{E_{m,3}}$ are depicted for setup C1 in Fig. 4.10, Fig. 4.11 and Fig. 4.12, respectively. Additionally, Fig. 4.13 illustrates the trajectories of the overload reservoirs, wheel-based motor torques, friction brake torques, the current and terminal voltage of the battery pack as well as the friction ellipse constraints for setup C1.
Due to the far higher limit for discharging currents than charging currents $|\underline{I}_b| \gg \overline{I}_b$ in (4.49), the battery pack is mainly overloaded to enhance acceleration rather than recuperation. Hence, the battery boosting mainly occurs at the race start and at corner exits to enable higher discharging currents that result in higher drive torques. Furthermore, the battery pack is chilled during the braking phase before corners, which is

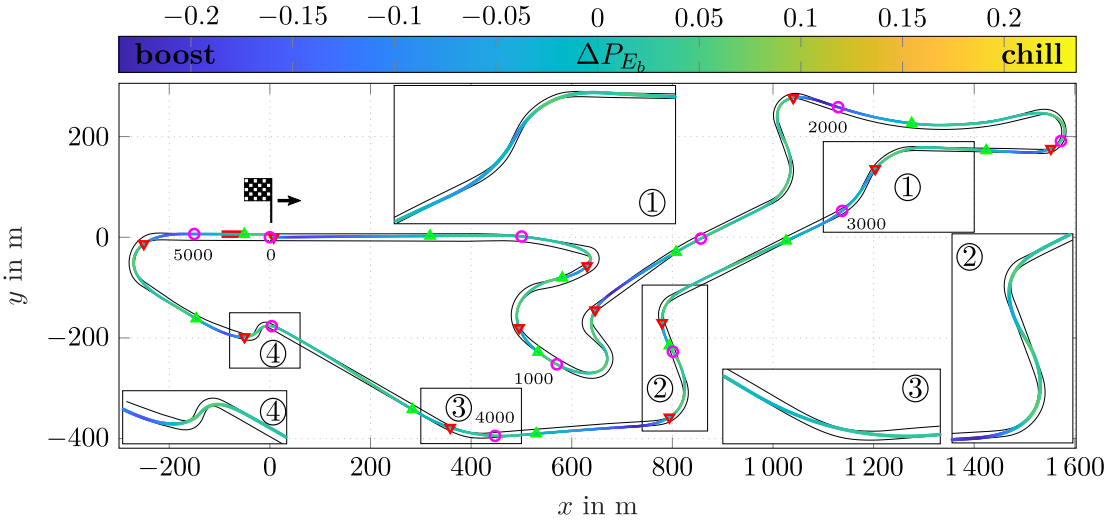


Figure 4.10: Overloading of battery pack for vehicle configuration C1. Symbols: ○ 500m marks, △ start chilling, ▽ start boosting, — empty overload reservoir.

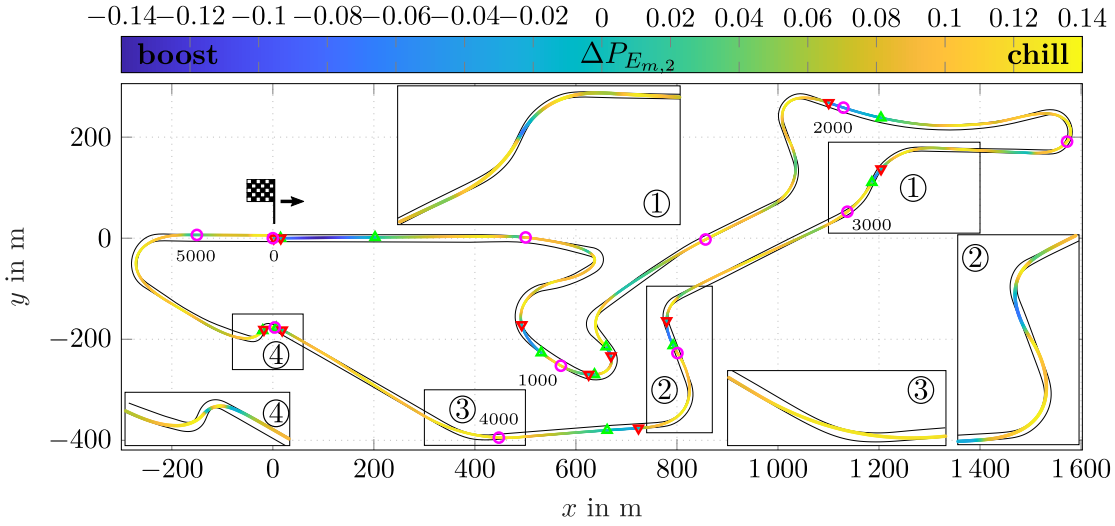


Figure 4.11: Overloading of front right motor for vehicle configuration C1. Symbols: ○ 500m marks, △ start chilling, ▽ start boosting.

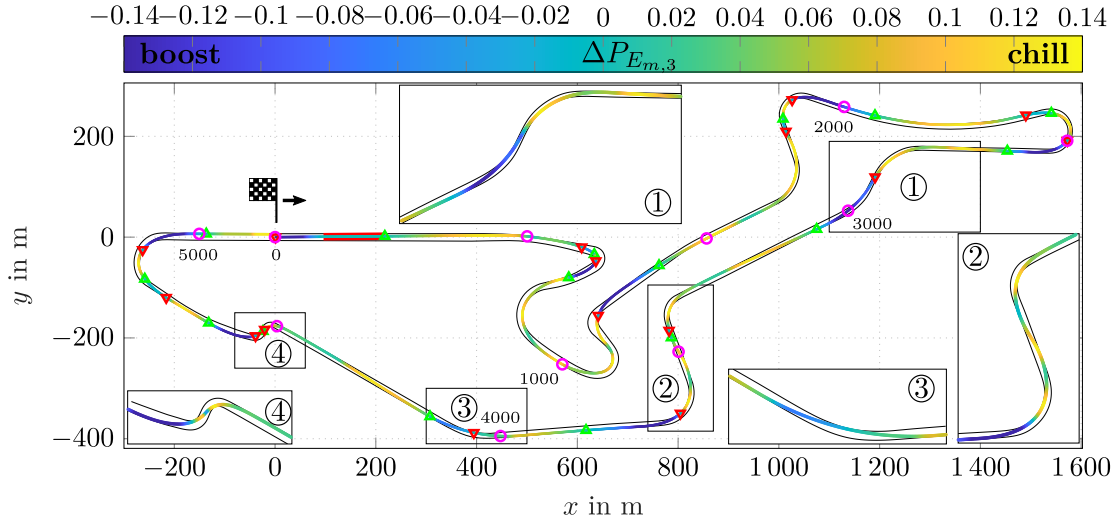


Figure 4.12: Overloading of rear left motor for vehicle configuration C1. Symbols: ○ 500m marks, △ start chilling, ▽ start boosting, — empty overload reservoir.

represented by recharging the overload tank of the battery pack. This replenishes the boosting capability for the subsequent acceleration phase. The overloading of the electric motors occurs during the initial acceleration at the race start as well as after corner apexes. Generally, motor overloading is used to increase the wheel torque in unison with the wheel load distribution to better exploit the force potential of the individual tires: Mainly the rear motors are overloaded when accelerating and the motors of the outer wheels are boosted more than the inner wheels when exiting corners. In summary, EOL

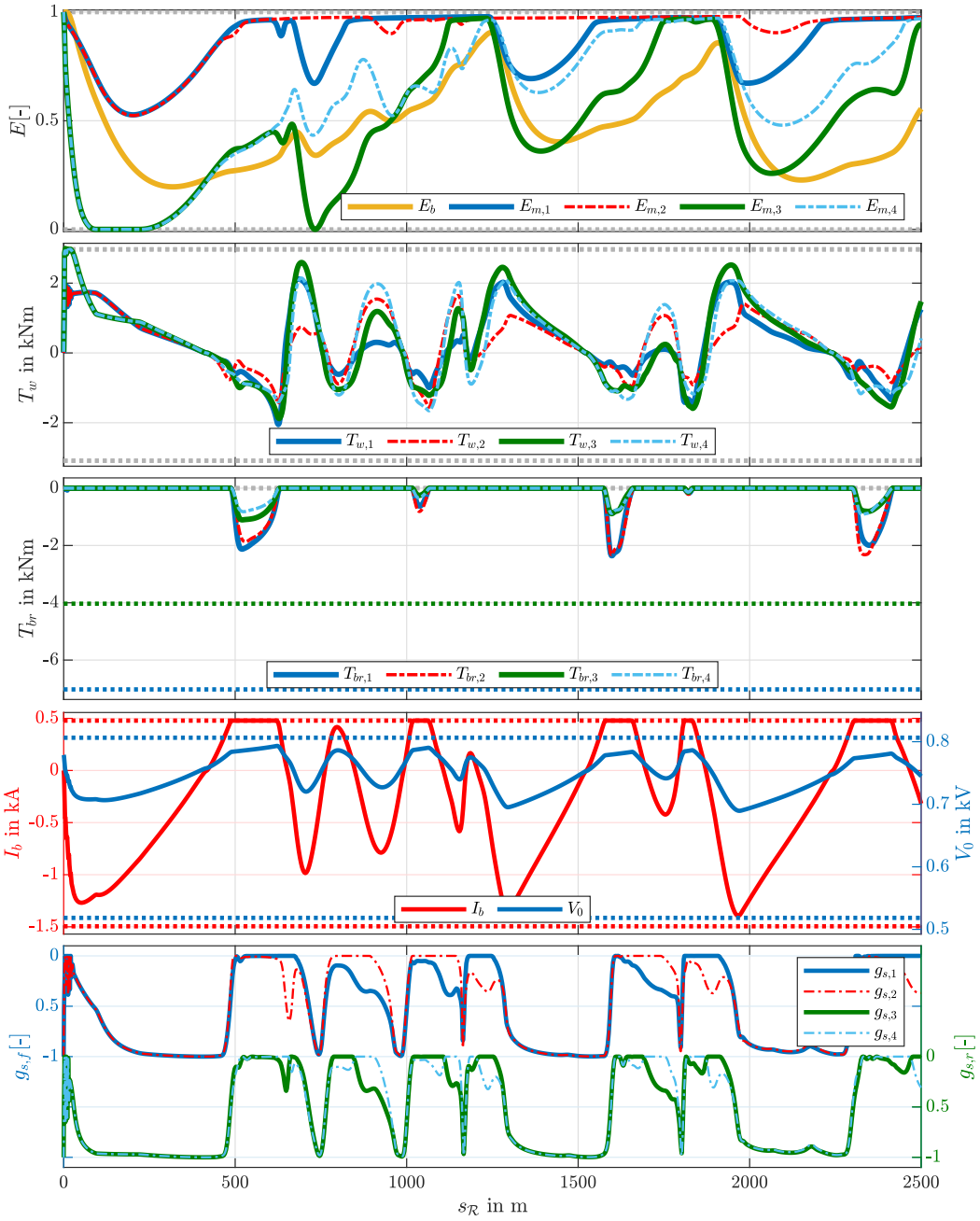


Figure 4.13: Overload reservoirs E, wheel-based motor torques T_w and friction brake torques T_{br}, current I_b and terminal voltage V_0 of battery pack as well as friction ellipse constraints g_s for configuration C1. Dotted lines represent bounds.

is mainly used for acceleration and therefore improves lap time primarily at the race start and after corners by enabling higher drive torques. As illustrated in Fig. 4.13, primarily the overload storages of battery pack and rear motors are drained due to the

preference for boosted acceleration. When accelerating from slow speeds, like at the race start or after tight corners, the overload storages of the rear motors are drained to emptiness for maximal acceleration. Overall, lap time of setup C1 is 4.22% smaller than the lap time of setup C0.
Fig. 4.13 reveals the general torque allocation strategy for minimum lap times. Besides the already mentioned longitudinal torque allocation, torque is also distributed laterally. Firstly, the lateral torque discrepancies generate an additional yaw moment that stabilises the vehicle when braking on corner entries and enhances agility when accelerating after corner apexes. Secondly, the tires are used more efficiently: The lateral wheel load distribution during cornering results in different optimal slips (4.13b) for the inner and outer wheel of an axle, which is considered by applying individual wheel torques. The motor brake torques saturate due to the maximum charging current $\overline{I}_b$ in constraint (4.49). When the battery pack current reaches this boundary, the friction brakes are used to provide additional brake torque. This facilitates the full exploitation of the tire potential when braking resulting in friction ellipse constraints (4.42) at the limit of adhesion. Since the braking system does not engage otherwise, the friction brake torques are far from saturating. However, the brake torque allocation between motors and braking system depends on the motor characteristics and the powertrain topology. It can be necessary to use friction brake torques more frequently when smaller motors with lower maximum torque are installed. The benefit of additional friction braking increases for vehicles with less than four electric machines due to the possibility of differential friction braking. Another factor that strongly influences the utilisation of the friction brakes is the remaining SOC. Lowering the available energy percentage p_{lap} reduces the engagement of the braking system. Then, it is beneficial to drive slower and not exploit the full tire potential. Thus, less energy is spent and more energy can be recuperated by the motor brake torques, which are then capable to account for the full brake torque.

Concurrent Optimisation In order to further improve lap time, the front and rear gear ratios as well as the longitudinal distribution of the motor performance have been simultaneously optimised for setup C2. The optimiser reduces the front gear ratios, whereas similar gear ratios are maintained at the rear. Considering the torque transfer from motors to wheels in (4.16), this enables larger torques at the rear wheels. Furthermore, less torque is required to accelerate the front wheels since the inertia of the front wheel-units reduces following (4.19). Although not representing a limit for the current application, the smaller front gear ratios enable higher rotational speeds at the front wheels via the upper wheel speed limit in (4.48). A higher gear ratio at the rear wheels than at the front wheels is beneficial for acceleration due to the rear weight bias of 54.0% and the larger rear axle loads when accelerating.
Besides the gear ratios, the fixed motor performance is shifted via k_{mot} to the rear yielding a performance bias of $p_{\mathrm{mot},r} = 59.3\%$. Following (4.38a)-(4.38b) and (4.26), this increases the maximum motor torque limits $\overline{T}_{m,k}$, the lower motor power limit $\underline{P}_{E_{m,k}}$ and the overload tank capacity $\overline{E}_{m,k}$ for the rear motors while reducing the corresponding values for the front motors. Consequently, overloading is improved for the rear motors

by allowing a higher continuously usable power and increasing the overload capacity. The overload tank trajectories in Fig. 4.13 illustrate that this entails a benefit when accelerating from slow speeds like at the race start or after tight corners. Although not depicted in a figure, the maximum wheel speed in (4.48) is not reached for the current track and setup. Thus, it does not represent a limiting factor in the current optimisation. However, the motor performance shift decreased the maximum rotational motor speed at the rear wheels and increased it at the front wheels via (4.38c).
While the mentioned measures improve acceleration potential, recuperation potential k_{rec} is marginally reduced by 0.1% compared to setup C1, as depicted in Table 4.5. The concurrent optimisation of the passive parameters results in an additional lap time improvement of 0.11%, compared to C1. The lap time improvement is only small since the basic vehicle setup of C1 is already quite good. However, the results show that different setups can achieve similar performances and the presented method can be used to simultaneously identify optimal vehicle parameters.

5 Convex Quadratic Programming via Space Splitting Convexification

Optimal control methods for realistic engineering problems generally require the solution of nonconvex OPs, which are rather hard to solve. The second part of this thesis deals with reducing the computation time required to solve such nonconvex problems. A novel successive convexification method, which is called space splitting convexification (SSC) and has been introduced in our publication [186], is presented in this chapter. It decomposes certain nonconvexities into affine parts by introducing auxiliary decision variables and absolute value constraints (AVCs). In an iterative procedure, these AVCs are linearised around the solution of the preceding iteration and transferred to the objective function using the concept of exact penalty functions. An intermediate projection routine identifies an adequate initial guess. The utilised AVCs possess a beneficial structure for linearisation and point projection onto constraints. The linearisation of the AVCs results in a binary decision: wrong or right sign of the absolute value term. In each iteration, the preceding solution is projected onto the AVCs and the sign is corrected if necessary enabling the linearisation errors to vanish in subsequent iterations. The violation of the linearised AVCs serves as feedback whether the correct sign was chosen, which is used as indicator for convergence. The approach is capable of considering two types of nonconvexities. Firstly, a class of nonconvex sets that can be split into convex subsets, which are called zonally convex sets (ZCSs) in this thesis. Such sets arise for instance when semi-active actuators are used. Secondly, equality constraints with possibly multiple but univariate nonlinearities. A common source for these nonlinear equality constraints are nonlinear system equations when applying optimal control methods. Requiring only simple mathematical operations, the AVC-projection is of linear computational complexity. Furthermore, the binary nature of the AVC-linearisation generally results in few superordinate iterations. The method produces an efficiently solvable substitute problem, which is a convex QP problem or even LP problem if the original objective is linear. Thus, the SSC algorithm greatly reduces computation time enabling an efficient solution in polynomial time. Furthermore, robust initialisation properties are given since the initial guess is not required to be feasible. The algorithm converges under certain conditions to local optima of the piecewise linear substitute problem, which represents a scalable approximation of the original problem. However, being a local solution method, the computed local solution generally depends on the provided initial guess. For good initialisations, the algorithm computes solutions that are close to the global optimum.
This chapter contains parts of our publication [186] and is organised as follows. The initial SOP for the numerical solution of the optimal control task is presented in Section 5.1. Afterwards, Section 5.2 derives the successive convexification algorithm and provides a

proof of convergence. The specific application of the SSC algorithm to two-dimensional ZCSs and nonlinear equality constraints is illustrated in Section 5.3. Remarks on the computation time of the algorithm are given in Section 5.4. Section 5.5 closes the chapter with a comparison of the presented method with related approaches.

5.1 Problem Formulation

Optimal control deals with computing the optimal input and state trajectories for a given system model. The motion of a nonlinear, continuous, time-invariant system is given by the differential equation system

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{5.1}$$

with states $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and inputs $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$. An optimal control trajectory $\mathbf{u}(t)$ for a system described by (5.1) and time interval $t \in [t_0, t_f]$ can be computed by solving a dynamic OP of the form (2.17). The SSC approach aims at solving such problems numerically, employing a direct collocation method due to the simpler initialisation of direct methods and the increased sparsity of collocation methods. Using separated Hermite-Simpson collocation with a discretisation into n_{seg} segments yields $n_{\text{coll}} = 2\,n_{\text{seg}} + 1$ collocation points and thus the vector of decision variables

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0^T & \mathbf{w}_{\frac{1}{2}}^T & \mathbf{w}_1^T & \dots & \mathbf{w}_{n_{\text{seg}}}^T \end{bmatrix}^T \in \mathbb{R}^{n_w} \quad \text{with} \quad \mathbf{w}_k^T \coloneqq \begin{bmatrix} \mathbf{x}_k^T & \mathbf{u}_k^T \end{bmatrix}. \tag{5.2}$$

The SSC procedure is applicable to NLP problems of the form

$$\begin{aligned}
\min_{\mathbf{w}} \quad & J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\,\mathbf{P}_0\,\mathbf{w} + \mathbf{q}_0^T\,\mathbf{w} + r_0 && \text{(5.3a)}\\
\text{s.t.} \quad & \mathbf{c}_{\text{coll},i}(\mathbf{w}_i, \mathbf{w}_{i+\frac{1}{2}}, \mathbf{w}_{i+1}) = \mathbf{0} \qquad \forall i \in \mathcal{I}_{\text{coll}}, && \text{(5.3b)}\\
& \mathbf{c}_k = \mathbf{A}_c\,\mathbf{w}_k + \mathbf{k}_c = \mathbf{0} \qquad \forall k \in \mathcal{K}, && \text{(5.3c)}\\
& \mathbf{h}_k(\mathbf{w}_k) \leq \mathbf{0} \qquad \forall k \in \mathcal{K}, && \text{(5.3d)}\\
& \mathbf{h}_{\text{zcs},k}(\mathbf{w}_k) \leq \mathbf{0} \qquad \forall k \in \mathcal{K} && \text{(5.3e)}
\end{aligned}$$

with indices $i \in \mathcal{I}_{\text{coll}}$ and $k \in \mathcal{K}$ representing the corresponding collocation segment and collocation point according to (3.18), respectively. Objective (5.3a) is assumed to be quadratic and convex in the decision variables with $r_0 \in \mathbb{R}$, $\mathbf{q}_0 \in \mathbb{R}^{n_w}$ and $0 \leq \mathbf{P}_0 \in \mathbb{R}^{n_w \times n_w}$. The positive semi-definiteness of $\mathbf{P}_0$ concludes the convexity of the objective. Collocation constraints (5.3b) ensure that the system equations (5.1) are satisfied. For Hermite-Simpson collocation, these equality constraints are given by

$$\mathbf{c}_{\text{coll},i} \coloneqq \begin{bmatrix} \mathbf{x}_{i+1} - \mathbf{x}_i - \frac{1}{6}\Delta_i(\mathbf{f}_i + 4\,\mathbf{f}_{i+\frac{1}{2}} + \mathbf{f}_{i+1}) \\ \mathbf{x}_{i+\frac{1}{2}} - \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_{i+1}) - \frac{1}{8}\Delta_i(\mathbf{f}_i - \mathbf{f}_{i+1}) \end{bmatrix} = \mathbf{0} \tag{5.4}$$

with $\mathbf{f}_j \coloneqq \mathbf{f}(\mathbf{x}_j, \mathbf{u}_j)$ and $\Delta_i = t_{i+1} - t_i$, as introduced in Section 3.5[9]. The SSC approach is capable of considering nonlinear right-hand sides of the form

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \left(\mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u} + \mathbf{k}\right) + \mathbf{f}_{\text{pwl}}(\mathbf{x}, \mathbf{u}) \tag{5.5}$$

[9] Without spatial reformulation of the OP, the collocation method performs a time discretisation.

with $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B} \in \mathbb{R}^{n_x \times n_u}$ and $\mathbf{k}, \mathbf{f}_{\text{pwl}} \in \mathbb{R}^{n_x}$. Therein $\mathbf{f}_{\text{pwl}}$ represents a composition of univariate nonlinearities that can be depicted or approximated by piecewise linear curves. For simplicity, it is assumed that the remaining equality constraints (5.3c) are affine in the decision variables. The inequality constraints (5.3d) are required to be convex in the decision variables. However, aiming at constructing a QP problem, it is assumed below that the inequality constraints are even affine. This stricter assumption can be posed without loss of generality since convex inequality constraints can be approximated via a polyhedron using multiple affine functions [19, p.32]. The inequality constraints (5.3e) depict ZCSs, which are nonconvex sets that can be split into convex subsets.

5.2 Successive Convexification Procedure

Considering the standard form of convex OPs (2.26), all equality constraints must be affine in the decision variables to enable the convexity of the problem[10]. Hence, a nonlinear right-hand side of the system equations (5.1) results in nonlinear equality constraints (5.4), yielding a nonconvex OP. Furthermore, the nonconvex inequality constraints (5.3e) also prohibit a convex problem.
As schematically illustrated in Fig. 5.1, the SSC algorithm iteratively solves a convex substitute problem (5.20), which is derived from the original problem (5.3). The core idea of the SSC concept is considering the transformed problem (5.13), which represents a piecewise linear approximation of the original problem (5.3). This transformation is achieved by introducing space splitting constraints. These constraints are nonconvex but render the original constraints convex. Furthermore, they possess an advantageous structure that is exploited by the algorithm. In order to deal with the remaining nonconvexity, the nonconvex space splitting constraints are transferred to the objective following the theory of exact penalty functions. The resulting problem (5.17) possesses a convex feasible set but a nonconvex objective. Thus, the nonconvex part of the objective is iteratively linearised around points that are adjusted based on the previous iterate using an intermediate projection routine. This yields the convex problem (5.20), which is sequentially solved within the SSC algorithm. The smoothable absolute value function (3.2) with smoothing parameter $0 \leq \varepsilon_s \ll 1$ is subsequently employed: It recovers the true absolute value for $\varepsilon_s = 0$. The intermediate OPs $(5.13)_\varepsilon$ and $(5.17)_\varepsilon$ depicted in Fig. 5.1 apply $\varepsilon_s > 0$ to provide continuous differentiability for the validity of the optimality conditions legitimising the application of exact penalty functions. Afterwards, this smoothing is removed since it represents an approximation that is not required in problem (5.20) for continuous differentiability. The subsequent sections describe the problem transformation process in more detail.

[10] The special case of geometric programming (2.34) is excluded for simplicity.

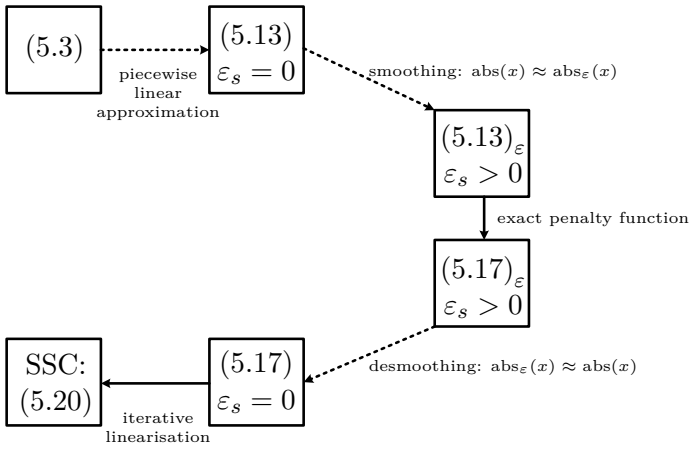


Figure 5.1: Transformation of OP. Dotted and solid arrows represent approximated and exact transformations, respectively.

5.2.1 Space Splitting Constraints

Before presenting the individual OPs, the basic principle for the space splitting of a decision variable is illustrated in this section providing graphical support. If collocation is used, the inputs as well as states represent decision variables. Thus, the input space and the state space can be split. The splitting procedure is presented using $w \in \mathcal{W} \coloneqq [\underline{w}, \overline{w}]$ as a representative decision variable that is split at the transition point $w = w_{\mathrm{tr}}$. As illustrated in Fig. 5.2a, the auxiliary decision variables $w_{\mathrm{low}} \in \mathcal{W}_{\mathrm{low}} \coloneqq [\underline{w}, w_{\mathrm{tr}}]$ and $w_{\mathrm{up}} \in \mathcal{W}_{\mathrm{up}} \coloneqq [w_{\mathrm{tr}}, \overline{w}]$ are introduced aiming at satisfying following relations:

$$w_{\mathrm{low}} = \begin{cases} w & \forall w \leq w_{\mathrm{tr}}, \\ w_{\mathrm{tr}} & \text{else} \end{cases}, \quad w_{\mathrm{up}} = \begin{cases} w & \forall w \geq w_{\mathrm{tr}}, \\ w_{\mathrm{tr}} & \text{else} \end{cases}. \tag{5.6}$$

Lemma 5.2.1. *The splitting relations* (5.6) *can be implemented via the splitting constraints*

$$g_{\mathrm{aff}} = w_{\mathrm{up}} + w_{\mathrm{low}} - (w + w_{\mathrm{tr}}) = 0 \tag{5.7a}$$

$$g_{\mathrm{abs}} = \underbrace{w_{\mathrm{up}} - w_{\mathrm{low}}}_{=:w_\Delta} - |\underbrace{w - w_{\mathrm{tr}}}_{=:\tilde{w}}| = 0. \tag{5.7b}$$

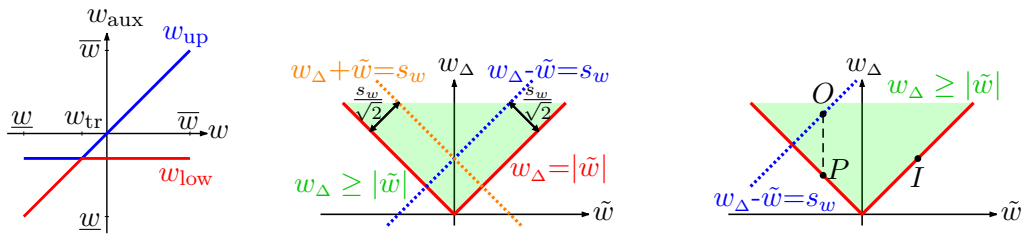
Proof. Solving (5.7a) for one of the auxiliary decision variables yields

$$w_{\mathrm{low}} = w + w_{\mathrm{tr}} - w_{\mathrm{up}}. \tag{5.8a}$$

Inserting (5.8a) into (5.7b) results in

$$w_{\mathrm{up}} = \frac{1}{2}\left(|w - w_{\mathrm{tr}}| + w + w_{\mathrm{tr}}\right) \quad \Rightarrow \quad \begin{cases} w_{\mathrm{up}} = w \overset{(5.8a)}{\Rightarrow} w_{\mathrm{low}} = w_{\mathrm{tr}} & \forall\ w \geq w_{\mathrm{tr}} \\ w_{\mathrm{up}} = w_{\mathrm{tr}} \overset{(5.8a)}{\Rightarrow} w_{\mathrm{low}} = w & \forall\ w \leq w_{\mathrm{tr}} \end{cases}. \tag{5.8b}$$

Thus, the constraints in (5.7) ensure the desired splitting according to (5.6). ■



(a) Splitting of decision variable w into w_{low} and w_{up}.

(b) AVC (5.7b) and its relaxed linearisation according to (5.11a).

(c) Projection after optimisation with initially wrong sign. I: initial guess. O: optimum. P: projection of optimum.

Figure 5.2: Space splitting.

The AVC (5.7b) is relaxed into a convex inequality constraint, which can be depicted using two linear inequality constraints:

$$g_{\text{abs}} = w_\Delta - |\tilde{w}| \geq 0 \quad \Leftrightarrow \quad \mathbf{h}_{\text{abs}} = \begin{bmatrix} w_\Delta - \tilde{w} \\ w_\Delta + \tilde{w} \end{bmatrix} \geq \mathbf{0}. \tag{5.9a}$$

Furthermore, the linearisation of the AVC based on the result of the preceding iteration $w^\star_{\text{prev}}$ is added as an additional objective term in form of an exact penalty function with penalty weight τ:

$$J_g \coloneqq \tau\, l_{\text{abs}} \quad \text{with} \tag{5.9b}$$

$$l_{\text{abs}} = w_\Delta - \sigma_w\, \tilde{w} = 0 \quad \text{and} \quad \sigma_w \coloneqq \operatorname{sign}(\tilde{w}^\star) = \operatorname{sign}(w^\star_{\text{prev}} - w_{\text{tr}}). \tag{5.9c}$$

Considering (5.9c), the linearisation breaks down to choosing the sign σ_w of the absolute value based on the previous solution $w^\star_{\text{prev}}$. Furthermore, the projection of the previous solution onto the constraints in (5.7) follows from (5.6) and provides the initial solution, marked as $\breve{(\cdot)}$, for the subsequent iteration:

$$\breve{w} = w^\star_{\text{prev}}, \quad \breve{w}_{\text{low}} = \min(w^\star_{\text{prev}}, w_{\text{tr}}), \quad \breve{w}_{\text{up}} = \max(w^\star_{\text{prev}}, w_{\text{tr}}). \tag{5.10}$$

The projection step is depicted in Fig. 5.2c for an initial guess with a different sign than the computed solution: $\operatorname{sign}(w^\star - w_{\text{tr}}) \neq \operatorname{sign}(w^\star_{\text{prev}} - w_{\text{tr}})$. For a graphical interpretation, the additional cost term (5.9b) can be replaced by an additional relaxed equality constraint with the slack variable $s_w \geq 0$ being minimised via a penalty objective:

$$w_\Delta - \sigma_w\, \tilde{w} = s_w \tag{5.11a}$$

$$J_s \coloneqq \tau\, s_w. \tag{5.11b}$$

Since adding slack variables increases the number of decision variables, this is not desirable from an implementation standpoint; however, it fosters comprehension. The relaxation of constraint (5.11a) via the slack variable s_w is necessary to avoid infeasibility, which is illustrated in Fig. 5.2b: Since $g_{\text{abs}} = w_\Delta - |\tilde{w}| \geq 0$ holds, an unrelaxed

constraint (5.11a) with $s_w \equiv 0$ would only allow values satisfying $\tilde{w} \leq 0$ for $\sigma_w = -1$ and only values $\tilde{w} \geq 0$ for $\sigma_w = 1$. However, the sign is based on the previous iteration and can thus be wrong. Then, a relaxation represented by the slack variable is necessary to allow values $\tilde{w}$ of the opposite region and avoid infeasibility. The sign is corrected for the subsequent iteration step to enable eradicating the additional objective (5.11b) and thus the slack variable. This concludes that the initial solution does not have to represent a feasible solution, which results in robust initialisation behaviour. The relaxation via the slack variable s_w in (5.11a) corresponds to the violation of constraint l_{abs} via (5.9b). The fact that the constraint violation will be large in case of a wrong sign is used as feedback for the algorithm. The algorithm will iterate until a feasible point satisfying $l_{\text{abs}} \approx 0 \approx s_w$ is reached. Then, the additional objective (5.9b) vanishes.

5.2.2 Piecewise Linear Approximation of Optimisation Problem

The first step towards a convex OP is the derivation of a piecewise linear approximation of the original problem (5.3). This is achieved by using the concept of space splitting presented in the previous section. Since multiple decision variables can be split, the formulation of the OP is posed using the augmented vector of decision variables

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_0^T & \mathbf{z}_{\frac{1}{2}}^T & \mathbf{z}_1^T & \dots & \mathbf{z}_{n_{\text{seg}}}^T \end{bmatrix}^T \quad \text{with} \quad \mathbf{z}_k^T \coloneqq \begin{bmatrix} \mathbf{x}_k^T & \mathbf{u}_k^T & \mathbf{w}_{\text{aux,k}}^T \end{bmatrix} \in \mathbb{R}^{1 \times n_z} \tag{5.12}$$

with the auxiliary decision variables $\mathbf{w}_{\text{aux,k}} \in \mathbb{R}^{n_s}$. Therein some of the original states and inputs are excluded when they are depicted using affine transformations as illustrated in Section 5.3.1. The corresponding substitute problem is given by

$$\begin{aligned}
\min_{\mathbf{z}} \quad & J(\mathbf{w}) && && \text{(5.13a)}\\
\text{s.t.} \quad & \hat{\mathbf{c}}_{\text{coll},i}(\mathbf{z}) = \mathbf{c}_{\text{coll},i}(\mathbf{w} = \mathbf{\Phi}(\mathbf{z}), \mathbf{f}_{\text{pwl}} = \mathbf{\Phi}_{\text{pwl}}) = \mathbf{0} && \forall i \in \mathcal{I}_{\text{coll}}, && \text{(5.13b)}\\
& \hat{\mathbf{c}}_k(\mathbf{z}) = \mathbf{c}_k(\mathbf{w} = \mathbf{\Phi}(\mathbf{z})) = \mathbf{0} && \forall k \in \mathcal{K}, && \text{(5.13c)}\\
& \hat{\mathbf{h}}_k(\mathbf{z}) = \mathbf{h}_k(\mathbf{w} = \mathbf{\Phi}(\mathbf{z})) \leq \mathbf{0} && \forall k \in \mathcal{K}, && \text{(5.13d)}\\
& \hat{\mathbf{h}}_{\text{zcs},k}(\mathbf{z}) \leq \mathbf{0} && \forall k \in \mathcal{K}, && \text{(5.13e)}\\
& \mathbf{g}_{\text{aff},k}(\mathbf{z}) \coloneqq \mathbf{E}_{\Sigma,k}\, \mathbf{w}_{\text{aux},k} - (\mathbf{E}_{z,k}\, \mathbf{z}_k + \mathbf{w}_{\text{tr}}) = \mathbf{0} && \forall k \in \mathcal{K}, && \text{(5.13f)}\\
& \mathbf{g}_{\text{abs},k}(\mathbf{z}) \coloneqq \mathbf{E}_{\Delta,k}\, \mathbf{w}_{\text{aux},k} - \alpha(\underbrace{\mathbf{E}_{z,k}\, \mathbf{z}_k - \mathbf{w}_{\text{tr}}}_{=:\tilde{\mathbf{z}}_k}) = \mathbf{0} && \forall k \in \mathcal{K}. && \text{(5.13g)}
\end{aligned}$$

Therein $\alpha(\cdot) : \mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$ represents a function that applies the smoothable absolute value function (3.2) to each vector component individually: No smoothing occurs for $\varepsilon_s = 0$. Furthermore, the right-hand side of the system equations (5.5) is depicted or approximated by the affine expression

$$\mathbf{f}_\Phi(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) = (\mathbf{A}\, \mathbf{\Phi}_x(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) + \mathbf{B}\, \mathbf{\Phi}_u(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) + \mathbf{k}) + \mathbf{\Phi}_{\text{pwl}}(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) \tag{5.14}$$

with $\mathbf{\Phi}_u : \mathbb{R}^{n_z} \to \mathbb{R}^{n_u}$ and $\mathbf{\Phi}_x, \mathbf{\Phi}_{\text{pwl}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$ representing functions that are affine in the decision variables. Thus, (5.14) enables affine collocation constraints (5.13b). The

transformation function $\mathbf{\Phi}(\mathbf{z})$ in problem (5.13) represents the accumulation of the affine transformations $\mathbf{\Phi}_x$ and $\mathbf{\Phi}_u$ with $n_\Phi = (n_x + n_u)\, n_{\text{coll}}$ yielding

$$\mathbf{w} = \mathbf{\Phi}(\mathbf{z}) = \begin{bmatrix} \mathbf{\Phi}_0 \\ \vdots \\ \mathbf{\Phi}_{n_{\text{seg}}} \end{bmatrix} \coloneqq \hat{\mathbf{A}}\,\mathbf{z} + \hat{\mathbf{b}} \quad \text{with} \quad \mathbf{\Phi}_k \coloneqq \begin{bmatrix} \mathbf{\Phi}_x|_k \\ \mathbf{\Phi}_u|_k \end{bmatrix}, \ \hat{\mathbf{A}} \in \mathbb{R}^{n_\Phi \times n_\Phi}, \ \hat{\mathbf{b}} \in \mathbb{R}^{n_\Phi}. \quad (5.15)$$

Inserting the affine mapping (5.15) into the affine constraints (5.3c) and (5.3d) results in the affine constraints (5.13c) and (5.13d), respectively [19, p.79]. Furthermore, the auxiliary variables are used to represent the ZCSs (5.3e) using convex functions $\hat{\mathbf{h}}_{\text{zcs}}$ in (5.13e). In order to get a QP problem, these constraints are assumed to be affine or approximated by affine inequality constraints. The application of space splitting for the convexification of the original constraints will be shown in more detail in Section 5.3. The splitting constraints (5.13f) and (5.13g) represent the constraints (5.7a) and (5.7b), respectively. These constraints are necessary to ensure that problem (5.13) correctly represents the original problem (5.3). They implement a bisection of selected decision variables at predefined transition values that are stored in the vector $\mathbf{w}_{\text{tr}}$. The matrices $\mathbf{E}_{\Sigma,k}$, $\mathbf{E}_{\Delta,k}$ and $\mathbf{E}_{z,k}$ are selection matrices extracting the desired decision variables. It is assumed that $\frac{n_s}{2}$ decision variables are split at each collocation point.
Depending on the individual approximations of the nonlinear constraints, the piecewise linear representation is subject to some error. The approximation error of this substitute problem is defined beforehand by the implemented number of space subdivisions. If the original problem is piecewise linear, this substitute problem can be an exact representation.

Remark 1. *Let $f_{nl}(w)$ represent a nonlinearity within a constraint depending on the decision variable representative w, as depicted in Fig. 5.3. Then, the total absolute error of the piecewise linear approximation $p_{pwl}(w)$ is given by the sum of absolute errors at*

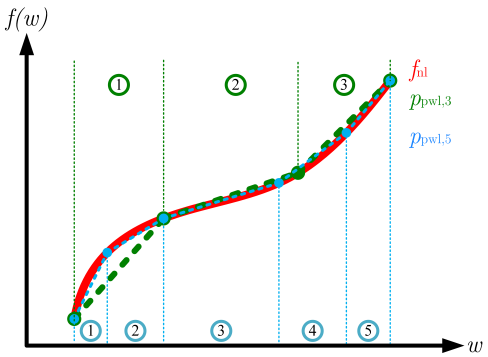


Figure 5.3: Approximation of nonlinearity f_{nl} with three-/five-segmented piecewise linear polynom $p_{\text{pwl},3}/p_{\text{pwl},5}$. Approximation error can be reduced with increasing number of splitting segments.

each collocation point of the decision variable:

$$e_{pwl} = \sum_{k \in \mathcal{K}} |f_{nl}(w_k) - p_{pwl}(w_k)|. \tag{5.16}$$

An adequate number and position of knot points for the segmentation into piecewise linear parts achieve sufficiently small approximation errors. However, a higher number of splitting segments increases the size of the OP influencing the computation time.

5.2.3 Nonconvex Optimisation Problem with Convex Feasible Set

The absolute values in equality constraint (5.13g) still prohibit convexity. Deploying the theory of exact penalty functions [46, 75], the absolute value equality constraints are moved to the objective without compromising optimality:

Theorem 5.2.2. *Let $\overline{\mathbf{z}}$ be a stationary point of the OP*

$$\min_{\mathbf{z}} \quad J(\mathbf{w}) + \tau \sum_{k \in \mathcal{K}} \|\mathbf{g}_{\text{abs},k}\|_1 = J(\mathbf{w}) + \tau \sum_{k \in \mathcal{K}} \sum_{j=1}^{n_s} g_{\text{abs},k,j} \tag{5.17a}$$

$$\text{s.t.} \quad \begin{bmatrix} \hat{\mathbf{c}}_{\text{coll},i}^T & \hat{\mathbf{c}}_k^T & \mathbf{g}_{\text{aff},k}^T \end{bmatrix} = \mathbf{0}^T \qquad \forall i \in \mathcal{I}_{coll},\ k \in \mathcal{K}, \tag{5.17b}$$

$$\begin{bmatrix} \hat{\mathbf{h}}_k^T & \hat{\mathbf{h}}_{\text{zcs},k}^T \end{bmatrix} \leq \mathbf{0}^T \qquad \forall k \in \mathcal{K}, \tag{5.17c}$$

$$\mathbf{g}_{\text{abs},k} \geq \mathbf{0} \qquad \forall k \in \mathcal{K} \tag{5.17d}$$

with $\boldsymbol{\lambda}$ representing the Lagrangian multipliers of the corresponding equality constraints. Since a $\mathcal{L}_1$-norm is an exact penalty function, $\overline{\mathbf{z}}$ is a critical point of problem (5.13) *for $\varepsilon_s > 0$ and sufficiently large but finite penalty weights $\tau \geq \|\boldsymbol{\lambda}\|_\infty$.*

Proof. The prerequisite for the optima recovery is that the optimality conditions are admissible for problem (5.13) [148, p.507]. For this purpose, the LICQ (2.8) is analysed subsequently. Since the transformation $\boldsymbol{\Phi}(\mathbf{z})$ in (5.15) is affine, the gradients of the transformed basic constraints (5.13c) and (5.13d) remain linearly independent if the original constraints $\mathbf{c}(\mathbf{w}) : \mathbb{R}^{n_w} \to \mathbb{R}^{n_c}$ and $\mathbf{h}(\mathbf{w}) : \mathbb{R}^{n_w} \to \mathbb{R}^{n_h}$ are linearly independent:

$$\nabla_{\mathbf{z}} \hat{c}_i(\mathbf{z}) = \left(\frac{\partial c_i(\boldsymbol{\Phi}(\mathbf{z}))}{\partial \boldsymbol{\Phi}} \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{z}} \right)^T = \hat{\mathbf{A}}^T \nabla_{\mathbf{w}} c_i(\mathbf{w} = \boldsymbol{\Phi}(\mathbf{z})) \quad \forall i \in 1, \ldots, n_c \tag{5.18a}$$

$$\nabla_{\mathbf{z}} \hat{h_j}(\mathbf{z}) = \left(\frac{\partial h_j(\boldsymbol{\Phi}(\mathbf{z}))}{\partial \boldsymbol{\Phi}} \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{z}} \right)^T = \hat{\mathbf{A}}^T \nabla_{\mathbf{w}} h_j(\mathbf{w} = \boldsymbol{\Phi}(\mathbf{z})) \quad \forall j \in 1, \ldots, n_h. \tag{5.18b}$$

The gradients of the splitting constraints (5.13f) and (5.13g) are by definition linearly independent from each other and from the remaining constraints. Furthermore, it can be assumed without loss of generality that the gradients of the collocation constraints (5.13b), of the basic constraints (5.13c)-(5.13d) and of the ZCS-constraints (5.13e) are linearly independent. Thus, if the LICQ holds for the original problem (5.3), then it is also satisfied by the problem (5.13). By choosing a sufficiently

small but positive smoothing parameter $0 < \varepsilon_s \ll 1$, the prerequisite of continuous differentiable objective and constraint functions is given while approximation errors are kept small. Thus, the optimal solution will satisfy the Karush-Kuhn-Tucker conditions [148, p.321]. ■

Constraints (5.17d) are added to enable omitting the norm in the objective as illustrated in (5.17a). Since $\mathbf{g}_{\mathrm{abs},k}$ is a vector of concave functions, the AVCs (5.17d) represent a convex set. Thus, the feasible set of OP (5.17) is convex.

5.2.4 Convex Optimisation Problem for Iterative Solution

From Theorem 5.2.2 follows that solving (5.17) provides the solution for problem (5.13) assuming $\varepsilon_s > 0$. Unfortunately, the augmented objective (5.17a) is nonconvex, which will be addressed via linearisation. Subsequently, the smoothing is removed by setting $\varepsilon_s = 0$ since it will not be required for continuous differentiability. Omitting the smoothing represents an approximation to the smoothed version of (5.17); however, it is beneficial from a practical standpoint since the splitting constraints are exact for $\varepsilon_s = 0$. Furthermore, it enables representing the AVCs (5.17d) by two affine inequalities, which yields exclusively affine constraints required for an LP and QP problem. For a convex OP, the absolute value of the AVCs in the augmented objective is linearised around the points $\tilde{\mathbf{z}}_k^\star \in \mathbb{R}^{n_s}$ yielding the linearised constraints

$$\mathbf{l}_{\mathrm{abs},k}(\mathbf{z}, \tilde{\mathbf{z}}_k^\star) \coloneqq \mathbf{E}_{\Delta,k}\, \mathbf{w}_{\mathrm{aux},k} - \boldsymbol{\Sigma}_k\, \tilde{\mathbf{z}}_k = \mathbf{0} \qquad \forall k \in \mathcal{K}, \tag{5.19a}$$

which represent (5.9c), with the linearisation points being stored in matrix

$$\boldsymbol{\Sigma} \coloneqq \begin{bmatrix} \boldsymbol{\Sigma}_0 & \boldsymbol{\Sigma}_{\frac{1}{2}} & \dots & \boldsymbol{\Sigma}_{n_{\mathrm{seg}}} \end{bmatrix} \text{ with } \boldsymbol{\Sigma}_k \coloneqq \operatorname{diag}\Big(\operatorname{sign}(\tilde{z}_{k,1}^\star), \dots, \operatorname{sign}(\tilde{z}_{k,n_s}^\star)\Big). \tag{5.19b}$$

Comparing (5.13g) and (5.19a) shows that the linearisation breaks down to a binary decision: choosing the correct sign of the absolute value term $\tilde{\mathbf{z}}_k$. The linearisation (5.19a) is affine in the decision variables. Since the sum of convex functions preserves convexity [19, p.79], the composite objective function is then convex in the decision variables. With all constraints being affine, using linearisation (5.19a) results in the convex QP problem

$$\min_{\mathbf{z}} \quad J_{\mathrm{tot}}(\mathbf{z}, \mathbf{z}^\star) \coloneqq J(\mathbf{w}) + \tau \sum_{k \in \mathcal{K}} \sum_{j=1}^{n_s} l_{\mathrm{abs},k,j} \tag{5.20a}$$

$$\text{s.t.} \quad \hat{\mathbf{c}}_{\mathrm{coll},i} = \mathbf{0} \qquad \forall i \in \mathcal{I}_{\mathrm{coll}}, \tag{5.20b}$$

$$\mathbf{c}_{\mathrm{tot},k}^T(\mathbf{z}) \coloneqq \begin{bmatrix} \hat{\mathbf{c}}_k^T & \mathbf{g}_{\mathrm{aff},k}^T \end{bmatrix} = \mathbf{0}^T \qquad \forall k \in \mathcal{K}, \tag{5.20c}$$

$$\mathbf{h}_{\mathrm{tot},k}^T(\mathbf{z}) \coloneqq \begin{bmatrix} \hat{\mathbf{h}}_k^T & \hat{\mathbf{h}}_{\mathrm{zcs},k}^T \end{bmatrix} \leq \mathbf{0}^T \qquad \forall k \in \mathcal{K}, \tag{5.20d}$$

$$\mathbf{h}_{\mathrm{abs},k}(\mathbf{z}) \coloneqq \begin{bmatrix} \mathbf{E}_{\Delta,k}\, \mathbf{w}_{\mathrm{aux},k} - \tilde{\mathbf{z}}_k \\ \mathbf{E}_{\Delta,k}\, \mathbf{w}_{\mathrm{aux},k} + \tilde{\mathbf{z}}_k \end{bmatrix} \geq \mathbf{0} \qquad \forall k \in \mathcal{K}. \tag{5.20e}$$

For linear objectives $J(\mathbf{w})$, the SSC approach results in an LP problem. The convex problem (5.20) is iteratively solved whereas the linearisation points in $\boldsymbol{\Sigma}$ are updated based on the solution $\mathbf{z}^{\star T} = \begin{bmatrix} \mathbf{z}_0^{\star T} & \dots & \mathbf{z}_{n_{\text{seg}}}^{\star T} \end{bmatrix}$ of the preceding iteration yielding

$$\tilde{\mathbf{z}}_k^\star = \mathbf{E}_{z,k}\, \mathbf{z}_k^\star - \mathbf{w}_{\text{tr}} \quad \forall k \in \mathcal{K}. \tag{5.21}$$

However, the question arises if replacing the constraint $\mathbf{g}_{\text{abs}}$ in the objective with its linearisation $\mathbf{l}_{\text{abs}}$ falsifies the solution. This motivates following lemma:

Lemma 5.2.3. *The satisfaction of* $\mathbf{h}_{\text{abs},k}(\mathbf{z}) \geq \mathbf{0}$ *and* $\mathbf{l}_{\text{abs},k}(\mathbf{z}, \tilde{\mathbf{z}}_k^\star) = \mathbf{0}$ *concludes* $\mathbf{g}_{\text{abs},k}(\mathbf{z}) = \mathbf{0}$*, independently of the linearisation point* $\tilde{\mathbf{z}}_k^\star$*.*

Proof. This can be verified using Fig. 5.2b as graphical support. For this purpose, the following equations are given in general notation and are compared with the equations from the illustrative example in Section 5.2.1 using the vector of decision variables $\mathbf{z}_{\text{ex}} = \begin{bmatrix} w & w_{\text{up}} & w_{\text{low}} \end{bmatrix}^T$. The nonsmoothed absolute value constraint $g_{\text{abs},k,j}$ spans the following set of points:

$$\mathcal{G}_{k,j}(\mathbf{z}) \coloneqq \left\{ \mathbf{z} \mid \mathbf{e}_{\Delta,k}^T\, \mathbf{w}_{\text{aux},k} - |\tilde{z}_{k,j}| = 0 \right\} \mathrel{\widehat{=}} \left\{ \mathbf{z}_{\text{ex}} \mid w_\Delta - |\tilde{w}| = 0 \right\} \tag{5.22a}$$

with $\mathbf{e}_{\Delta,k}^T$ representing the j^{th} row of matrix $\mathbf{E}_{\Delta,k}$. The admissible points for the AVC-linearisation $l_{\text{abs},k,j}$ are given by

$$\mathcal{L}_{k,j}(\mathbf{z}, \tilde{z}_{k,j}^\star) \coloneqq \left\{ \mathbf{z} \mid \mathbf{e}_{\Delta,k}^T\, \mathbf{w}_{\text{aux},k} - \operatorname{sign}(\tilde{z}_{k,j}^\star)\, \tilde{z}_{k,j} = 0 \right\} \mathrel{\widehat{=}} \left\{ \mathbf{z}_{\text{ex}} \mid w_\Delta - \sigma_w\, \tilde{w} = 0 \right\} \tag{5.22b}$$

representing the infinite extension of the positive and negative branch of the AVC for $\tilde{z}_{k,j}^\star, \sigma_w \geq 0$ and $\tilde{z}_{k,j}^\star, \sigma_w \leq 0$, respectively. Then, the intersection with the admissible points of the inequality constraint $\mathbf{h}_{\text{abs},k,j}$

$$\mathcal{H}_{k,j}(\mathbf{z}) \coloneqq \left\{ \mathbf{z} \;\middle|\; \begin{bmatrix} \mathbf{e}_{\Delta,k}^T\, \mathbf{w}_{\text{aux},k} - \tilde{z}_{k,j} \\ \mathbf{e}_{\Delta,k}^T\, \mathbf{w}_{\text{aux},k} + \tilde{z}_{k,j} \end{bmatrix} \geq \mathbf{0} \right\} \mathrel{\widehat{=}} \left\{ \mathbf{z}_{\text{ex}} \;\middle|\; \begin{bmatrix} w_\Delta - \tilde{w} \\ w_\Delta + \tilde{w} \end{bmatrix} \geq \mathbf{0} \right\} \tag{5.22c}$$

provides the sets

$$\mathcal{N}_{k,j}(\mathbf{z}) \coloneqq \left\{ \mathbf{z} \mid \mathbf{z} \in \mathcal{H}_{k,j},\ \mathbf{z} \in \mathcal{L}_{k,j}(\mathbf{z}, \tilde{z}_{k,j}^\star)\big|_{\tilde{z}_{k,j}^\star \leq 0} \right\} \tag{5.22d}$$

$$\mathcal{P}_{k,j}(\mathbf{z}) \coloneqq \left\{ \mathbf{z} \mid \mathbf{z} \in \mathcal{H}_{k,j},\ \mathbf{z} \in \mathcal{L}_{k,j}(\mathbf{z}, \tilde{z}_{k,j}^\star)\big|_{\tilde{z}_{k,j}^\star \geq 0} \right\}, \tag{5.22e}$$

which represent the negative and positive branch of the AVC and are thus a subset: $\mathcal{N}_{k,j},\ \mathcal{P}_{k,j} \subset \mathcal{G}_{k,j}$. ■

This important lemma confirms that the solution to which problem (5.20) converges, which satisfies the linearised constraints $\mathbf{l}_{\text{abs},k}(\mathbf{z}, \tilde{\mathbf{z}}_k^\star) = \mathbf{0}$, also satisfies the original nonlinear constraints $\mathbf{g}_{\text{abs},k}(\mathbf{z}) = \mathbf{0}$ in (5.13g).

5.2.5 Space Splitting Convexification Algorithm

Subsequently, the SSC algorithm 2 is presented in detail and the proof of its convergence is provided. The algorithm inputs are described in line 1. An initial solution for the inputs $\mathbf{U}_0^\star \in \mathbb{R}^{n_u \times n_{\text{coll}}}$ and states $\mathbf{X}_0^\star \in \mathbb{R}^{n_x \times n_{\text{coll}}}$ must be provided. The algorithm iteratively solves the convex problem (5.20) using the solution of the previous iterate to correct the signs of the linearisation if necessary. The algorithm terminates when all signs are chosen correctly or the maximum number of iterations $q_{\max}$ is reached. Thus, if the algorithm converges before reaching the maximum number of iterations $q < q_{\max}$, the largest constraint violation is smaller than a small value $\bar{l}_{\text{abs}} \leq \varepsilon_g$ with $0 \leq \varepsilon_g \ll 1$. As already mentioned, the violation of the linearised AVCs indicates that a wrong sign was chosen. The reason why the constraint violations are a suitable feedback for correct sign selection can be visualised using Fig. 5.2b. The signs and thus the linearised constraints $l_{\text{abs},k,j}$ in the objective (5.20a) are fixed before optimisation. Then, running into the region of $\tilde{w} \widehat{=} \tilde{z}_{k,j}$ with the opposite sign requires increasing $w_\Delta \widehat{=} \mathbf{e}_{\Delta,k}^T \mathbf{w}_{\text{aux},k}$ due to the constraints $\mathbf{h}_{\text{abs}} \widehat{=} \mathbf{h}_{\text{abs},k} \geq \mathbf{0}$, which in turn increases the linearised constraints in the objective. Based on the solution of the preceding iteration, the updating routine in line 7 adjusts the convex QP problem (5.20) to satisfy (5.13f)-(5.13g) using the projection procedure (5.10): The update routine computes the projected initial solution $\breve{\mathbf{Z}}$ and the correct signs $\mathbf{\Sigma}$ for the linearisation of the AVCs. Afterwards, the optimiser is invoked in line 8 to solve the convex QP problem or LP problem if the main objective is linear. The theory of exact penalty functions states that the penalty parameter τ must be larger than the largest optimal dual variable of the corresponding constraints.

Algorithm 2 SSC: a procedure for iterative convex optimisation.

1: **INPUTS**:
- $\mathbf{U}_0^\star, \mathbf{X}_0^\star$: initial solution for input and state decision variables
- $q_{\max}$: maximum number of iterations
- ε_g: error tolerance for linearised AVCs
- $\underline{\tau}, \overline{\tau}$: weight bounds for penalty objectives

2: **OUTPUT**: $\mathbf{U}^\star, \mathbf{X}^\star$: system solution within tolerance ε_g
3: **MAIN FUNCTION**:
4: $q = 1, \quad \bar{l}_{\text{abs}} = \infty$
5: **while** $q \leq q_{\max}$ and $\bar{l}_{\text{abs}} > \varepsilon_g$ **do**
6: $\quad \tau = \frac{\overline{\tau} - \underline{\tau}}{q_{\max}} \cdot q + \underline{\tau}$ // weight for additional objectives
7: $\quad [\mathbf{\Sigma}, \breve{\mathbf{Z}}] = \textbf{updateQP}(\mathbf{U}_{q-1}^\star, \mathbf{X}_{q-1}^\star)$ // linearisation and initialisation
8: $\quad [\mathbf{U}_q^\star, \mathbf{X}_q^\star] = \textbf{solveQP}(\tau, \mathbf{\Sigma}, \breve{\mathbf{Z}})$
9: $\quad \bar{l}_{\text{abs}} = \left\| \mathbf{l}_{\text{abs},0} \quad \mathbf{l}_{\text{abs},\frac{1}{2}} \quad \cdots \quad \mathbf{l}_{\text{abs},n_{\text{seg}}} \right\|_{\max}$ // maximum value of all relaxations
10: $\quad q = q + 1$
11: $\mathbf{U}^\star = \mathbf{U}_q^\star, \quad \mathbf{X}^\star = \mathbf{X}_q^\star$ // output final solution

Since estimating the limit value for the penalty parameter is a rather complicated task, Algorithm 2 pursues the straightforward approach in line 6 of gradually increasing the penalty parameter up to a high value. A reasonably high but finite upper bound on the penalty parameter $\overline{\tau}$ avoids numerical problems. Although this is not the best updating approach, it can work well in practice [148, p.511] as it is the case for the examples considered in this thesis.
Finally, following theorem verifies that the algorithm produces a sequence of solutions that eventually converges to a local solution satisfying the AVCs:

Theorem 5.2.4. *Let the feasible set of problem* (5.20) *be denoted by the convex set*

$$\Omega_f \coloneqq \left\{\mathbf{z} \middle| \hat{\mathbf{c}}_{coll,i} = \mathbf{0}, \mathbf{c}_{\mathrm{tot},k}(\mathbf{z}) = \mathbf{0}, \mathbf{h}_{\mathrm{tot},k}(\mathbf{z}) \leq \mathbf{0}, \mathbf{h}_{\mathrm{abs},k}(\mathbf{z}) \geq \mathbf{0}, i \in \mathcal{I}_{coll}, k \in \mathcal{K}\right\}. \quad (5.23)$$

Starting from a point $\mathbf{z}^{(0)}$*, Algorithm 2 generates a sequence* $\left\{\mathbf{z}^{(q)}\right\}$ *according to*

$$\mathbf{z}^{(q+1)} = \min_{\mathbf{z}} \left\{ J_{\mathrm{tot}}\left(\mathbf{z}, \mathbf{z}^{(q)}\right) \middle| \mathbf{z} \in \Omega_f \right\} \quad q = 0, 1, \ldots \quad (5.24)$$

with J_{tot} *being a convex function. This sequence eventually converges to a limit point* $\mathbf{z}^\star$ *that satisfies* $\mathbf{g}_{\mathrm{abs},k} = \mathbf{0}\ \forall k \in \mathcal{K}$*.*

Proof. Algorithm 2 iteratively solves problem (5.20) and gradually increases the penalty parameter up to a finite value $\tau \leq \overline{\tau}$. Let the condition $\tau \geq \|\boldsymbol{\lambda}\|_\infty$ mentioned in Theorem 5.2.2 hold for $q \geq \mathcal{Q}_\lambda \in \mathbb{N}$. Then, the theory of exact penalty functions states that $\mathbf{l}_{\mathrm{abs},k} = \mathbf{0}\ \forall k \in \mathcal{K}$ is satisfied for all $q \geq \mathcal{Q}_\lambda$. As described in Lemma 5.2.3, this concludes that only solutions on one branch of the AVCs are admissible. Thus, no sign changes of $\tilde{z}_{k,j} \widehat{=} \tilde{w}$ are possible for $q \geq \mathcal{Q}_\lambda$. This concludes that the linearisation points of $\mathbf{l}_{\mathrm{abs},k}\ \forall k \in \mathcal{K}$ and thus OP (5.20) remain unchanged for $q \geq \mathcal{Q}_\lambda$. The convexity of the problem entails that only one optimum exists yielding

$$\mathbf{z}^{(q+1)} - \mathbf{z}^{(q)} = \mathbf{0} \quad \Rightarrow \quad \left\|\mathbf{z}^{(q+1)} - \mathbf{z}^{(q)}\right\| = 0 \quad \forall q \geq \mathcal{Q}_\lambda. \quad (5.25)$$

This concludes that for every $\varepsilon_\tau > 0$, there is a $\mathcal{Q} \in \mathbb{N}$ such that

$$\|\Delta\mathbf{z}\| \coloneqq \left\|\mathbf{z}^{(n)} - \mathbf{z}^{(m)}\right\| < \varepsilon_\tau \quad \text{for every} \quad n, m \geq \mathcal{Q} \quad (5.26)$$

since (5.26) holds at the latest from the value $\mathcal{Q} = \mathcal{Q}_\lambda$ with $\|\Delta\mathbf{z}\| = 0 < \varepsilon_\tau$. This proves that (5.24) is a Cauchy sequence and thus converges [232, p.85]. Hence, the solutions of the individual iterations can oscillate; however, this oscillation vanishes with increasing number of iterations finally converging to a single point $\mathbf{z}^\star$. Due to Lemma 5.2.3, $\mathbf{l}_{\mathrm{abs},k} = \mathbf{0}\ \forall k \in \mathcal{K}$ concludes that this point satisfies $\mathbf{g}_{\mathrm{abs},k} = \mathbf{0}\ \forall k \in \mathcal{K}$. ■

Theorem 5.2.4 concludes that the iterative solution of OP (5.20) within the SSC algorithm basically solves the OP (5.13) with $\varepsilon_s = 0$: From convergence to $\mathbf{l}_{\mathrm{abs},k} = \mathbf{0}\ \forall k \in \mathcal{K}$ follows that the computed solution satisfies $\mathbf{g}_{\mathrm{abs},k} = \mathbf{0} = \mathbf{h}_{\mathrm{abs},k}\ \forall k \in \mathcal{K}$ and the augmented objective term vanishes. The remaining constraints of problems (5.13) and (5.20) are identical. Thus, the SSC algorithm computes a solution that is arbitrarily close, but

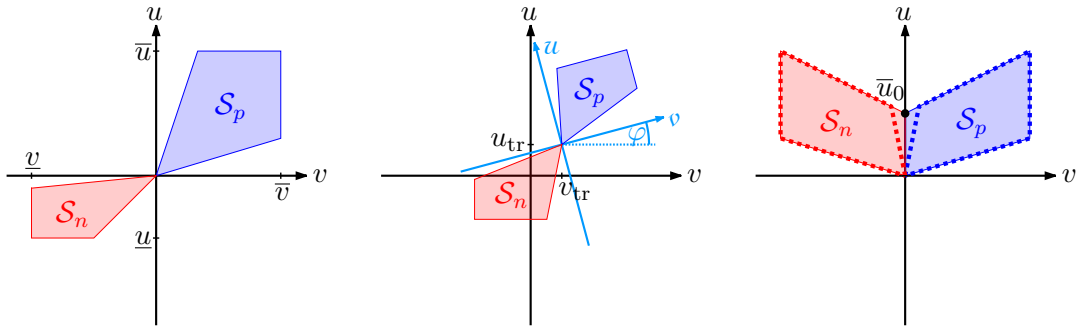
not necessarily identical, to the solution of the piecewise linearly approximated problem (5.13). The potential discrepancy in solutions is rooted in the necessity to smoothen problem (5.13) with $\varepsilon_s > 0$ for the admissibility of the optimality conditions in Theorem 5.2.2. Since the linearisation removes the discontinuity, the smoothing is disregarded within the SSC algorithm. From a practical standpoint, the discrepancy is negligible since the smoothing parameter can be assumed arbitrarily small $\varepsilon_s \to 0$ making the impact of the smoothing numerically irrelevant. This procedure for the approximation of discontinuities in OCPs is common practice, see e.g. [158]. In summary, the SSC approach chooses one branch of the discontinuity, resulting in the continuously differentiable problem (5.20) with $\varepsilon_s = 0$, and switches in the following iterations to the other branch if the wrong side of the discontinuity was selected.

5.3 Constraint Convexification via Space Splitting

The previous sections derived the basic structure of the convex OP (5.20), which is solved in an iterative manner. However, the question of how the original constraints can be convexified via space splitting to gain the convex constraints (5.13b)-(5.13e) is still open. The basic procedure of bisecting the space of a variable into two subdomains has been illustrated in Section 5.2.1. This can be employed to consider ZCSs and equality constraints with univariate nonlinearities in a convex manner, which is demonstrated in Section 5.3.1 and Section 5.3.2, respectively. Subsequently, the index for the collocation discretisation $k \in \mathcal{K}$ is omitted for readability.

5.3.1 Convexification of Zonally Convex Sets

One prominent physical example of ZCSs are constraints ensuring semi-activity. Semi-active actuators like limited slip differentials and semi-active dampers are used in many engineering applications due to their good compromise between low energy consumption and high performance [179, 184, 185]. A typical ZCS for a semi-active damper is depicted in Fig. 5.4a with damper force $F_d = u$ and damper velocity v. The nonconvexity of the



(a) ZCS with single transition point. **(b)** Shifted, rotated ZCS with single transition point. **(c)** ZCS with shared line at transition.

Figure 5.4: Space splitting convexification of zonally convex sets.

set is apparent considering that it is easy to draw a line that contains non-admissible points and connects an admissible point in the first quadrant with an admissible point in the third quadrant. Note that simply linearising the inequality constraints depicting this nonconvex set results in a half-plane, which represents a poor approximation of the set. The main idea for the novel convexification approach is appropriately decomposing nonconvex sets, described in the original decision variables, into convex subsets using auxiliary decision variables. The applicability of the SSC method to the ZCS types depicted in Fig. 5.4 is proven in the subsequent paragraphs.
The approach is initially illustrated using the ZCS depicted in Fig. 5.4a. Let the convex subset in the first and third quadrant be given by

$$\mathcal{S}_p \coloneqq \left\{ (u,v) \;\middle|\; u \in \mathcal{U},\ v \in \mathcal{V},\ \hat{\mathbf{h}}_{\mathrm{zcs},p}(u,v) \leq \mathbf{0} \right\} \quad \text{and} \tag{5.27a}$$

$$\mathcal{S}_n \coloneqq \left\{ (u,v) \;\middle|\; u \in \mathcal{U},\ v \in \mathcal{V},\ \hat{\mathbf{h}}_{\mathrm{zcs},n}(u,v) \leq \mathbf{0} \right\}, \tag{5.27b}$$

respectively. Therein v and u are original decision variables. Moreover, $\hat{\mathbf{h}}_{\mathrm{zcs},p} \leq \mathbf{0}$ and $\hat{\mathbf{h}}_{\mathrm{zcs},n} \leq \mathbf{0}$ are convex sets. The union of convex sets does not necessarily result in a convex set. As illustrated in Fig. 5.4a, the union $\mathcal{S} = \mathcal{S}_p \cup \mathcal{S}_n$ is assumed to be nonconvex in this thesis. Thus, $\mathcal{S}$ is a nonconvex set comprised of convex subsets: a ZCS. The velocity v is split at $v = v_{\mathrm{tr}} = 0$ by a vertical line following Lemma 5.2.1. The two auxiliary decision variables $v_n \in \mathcal{V}_n \coloneqq [\underline{v}, v_{\mathrm{tr}}] = [\underline{v}, 0]$ and $v_p \in \mathcal{V}_p \coloneqq [v_{\mathrm{tr}}, \overline{v}] = [0, \overline{v}]$ are introduced aiming at implementing

$$v_n = \begin{cases} v & \forall v \leq 0, \\ 0 & \text{else} \end{cases} \quad \text{and} \quad v_p = \begin{cases} v & \forall v \geq 0, \\ 0 & \text{else} \end{cases}. \tag{5.28}$$

Introducing analogous constraints to (5.7a) and (5.9a) as well as additional objective terms in form of (5.9b) ensures the relations in (5.28). Furthermore, the auxiliary variables $u_n \in \mathcal{U}_n \coloneqq [\underline{u}, u_{\mathrm{tr}}]$ and $u_p \in \mathcal{U}_p \coloneqq [u_{\mathrm{tr}}, \overline{u}]$ are introduced for the individual subsets. Therein u_{tr} represents the ordinate value of the common point, which lies on the origin in the current example depicted in Fig. 5.4a resulting in $u_{\mathrm{tr}} = 0$. However, the common point of the subsets is not required to lie on the abscissa or ordinate. The vector of auxiliary decision variables is given by $\mathbf{w}_{\mathrm{aux}} = \begin{bmatrix} v_p & v_n & u_p & u_n \end{bmatrix}$. While the original state variable v remains a decision variable in the problem, the input variable is replaced according to the transformation

$$\Phi_u \coloneqq u_n + u_p - u_{\mathrm{tr}} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ u_n \end{bmatrix} - u_{\mathrm{tr}}, \tag{5.29}$$

which is affine in the decision variables. The convex subsets in terms of the auxiliary decision variables are given by

$$\hat{\mathcal{S}}_p \coloneqq \left\{ (u_p, v_p) \;\middle|\; u_p \in \mathcal{U}_p,\ v_p \in \mathcal{V}_p,\ \hat{\mathbf{h}}_{\mathrm{zcs},p}(u_p, v_p) \leq \mathbf{0} \right\} \tag{5.30a}$$

$$\hat{\mathcal{S}}_n \coloneqq \left\{ (u_n, v_n) \;\middle|\; u_n \in \mathcal{U}_n,\ v_n \in \mathcal{V}_n,\ \hat{\mathbf{h}}_{\mathrm{zcs},n}(u_n, v_n) \leq \mathbf{0} \right\}. \tag{5.30b}$$

Hence, the convex subsets $\hat{\mathcal{S}}_p$ and $\hat{\mathcal{S}}_n$ are now defined for separate decision variables and will be linked via the affine mapping (5.29). This enables the formulation of convex constraints as depicted in problem (5.13). The proof that this procedure produces the same set as $\mathcal{S}$ will be given in Theorem 5.3.1.
For convexity, $\hat{\mathbf{h}}_{\mathrm{zcs},p}$ and $\hat{\mathbf{h}}_{\mathrm{zcs},n}$ in (5.30a) and (5.30b) must be convex functions. As exemplarily illustrated in Fig. 5.4a, the convex subsets are assumed in this thesis to be representable by $n_{\mathrm{zcs}} = n_{\mathrm{zcs},n} + n_{\mathrm{zcs},p}$ affine inequality constraints:

$$\hat{\mathbf{h}}_{\mathrm{zcs},p}(u,v) \coloneqq \mathbf{A}_{\mathrm{zcs},p} \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b}_{\mathrm{zcs},p} \leq \mathbf{0} \text{ with } \mathbf{A}_{\mathrm{zcs},p} \in \mathbb{R}^{n_{\mathrm{zcs},p} \times 2}, \mathbf{b}_{\mathrm{zcs},p} \in \mathbb{R}^{n_{\mathrm{zcs},p}} \tag{5.31a}$$

$$\hat{\mathbf{h}}_{\mathrm{zcs},n}(u,v) \coloneqq \mathbf{A}_{\mathrm{zcs},n} \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b}_{\mathrm{zcs},n} \leq \mathbf{0} \text{ with } \mathbf{A}_{\mathrm{zcs},n} \in \mathbb{R}^{n_{\mathrm{zcs},n} \times 2}, \mathbf{b}_{\mathrm{zcs},n} \in \mathbb{R}^{n_{\mathrm{zcs},n}}. \tag{5.31b}$$

These constraints represent (5.13e) in the convex OP. Although nonlinear, convex functions for $\hat{\mathbf{h}}_{\mathrm{zcs},p}$ and $\hat{\mathbf{h}}_{\mathrm{zcs},n}$ would not prevent convexity, introducing such constraints would prohibit an LP or QP problem, which can be solved especially efficiently. Asymmetric subsets like the one illustrated in Fig. 5.4a can be easily implemented by using varying coefficients in (5.31a) and (5.31b) or by using different domains for $\mathcal{U}_n$ and $\mathcal{U}_p$.

Theorem 5.3.1. *The subsets* (5.30a) *and* (5.30b) *in combination with the affine mapping* (5.29) *depict the set* $\mathcal{S}$ *that is given by the union of the convex subsets* (5.27a) *and* (5.27b).

Proof. For proving that the splitting approach provides sets that depict the original one, the individual cases for the bisected variable space are considered subsequently.
Case 1: $v_{\mathrm{tr}} < v$
The decision variable v is split at $v = v_{\mathrm{tr}}$. Thus, $v_n = v_{\mathrm{tr}}$ holds according to Lemma 5.2.1. This concludes $u_n = u_{\mathrm{tr}}$ since the subsets converge to the common point $(u_{\mathrm{tr}}, v_{\mathrm{tr}})$ when leaving the subset towards the other subset. Lemma 5.2.1 also concludes $v_p = v$. Then, the affine mapping (5.29) is given by $\Phi_u = u_p \in \hat{\mathcal{S}}_p$. For $v > v_{\mathrm{tr}}$, $\mathcal{S} = \mathcal{S}_p$ holds. Since $v_p = v$, $u \in \mathcal{S}_p$ and $\Phi_u \in \hat{\mathcal{S}}_p$ provide the same feasible set. The chain of conclusions for the remaining cases follows the same reasoning but is given below in equations for conciseness.
Case 2: $v < v_{\mathrm{tr}}$

$$\left.\begin{array}{c} v_p = v_{\mathrm{tr}} \;\Rightarrow\; u_p = u_{\mathrm{tr}} \\ v_n = v \end{array}\right\} \;\Rightarrow\; \Phi_u = u_n \in \hat{\mathcal{S}}_n \quad \widehat{=} \quad u \in \mathcal{S} = \mathcal{S}_n. \tag{5.32a}$$

Case 3: $v = v_{\mathrm{tr}}$

$$\left.\begin{array}{l} v_p = v_{\mathrm{tr}} \;\Rightarrow\; u_p = u_{\mathrm{tr}} \\ v_n = v_{\mathrm{tr}} \;\Rightarrow\; u_n = u_{\mathrm{tr}} \end{array}\right\} \;\Rightarrow\; \Phi_u = u_{\mathrm{tr}} \quad \widehat{=} \quad u = u_{\mathrm{tr}}. \tag{5.32b}$$

Thus, inserting the affine mapping (5.29) into the constraints while replacing the original set $\mathcal{S}$ with the sets (5.30a) and (5.30b) yields the same feasible set. ■

The presented proof requires a space splitting of the variable v, which is performed by splitting the convex subsets via a vertical line. Thus, the SSC approach is capable of convexifying two-dimensional ZCSs that can be divided into two convex subsets connected in a single point using any vertical line. Following theorem is added to show that this also holds for any splitting line that is straight but possibly rotated:

Theorem 5.3.2. *Space splitting applied to a rotated, two-dimensional ZCS also results in convex constraints.*

Proof. This can be verified by applying the previously described procedure to a rotated coordinate frame, exemplarily depicted in Fig. 5.4b. The auxiliary decision variables need to split the rotated two-dimensional space spanned by the variables υ and u. Assuming the coordinate system is rotated by the angle φ with counter-clockwise rotations depicting positive rotations, following relations hold between the original coordinates $v - u$ and the rotated coordinates $\upsilon - u$:

$$\begin{bmatrix} \upsilon \\ u \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}}_{=:\mathrm{R}(\varphi)} \begin{bmatrix} v - v_{\mathrm{tr}} \\ u - u_{\mathrm{tr}} \end{bmatrix} \text{ and } \begin{bmatrix} v \\ u \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}}_{=:\mathrm{R}(-\varphi)} \begin{bmatrix} \upsilon \\ u \end{bmatrix} + \begin{bmatrix} v_{\mathrm{tr}} \\ u_{\mathrm{tr}} \end{bmatrix}. \tag{5.33}$$

The vector of auxiliary decision variables is chosen as $\mathbf{w}_{\mathrm{aux}} = \begin{bmatrix} \upsilon_p & \upsilon_n & u_p & u_n \end{bmatrix}$. For a concise notation, the equations below use the abbreviations $\mathrm{c}_\varphi \coloneqq \cos(\varphi)$ and $\mathrm{s}_\varphi \coloneqq \sin(\varphi)$. Constraints (5.31) are analogously formulated on the rotated, auxiliary decision variables: $\hat{\mathbf{h}}_{\mathrm{zcs},p}(u_p, \upsilon_p)$ and $\hat{\mathbf{h}}_{\mathrm{zcs},n}(u_n, \upsilon_n)$. However, an adjustment using (5.33) is necessary for the constraints (5.7a), (5.9a) and (5.9c) as well as the input mapping (5.29):

$$g_{\mathrm{aff}} = (\upsilon_p + \upsilon_n) - \Big[\underbrace{\mathrm{c}_\varphi \left(v - v_{\mathrm{tr}}\right) + \mathrm{s}_\varphi \Big(\mathrm{s}_\varphi \left(\upsilon_p + \upsilon_n\right) + \mathrm{c}_\varphi (u_p + u_n) \Big)}_{=:\tilde{\upsilon}} \Big] = 0 \tag{5.34a}$$

$$g_{\mathrm{abs}} = (\upsilon_p - \upsilon_n) - |\tilde{\upsilon}| \geq 0 \tag{5.34b}$$

$$l_{\mathrm{abs}} = (\upsilon_p - \upsilon_n) - \mathrm{sign}(\tilde{\upsilon})\, \tilde{\upsilon} \geq 0 \tag{5.34c}$$

$$\Phi_u = u = \begin{bmatrix} \mathrm{s}_\varphi & \mathrm{s}_\varphi \end{bmatrix} \begin{bmatrix} \upsilon_p \\ \upsilon_n \end{bmatrix} + \begin{bmatrix} \mathrm{c}_\varphi & \mathrm{c}_\varphi \end{bmatrix} \begin{bmatrix} u_p \\ u_n \end{bmatrix} + u_{\mathrm{tr}}. \tag{5.34d}$$

Since the angle φ is constant, the trigonometric functions represent constant values. Thus, a rotated space splitting procedure also yields affine constraints and mappings enabling the formulation of a convex OP. ■

The previous theorems required the subsets to be connected in a single point. Following theorem states that the SSC approach can be used to approximate ZCSs that share a line of points:

Theorem 5.3.3. *When a two-dimensional ZCSs is comprised of two convex subsets sharing a line of points, the SSC approach can only approximate the original set however with sufficiently small approximation error.*

Proof. Without loss of generality, this can be analysed considering Fig. 5.4c as example. For the depicted case, the transition line lies on the ordinate $v_{\mathrm{tr}} = 0$. Analysing the individual cases shows that using the original subsets within the SSC approach would yield a larger set than the original set:
Case 1: $v_{\mathrm{tr}} < v$

$$\left.\begin{array}{r} v_p = v \;\Rightarrow\; u_p \in \hat{\mathcal{S}}_p \\ v_n = v_{\mathrm{tr}} \;\Rightarrow\; u_n \in [0, \overline{u}_0] \end{array}\right\} \;\Rightarrow\; \Phi_u = (u_n + u_p) \in \hat{\mathcal{S}}_{\Phi,p} \supset \hat{\mathcal{S}}_p \quad \not\Rightarrow \quad u \in \mathcal{S} = \mathcal{S}_p. \tag{5.35a}$$

Case 2: $v < v_{\mathrm{tr}}$

$$\left.\begin{array}{r} v_p = v_{\mathrm{tr}} \;\Rightarrow\; u_p \in [0, \overline{u}_0] \\ v_n = v \;\Rightarrow\; u_n \in \hat{\mathcal{S}}_n \end{array}\right\} \;\Rightarrow\; \Phi_u = (u_n + u_p) \in \hat{\mathcal{S}}_{\Phi,n} \supset \hat{\mathcal{S}}_n \quad \not\Rightarrow \quad u \in \mathcal{S} = \mathcal{S}_n. \tag{5.35b}$$

Case 3: $v = v_{\mathrm{tr}}$

$$\left.\begin{array}{r} v_p = v_{\mathrm{tr}} \;\Rightarrow\; u_p \in [0, \overline{u}_0] \\ v_n = v_{\mathrm{tr}} \;\Rightarrow\; u_n \in [0, \overline{u}_0] \end{array}\right\} \;\Rightarrow\; \Phi_u = u_n + u_p \in [0, 2\,\overline{u}_0] \quad \not\Rightarrow \quad u \in [0, \overline{u}_0]. \tag{5.35c}$$

Thus, using the original subsets can render non-admissible points of the original problem admissible. When the input variable is replaced by the auxiliary inputs according to (5.29), both sets must be adjusted at the transition line $v = v_{\mathrm{tr}}$. The simplest set adjustment is enforcing $u_n = 0 = u_p$ at the transition axis, as illustrated in Fig. 5.4c by the dotted lines. Then, the subsets are connected in a single point enabling the application of Theorem 5.3.2. However, other set manipulations are possible and can be numerically better. Generally, such set approximations introduce conservativeness for the sake of feasible trajectories. However, the introduced error is negligible when the removed area is sufficiently small, which can be implemented by choosing steep linear constraints at the transition between the subsets. ■

Remark 2. *In this thesis, only two-dimensional ZCSs that can be split into two convex subsets have been considered. However, the SSC approach can be applied to certain two-dimensional ZCSs with more than two subsets. For instance, consider three convex subsets positioned horizontally to each other whereas each subset is connected to the next subset in a single point on the abscissa. This can be verified using the procedures described previously in this section.*

This section illustrated the convexification procedure for ZCSs in regards to inputs. This results in the depicted affine input transformation Φ_u. Implementing the approach for a ZCS with state decision variables yields an affine transformation Φ_x. This procedure can be applied to multiple inputs and states. The corresponding vector functions for the affine transformations are given by

$$\boldsymbol{\Phi}_u = \begin{bmatrix} \Phi_{u,1} \\ \vdots \\ \Phi_{u,n_u} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Phi}_x = \begin{bmatrix} \Phi_{x,1} \\ \vdots \\ \Phi_{x,n_x} \end{bmatrix} \tag{5.36}$$

with $\Phi_{u,i}$ and $\Phi_{x,j}$ representing the transformation for the individual input and state, respectively. If no transformation is necessary, a direct mapping is implemented according to $\Phi_{u,i} = u_i$ and $\Phi_{x,i} = x_i$. The vectors of transformations (5.36) are used to generate $\mathbf{\Phi}$ in (5.15), which is used for the convex depiction of the original constraints in (5.13b)-(5.13d).

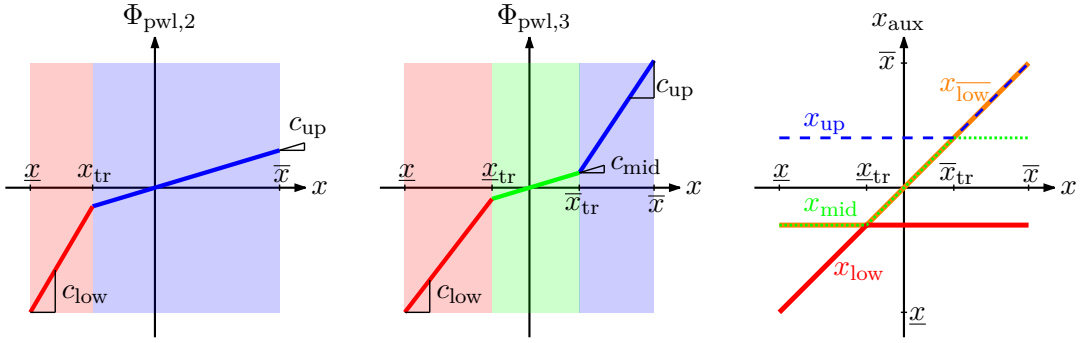
5.3.2 Convexification of Nonlinear Equality Constraints

Some nonlinearities in physical systems are characterised by a univariate function. Examples are nonlinear springs, saturation, dead-zones or the tire force shape curve of vehicles. Since nonlinear system equations yield nonlinear thus nonconvex equality constraints, this section presents how such nonlinear univariate terms can be convexified. The nonlinear univariate curve is depicted using piecewise affine parts. Hence, if the nonlinearity is not a piecewise linear function, SSC introduces approximation errors. The subsequent paragraphs illustrate the application of the SSC algorithm for convexifying piecewise linear, univariate nonlinearities.

Theorem 5.3.4. *The splitting constraints in* (5.7) *can be used to convexify equality constraints by transforming a univariate nonlinearity with two piecewise linear segments into an expression that is affine in multiple decision variables.*

Proof. Proof is provided using the piecewise linear curve depicted in Fig. 5.5a as example, which represents a typical function for piecewise linear springs. Assuming the position x is a state of the system, it represents a decision variable that will be split at the transition point $x = x_{\mathrm{tr}} < 0$. Thus, the auxiliary position variables $x_{\mathrm{low}} \in \mathcal{X}_{\mathrm{low}} \coloneqq [\underline{x}, x_{\mathrm{tr}}]$ and $x_{\mathrm{up}} \in \mathcal{X}_{\mathrm{up}} \coloneqq [x_{\mathrm{tr}}, \overline{x}]$ are introduced. By including analogous constraints to (5.7a) and (5.9a) as well as additional objective terms in form of (5.9b), following relations are ensured using Lemma 5.2.1:

$$x_{\mathrm{low}} = \begin{cases} x & \forall x \leq x_{\mathrm{tr}}, \\ x_{\mathrm{tr}} & \text{else} \end{cases}, \quad x_{\mathrm{up}} = \begin{cases} x & \forall x \geq x_{\mathrm{tr}}, \\ x_{\mathrm{tr}} & \text{else} \end{cases}. \tag{5.37}$$



(a) Nonlinearity with two linear segments. **(b)** Nonlinearity with three linear segments. **(c)** Splitting of state variable x for three segments.

Figure 5.5: Space splitting convexification of piecewise linear equalities.

Hence, the piecewise linear function can be represented by the expression

$$\Phi_{\text{pwl,2}} = c_{\text{up}}\, x_{\text{up}} + c_{\text{low}}\,(x_{\text{low}} - x_{\text{tr}}) = \begin{bmatrix} c_{\text{up}} & c_{\text{low}} \end{bmatrix} \underbrace{\begin{bmatrix} x_{\text{up}} \\ x_{\text{low}} \end{bmatrix}}_{=:\mathbf{w}_{\text{aux}}} + (-c_{\text{low}}\, x_{\text{tr}})\,, \tag{5.38}$$

which is affine in the decision variables $\mathbf{w}_{\text{aux}}$. The relations in (5.37) yield following two segments for the piecewise linear function (5.38):

$$\text{I: } x_{\text{tr}} \leq x \;\Rightarrow \begin{cases} x_{\text{up}} = x \\ x_{\text{low}} = x_{\text{tr}} \end{cases} \Rightarrow \Phi_{\text{pwl,2}} = c_{\text{up}}\, x. \tag{5.39a}$$

$$\text{II: } x < x_{\text{tr}} \Rightarrow \begin{cases} x_{\text{up}} = x_{\text{tr}} \\ x_{\text{low}} = x \end{cases} \Rightarrow \Phi_{\text{pwl,2}} = c_{\text{up}}\, x_{\text{tr}} + c_{\text{low}}\,(x - x_{\text{tr}}). \tag{5.39b}$$

Thus, (5.39a) and (5.39b) depict the upper and lower segment of the nonlinearity, respectively. ■

Since (5.38) is an affine function in the decision variables, it can be inserted into the differential equations without preventing convexity. The subsequent theorem proves that this approach can be extended to univariate nonlinearities with more than two piecewise linear segments:

Theorem 5.3.5. *A repeated application of the space bisection approach using the splitting constraints in* (5.7) *can be used to convexify equality constraints by transforming a univariate nonlinearity with more than two piecewise linear segments into an expression that is affine in multiple decision variables.*

Proof. The general procedure for this extension is illustrated using the piecewise linear curve depicted in Fig. 5.5b. The position x is split at the transition points $x = \underline{x}_{\text{tr}} < 0$ and $x = \overline{x}_{\text{tr}} > 0$. Thus, the auxiliary position variables $x_{\text{low}} \in \mathcal{X}_{\text{low}} \coloneqq [\underline{x}, \underline{x}_{\text{tr}}]$, $x_{\overline{\text{low}}} \in \mathcal{X}_{\overline{\text{low}}} \coloneqq [\underline{x}_{\text{tr}}, \overline{x}]$, $x_{\text{mid}} \in \mathcal{X}_{\text{mid}} \coloneqq [\underline{x}_{\text{tr}}, \overline{x}_{\text{tr}}]$ and $x_{\text{up}} \in \mathcal{X}_{\text{up}} \coloneqq [\overline{x}_{\text{tr}}, \overline{x}]$ are employed. The space is repeatedly bisected following Lemma 5.2.1 to gain

$$x_{\text{low}} = \begin{cases} x & \forall x \leq \underline{x}_{\text{tr}}, \\ \underline{x}_{\text{tr}} & \text{else} \end{cases}, \quad x_{\overline{\text{low}}} = \begin{cases} x & \forall x \geq \underline{x}_{\text{tr}}, \\ \underline{x}_{\text{tr}} & \text{else} \end{cases} \tag{5.40a}$$

$$x_{\text{mid}} = \begin{cases} x_{\overline{\text{low}}} & \forall x_{\overline{\text{low}}} \leq \overline{x}_{\text{tr}}, \\ \overline{x}_{\text{tr}} & \text{else} \end{cases}, \quad x_{\text{up}} = \begin{cases} x_{\overline{\text{low}}} & \forall x_{\overline{\text{low}}} \geq \overline{x}_{\text{tr}}, \\ \overline{x}_{\text{tr}} & \text{else} \end{cases} \tag{5.40b}$$

which is depicted in Fig. 5.5c. Thus, the position space is bisected into a region below and a region above $x = \underline{x}_{\text{tr}}$ whereas the upper region is again bisected into a region below and above $x = \overline{x}_{\text{tr}}$. The piecewise linear function can then be represented by the expression

$$\Phi_{\text{pwl,3}} = \begin{bmatrix} c_{\text{up}} & c_{\text{mid}} & c_{\text{low}} \end{bmatrix} \underbrace{\begin{bmatrix} x_{\text{up}} \\ x_{\text{mid}} \\ x_{\text{low}} \end{bmatrix}}_{=:\mathbf{w}_{\text{aux}}} - (c_{\text{up}}\, \overline{x}_{\text{tr}} + c_{\text{low}}\, \underline{x}_{\text{tr}}), \tag{5.41}$$

which is affine in the decision variables $\mathbf{w}_{\text{aux}}$. The relations in (5.40) yield following three segments for the piecewise linear function (5.41):

$$\text{I: } \overline{x}_{\text{tr}} < x \quad \Rightarrow \begin{cases} x_{\text{up}} = x \\ x_{\text{mid}} = \overline{x}_{\text{tr}} \\ x_{\text{low}} = \underline{x}_{\text{tr}} \end{cases} \Rightarrow \Phi_{\text{pwl},3} = c_{\text{mid}}\,\overline{x}_{\text{tr}} + c_{\text{up}}\,(x - \overline{x}_{\text{tr}}). \tag{5.42a}$$

$$\text{II: } \underline{x}_{\text{tr}} \leq x \leq \overline{x}_{\text{tr}} \Rightarrow \begin{cases} x_{\text{up}} = \overline{x}_{\text{tr}} \\ x_{\text{mid}} = x \\ x_{\text{low}} = \underline{x}_{\text{tr}} \end{cases} \Rightarrow \Phi_{\text{pwl},3} = c_{\text{mid}}\,x. \tag{5.42b}$$

$$\text{III: } x < \underline{x}_{\text{tr}} \quad \Rightarrow \begin{cases} x_{\text{up}} = \overline{x}_{\text{tr}} \\ x_{\text{mid}} = \underline{x}_{\text{tr}} \\ x_{\text{low}} = x \end{cases} \Rightarrow \Phi_{\text{pwl},3} = c_{\text{mid}}\,\underline{x}_{\text{tr}} + c_{\text{low}}\,(x - \underline{x}_{\text{tr}}). \tag{5.42c}$$

Hence, (5.42a), (5.42b) and (5.42c) depict the upper, mid and lower segment of the nonlinearity, respectively. ■

This repeated bisection procedure enables the generation of a characteristic curve with an arbitrary number of segments. However, an increasing number of segments also increases the number of auxiliary decision variables and additional constraints making the OP larger.
The approach presented in this section yields a scalar function Φ_{pwl} that is affine in the decision variables. The procedure can be applied to multiple univariate, nonlinear terms yielding further scalar mapping functions. The affine vector-valued function in (5.14) is given by

$$\boldsymbol{\Phi}_{\text{pwl}} = \begin{bmatrix} \Phi_{\text{pwl},1,1} + \Phi_{\text{pwl},1,2} + \dots \\ \vdots \\ \Phi_{\text{pwl},n_x,1} + \Phi_{\text{pwl},n_x,2} + \dots \end{bmatrix} \tag{5.43}$$

with $\Phi_{\text{pwl},i,j}$ representing the individual affine mapping functions. For states that are not affected by the space splitting approach, $\Phi_{\text{pwl},i,j} = 0$ holds. The transformation (5.43) is used to generate the convex collocation constraints (5.13b).

5.4 Remarks on Runtime

The application of optimisation methods within a real-time capable controller generally requires upper bounds on the necessary computation time. The SSC approach runs through a loop that is comprised of a projection step and an optimisation step resulting in the following theorem regarding computational complexity:

Theorem 5.4.1. *The computational complexity of the SSC algorithm is polynomial.*

Proof. SSC solves a sequence of LP problems or convex QP problems, which are known to be solvable in polynomial time [91, 107]. The projection step (5.10) of the SSC algorithm uses minimum and maximum functions, which possess linear time complexity. This concludes a polynomial time complexity of the SSC algorithm:

$$T_{\mathrm{SSC}}(n,q) = q \cdot \left(\mathcal{O}(n^{\alpha}) + c\,\mathcal{O}(n)\right) \quad \text{with} \quad \alpha > 1. \tag{5.44}$$

Therein n can be interpreted as the number of operations required by the algorithm. The parameters in (5.44) depend on the utilised solver as well as the number of decision variables, which is problem specific. Limiting the maximum number of superordinate iterations to $q \leq q_{\max}$ enables terminating the algorithm prematurely at suboptimal solutions, if necessary. ■

Theorem 5.4.1 provides valuable insight for the application in a real-time capable controller. Firstly, a worst-case bound on computation time can be experimentally identified for the specific OP and solver. Secondly, the computation time scales well with increasing problem size. Since the number of superordinate SSC iterations is rather low in practice, this results in a fast solution of OCPs. Finally, the influences on computation time are summarised below:

1. **System and discretisation.** The number of decision variables increases with rising number of inputs, states and collocation segments for both the original and the convexified OP. This increases computation time for NLP methods and the SSC algorithm. Due to the polynomial complexity, this increase in time will be generally less for SSC than for NLP.
2. **Nonlinearity.** The more complicated the nonlinearity, the more splitting segments are required to reduce approximation errors. However, the space splitting increases the problem size compared to the original problem, which influences the variable n in (5.44). This can put a practical bound on the number of space divisions via the SSC approach. Nevertheless, many physically motivated nonlinearities can be approximated sufficiently well using a moderate number of affine segments.
3. **Initial guess.** Being a local method, the solution computed by the SSC approach depends on the provided initial solution. Furthermore, this initial guess can influence the order of sign-corrections regarding the AVC-linearisations, which can impact the number of superordinate iterations and therefore the convergence rate. For MPC frameworks, the optimal solution of the preceding time step often represents a good initial guess [71] promising low computation times.
4. **Penalty parameter update.** Too small or too large penalty parameters can slow down convergence [148, p.511]. The straightforward approach of gradually increasing the penalty parameter is used in this thesis. However, more sophisticated routines have been proposed in [135, 140].

5.5 Comparison with Related Methods

In order to highlight the novelty of the proposed algorithm, this section compares the SSC approach with the most related existing methods.
From an algorithmic standpoint, linearising only the nonconvex part of constraints combined with a relaxation and an increasing objective term penalising constraint violations corresponds to penalty CCP [121]. CCP has the advantage over SQP and other linearisation techniques that it retains more problem information in each iterate: The information of the convex parts is kept and only the concave portions are linearised. Furthermore, CCP does not require to use line-search procedures or limit the progress in each iteration via trust-region methods [121]. Inspired by [130], the SSC approach avoids additional slack variables by considering the linearised constraints directly in the penalty objective. This avoids further increasing the size of the OP as in penalty CCP. Moreover, SSC employs an intermediate projection to enhance convergence. While CCP linearises the concave part of all nonconvexities, SSC transforms the problem beforehand requiring only the linearisation of AVCs. This enables providing a feedback about the correctness of the selected linearisation points, which are subsequently adjusted. Thus, the linearisation error can vanish completely after the correct signs have been chosen. The binary nature of the linearisation facilitates a rapid convergence in a small number of superordinate iterations. Especially the straightforward linearisation of ZCS-constraints yields very poor approximations that can contain a large set of infeasible points. Thus, the vanishing AVC-linearisation errors result in an advantage of SSC regarding accuracy, also over further linearisation techniques like the ones presented in [133, 134]. Although the SSC method only computes solutions to the piecewise linear approximation of the original problem, the approximation can be designed to sufficient accuracy enabling small and known approximation errors. Generally, a scalable trade-off between accuracy and size of the OP, thus computation time, can be chosen.
While LC possesses the advantage of avoiding approximation errors, it is only applicable to OPs with annulus-like, nonconvex sets resulting from excluding a convex subset from a convex set [165, p.340]. Opposed to that, SSC provides a method for the convexification of ZCS, which cannot be considered via LC.
Similar to the LnP approach presented in [134], the SSC method is an iterative linearisation algorithm using intermediate projection steps. The updating routine for SSC is of linear computational complexity with few and simple mathematical operations. The projection step of the LnP approach generally requires the solution of a convex programming problem, which is of polynomial complexity at best. Thus, the projection step used by the SSC approach is computationally less costly, which is beneficial for splitting methods [61]. Additionally, the binary nature of the AVC-linearisation provides the advantage that generally only few superordinate iterations are required, resulting in short computation times. Moreover, SSC is more robust regarding the initialisation since the LnP approach requires an additional lifting procedure to cope with arbitrary initialisations, which costs computation time [134]. Furthermore, the LnP method requires the right-hand side of the system equations to be convex and the inequality constraints to be concave. However, nonconvex collision avoidance constraints can be easily considered

via the LnP approach whereas the SSC procedure requires further measures.
The ADMM is a splitting procedure utilising the augmented Lagrangian method instead of the exact penalty function method, which is employed by the SSC algorithm. Due to the alternating approach, the ADMM requires the solution of two QCQP problems in each iteration. The SSC approach only requires the solution of a single convex QP problem in each iteration, which is beneficial for computation time.
As mentioned in Section 1.2, the notion of space decomposition is also used by GO methods, which also partly use auxiliary variables to reformulate the OP. SSC uses additional decision variables to approximate the original problem via piecewise linear segments. In contrast, the auxiliary variables are used by GO methods to decompose nonconvex functions into individual terms for which convex relaxations are available. An OP with piecewise linear segments can also be represented by a MIP problem [8, 68]. Moreover, special ordered sets of type one could be used to consider that only one of the piecewise linear segments is active at a time. This would reduce the search space and thus enhance convergence speed. Solving such a MIP via the BnB approach would provide the global optimum by solving multiple OPs with convex relaxations on increasingly smaller subdomains. These subproblems provide lower bounds on the objective function for the considered subdomain. By iteratively dividing the domain into smaller segments, the lower bound converges to the local solution on the subdomain. Comparing the solutions on the individual subdomains provides the global solution. However, the SSC approach solves a piecewise linear approximation of the original problem, which is iteratively adjusted, on the entire space domain of the decision variables. Opposed to that, the BnB method solves multiple OPs that represent convexly relaxed versions of the original problem on separate subdomains of the decision variables. BnB techniques determine upper bounds by evaluating the objective function at a feasible point. Since the problem is generally nonconvex, finding a feasible point can be difficult. This is often tackled by solving the nonconvex subproblem locally, which is computationally expensive [115, p.253]. Thus, the convergence rate of the SSC approach is most likely to be superior if sufficiently good initial guesses are provided. However, this depends on the nonlinearities of the OP, which influence the number of decision variables for SSC but also for BnB approaches that decompose factorable functions [204, p.125]. Moreover, SSC only guarantees to find local optima of an approximation of the OP and is only applicable to certain classes of OPs. GO techniques cover a wide range of applicability and provide the global optimum. A direct comparison of GO methods and the SSC algorithm is subject of future research.

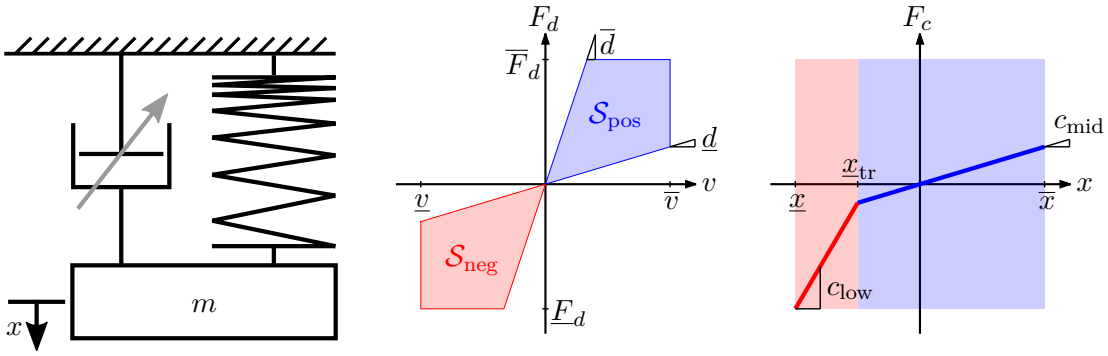
6 Applications for Space Splitting Convexification

The main drawback of optimal control methods is the rather long computation time, which often prevents real-time implementations. Thus, a fast solution procedure for nonconvex OCPs, which iteratively solves convex QP problems, has been presented in Chapter 5. In this chapter, which contains parts of our publication [186], the accuracy and computation time of the presented algorithm are analysed by studying several applications.
In Section 6.1, optimal control inputs are computed for a hanging single-mass oscillator (SMO) with two nonconvexities. Firstly, a piecewise linear spring with two segments results in nonlinear thus nonconvex collocation constraints. Secondly, a semi-active damper yields nonconvex inequality constraints, which are input and state dependent. The solution generated via SSC is compared to the solution computed via the unscaled NLP-approach presented in Chapter 3. The performance of the SSC algorithm is studied for multiple OPs varying in number of decision variables and initial value condition. In Section 6.1.3, the application is extended to a nonlinear spring with three linear segments. This illustrates the possibility to depict an arbitrary number of piecewise linear segments using the successive bisection approach presented in Section 5.3.2. Furthermore, the influence of the initial guess on the solution is analysed by varying the initial solutions fed to the SSC algorithm. An artificial OP in Section 6.1.4 briefly illustrates the possibility of splitting a ZCS across a rotated line, which has been introduced in Section 5.3.1. The application of SSC to a system with multiple univariate nonlinear relations is presented in Section 6.2: Using a simplified single-track vehicle model, optimal control inputs are computed for a drag race application. Therein the nonlinear tire force shape curve and air resistance force are approximated using piecewise linear segments. Furthermore, space splitting is applied to approximate the nonconvex set defining the admissible torques.

6.1 Minimisation of Steady-State Position Deviation for Hanging Single-Mass Oscillator

In this section, the proposed SSC algorithm is evaluated using a hanging SMO example with piecewise linear spring and semi-active damping as illustrated in Fig. 6.1. Semi-active dampers can only generate a force in the opposite moving direction and cannot impose a force at steady-state complying with the passivity constraint [179, p.16]. Hence, the admissible set for semi-active dampers always contains a subset in the first quadrant

and a subset in the third quadrant which are connected at a single point, namely the origin. A performance-oriented OP is formulated and solved using NLP and SSC, respectively. The results of both algorithms are compared regarding accuracy and computation time. Both problems are posed using the modelling language *JuMP* [52] for mathematical OPs. For a fair comparison, the necessary derivatives are computed beforehand via automatic differentiation. Furthermore, the tolerances for constraint feasibility and ob-



(a) Hanging SMO model. **(b)** ZCS for semi-active damper force. **(c)** Piecewise linear spring force characteristic.

Figure 6.1: Characteristics of hanging SMO with semi-active damper and nonlinear spring.

Table 6.1: Parameters for single-mass oscillator application.

	Symbol	Value	Unit	Description
system	g	9.81	$\mathrm{m/s^2}$	gravitational acceleration
	m	5.00	kg	mass
	c_{up}	10.00	N/m	overload spring stiffness for rebound
	c_{mid}	3.00	N/m	main spring stiffness
	c_{low}	5.00	N/m	overload spring stiffness for compression
	$\overline{x}_{\text{tr}}$	5.00	m	transition point for rebound overload
	$\underline{x}_{\text{tr}}$	−5.00	m	transition point for compression overload
optimisation	$x_{\text{ss},2}$	16.35	m	steady-state position for 2-segment spring
	$x_{\text{ss},3}$	8.405	m	steady-state position for 3-segment spring
	$\overline{x}$	100.00	m	upper position bound
	$\underline{x}$	−100.00	m	lower position bound
	$\overline{d}$	20.00	Ns/m	upper bound on damper coefficient
	$\underline{d}$	0.50	Ns/m	lower bound on damper coefficient
	$\overline{v}$	100.00	m/s	upper speed bound
	$\underline{v}$	−100.00	m/s	lower speed bound
	$\overline{F}_d$	400.00	N	upper bound on damper force
	$\underline{F}_d$	−400.00	N	lower bound on damper force
	ε_g	10^{-6}	-	violation tolerance for SSC penalty term
	$\underline{\tau}/\overline{\tau}$	$1.00/10^4$	-	initial/final value for penalty parameter

jective improvement are set to 10^{-6} for both solvers. The NLP problem is solved using the optimiser *IPOPT*, which employs a sophisticated IPM [222]. The novel SSC method transforms the OP into convex QP subproblems, which are solved iteratively. In order to capitalise on this problem structure, the elaborate optimiser *Gurobi* [74] is selected. The utilised parameters are listed with description in Table 6.1.

6.1.1 Optimisation Problems

The equations of motion for the state vector $\mathbf{x} = \begin{bmatrix} x & v \end{bmatrix}^T$ result in

$$\dot{\mathbf{x}} = \mathbf{f} = \begin{bmatrix} v \\ \frac{1}{m}\left(m\,g - F_d - F_c\right) \end{bmatrix} \tag{6.1}$$

with x, v, F_d and F_c denoting the deflection position, deflection speed, damper force and spring force, respectively. The expression of these forces differs for the NLP method and the SSC approach. Starting from a specified initial state $\mathbf{x}_{\mathrm{iv}} = \begin{bmatrix} x_{\mathrm{iv}} & v_{\mathrm{iv}} \end{bmatrix}^T$ with damper force $F_{d,\mathrm{iv}} = \underline{d}\, v_{\mathrm{iv}}$, the goal is to minimise the deviation from the steady-state position x_{ss} while considering the limitations of the system. Applying Simpson quadrature (3.14), the main objective penalises the steady-state position deviation $e_{\mathrm{ss},j} \coloneqq x_j - x_{\mathrm{ss}}$ according to

$$J_{\mathrm{ss}} \coloneqq \sum_{i=0}^{n_{\mathrm{seg}}-1} \frac{\Delta_i}{6}\left(e_{\mathrm{ss},i}^2 + 4\,e_{\mathrm{ss},i+\frac{1}{2}}^2 + e_{\mathrm{ss},i+1}^2\right). \tag{6.2}$$

Considering the values listed in Table 6.1, the steady-state position can be computed using a root-finding approach like Newton's method: At steady-state position, the gravitational force must equal the spring force. For the considered nonlinear springs with two piecewise linear segments the value $x_{\mathrm{ss}} = x_{\mathrm{ss},2}$ is chosen and $x_{\mathrm{ss}} = x_{\mathrm{ss},3}$ if the spring with three segments is used. A uniform discretisation of the time grid $t \in [t_0, t_f] = [0, 10]$ is employed yielding a constant segment width $\Delta_i = \Delta = \frac{t_f - t_0}{n_{\mathrm{seg}}}$. Both, the NLP approach and the SSC method, are initialised with zero vectors $\mathbf{U}_0^\star = \mathbf{0}$ and $\mathbf{X}_0^\star = \mathbf{0}$. From

$$f_{\mathrm{ss},i}(x) \coloneqq \frac{\Delta_i}{6}(x_i - x_{\mathrm{ss}})^2 \quad \Rightarrow \quad f'_{\mathrm{ss},i}(x) = 2\,\frac{\Delta_i}{6}(x_i - x_{\mathrm{ss}}) \quad \Rightarrow \quad f''_{\mathrm{ss},i}(x) = 2\,\frac{\Delta_i}{6} > 0 \tag{6.3}$$

follows that the Hessian of (6.2) is a diagonal matrix with only positive and zero diagonal entries and therefore positive semi-definite. Hence, the main objective (6.2) is a convex function.

Nonlinear, Nonconvex Optimisation Problem For the solution of the OP via NLP, the decision variables are chosen as

$$\mathbf{u} = \begin{bmatrix} u_0 & u_{0.5} & \dots & u_{n_{\mathrm{seg}}} \end{bmatrix} \in \mathbb{R}^{1 \times n_{\mathrm{coll}}} \text{ with } u_k = d_k \tag{6.4a}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_{0.5} & \dots & \mathbf{x}_{n_{\mathrm{seg}}} \end{bmatrix} \in \mathbb{R}^{2 \times n_{\mathrm{coll}}} \text{ with } \mathbf{x}_k \coloneqq \begin{bmatrix} x_k \\ v_k \end{bmatrix} \tag{6.4b}$$

whereas the input represents the variable damping coefficient d_k of the semi-active actuator. With $i \in \mathcal{I}_{\text{coll}}$ and $k \in \mathcal{K}$, the original OP is given by

$$\min_{\mathbf{u},\mathbf{X}} \quad J_{\text{ss}} \tag{6.5a}$$

$$\text{s.t.} \quad \mathbf{c}_{\text{coll},i} \overset{(5.4)}{=} \mathbf{0} \;\text{ with }\; \mathbf{f}_k = \begin{bmatrix} v_k \\ \frac{1}{m}\left(m\,g - F_{d,k}^{\text{NLP}} - F_{c,2,k}^{\text{NLP}}\right) \end{bmatrix},\; F_{d,k}^{\text{NLP}} = u_k\, v_k, \tag{6.5b}$$

$$F_{c,2,k}^{\text{NLP}} = \min\nolimits_{\varepsilon}\left(c_{\text{mid}} x_k,\; c_{\text{low}} x_k - \underline{x}_{\text{tr}}(c_{\text{low}} - c_{\text{mid}})\right)$$

$$u_k \in \mathcal{D} \coloneqq \left[\underline{d}, \overline{d}\right], \quad x_k \in \mathcal{X} \coloneqq [\underline{x}, \overline{x}], \quad v_k \in \mathcal{V} \coloneqq [\underline{v}, \overline{v}], \tag{6.5c}$$

$$\underline{F}_d \leq u_k\, v_k \leq \overline{F}_d, \tag{6.5d}$$

$$u_0 = \underline{d}, \quad \mathbf{x}_0 = \mathbf{x}_{\text{iv}}. \tag{6.5e}$$

The nonconvex NLP problem (6.5) is comprised of following parts: objective function (6.5a), collocation constraints (6.5b), bounds for the decision variables (6.5c), damper force bounds (6.5d) and initial value condition (6.5e). The piecewise linear spring force is approximated in (6.5b) via the smoothed minimum-function (3.3b) using $\varepsilon = 10^{-16}$ to guarantee that the problem is twice continuously differentiable. Problem (6.5) is nonconvex due to the right-hand side of the differential equation system in the collocation constraints (6.5b) and due to the damper force bounds (6.5d).

Convex QP Problem for Iterative Solution Following the SSC approach, auxiliary optimisation variables are introduced, which results in the subsequent decision variables:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_{0.5} & \dots & \mathbf{u}_{n_{\text{seg}}} \end{bmatrix} \in \mathbb{R}^{2\times n_{\text{coll}}} \text{ with } \mathbf{u}_k \coloneqq \begin{bmatrix} u_{\text{pos},k} \\ u_{\text{neg},k} \end{bmatrix} = \begin{bmatrix} F_{d,\text{pos},k} \\ F_{d,\text{neg},k} \end{bmatrix} \tag{6.6a}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_{0.5} & \dots & \mathbf{x}_{n_{\text{seg}}} \end{bmatrix} \in \mathbb{R}^{2\times n_{\text{coll}}} \text{ with } \mathbf{x}_k \coloneqq \begin{bmatrix} x_k \\ v_k \end{bmatrix} \tag{6.6b}$$

$$\mathbf{X}_{\text{aux}} = \begin{bmatrix} \mathbf{x}_{\text{aux},0} & \mathbf{x}_{\text{aux},0.5} & \dots & \mathbf{x}_{\text{aux},n_{\text{seg}}} \end{bmatrix} \in \mathbb{R}^{2\times n_{\text{coll}}} \text{ with } \mathbf{x}_{\text{aux},k} \coloneqq \begin{bmatrix} x_{\text{mid},k} \\ x_{\text{low},k} \end{bmatrix} \tag{6.6c}$$

$$\mathbf{V}_{\text{aux}} = \begin{bmatrix} \mathbf{v}_{\text{aux},0} & \mathbf{v}_{\text{aux},0.5} & \dots & \mathbf{v}_{\text{aux},n_{\text{seg}}} \end{bmatrix} \in \mathbb{R}^{2\times n_{\text{coll}}} \text{ with } \mathbf{v}_{\text{aux},k} \coloneqq \begin{bmatrix} v_{\text{pos},k} \\ v_{\text{neg},k} \end{bmatrix}. \tag{6.6d}$$

Opposed to the inputs (6.4a) of the NLP problem (6.5), the inputs in (6.6a) represent the positive and negative part of the damper force.
As illustrated in Algorithm 2, a projection routine is executed in each iteration. Based on the preceding solution marked by values $(\cdot)^\star$, this routine computes the projected initial solution $\breve{(\cdot)}$ for $k \in \mathcal{K}$ via

$$\breve{\mathbf{u}}_k = \begin{bmatrix} \breve{u}_{\text{pos},k} \\ \breve{u}_{\text{neg},k} \end{bmatrix} = \begin{bmatrix} \max(u_{\text{neg},k}^\star + u_{\text{pos},k}^\star, 0) \\ \min(u_{\text{neg},k}^\star + u_{\text{pos},k}^\star, 0) \end{bmatrix}, \quad \breve{\mathbf{x}}_k = \mathbf{x}_k^\star \tag{6.7a}$$

$$\breve{\mathbf{x}}_{\text{aux},k} = \begin{bmatrix} \breve{x}_{\text{mid},k} \\ \breve{x}_{\text{low},k} \end{bmatrix} = \begin{bmatrix} \max(x_k^\star, \underline{x}_{\text{tr}}) \\ \min(x_k^\star, \underline{x}_{\text{tr}}) \end{bmatrix}, \quad \breve{\mathbf{v}}_{\text{aux},k} = \begin{bmatrix} \breve{v}_{\text{pos},k} \\ \breve{v}_{\text{neg},k} \end{bmatrix} = \begin{bmatrix} \max(v_k^\star, 0) \\ \min(v_k^\star, 0) \end{bmatrix} \tag{6.7b}$$

and the correct signs according to

$$\sigma_{x,k} = \operatorname{sign}(x_k^\star - \underline{x}_{\mathrm{tr}}), \quad \sigma_{v,k} = \operatorname{sign}(v_k^\star). \tag{6.7c}$$

For a concise notation in the OP, the decision variables are lumped together in $\mathbf{P} \coloneqq \{\mathbf{U}, \mathbf{X}, \mathbf{X}_{\mathrm{aux}}, \mathbf{V}_{\mathrm{aux}}\}$. With $i \in \mathcal{I}_{\mathrm{coll}}$ and $k \in \mathcal{K}$, the convex QP problem, which is iteratively solved for updated values of τ, $\sigma_{x,k}$, $\sigma_{v,k}$ and corresponding initial guesses, is given by

$$\min_{\mathbf{P}} \quad J_{\mathrm{ss}} + \tau \sum_{k\in\mathcal{K}} l_{\mathrm{abs},x,k} + l_{\mathrm{abs},v,k} \quad \text{with} \tag{6.8a}$$

$$\begin{aligned} l_{\mathrm{abs},x,k} &\coloneqq (x_{\mathrm{mid},k} - x_{\mathrm{low},k}) - \sigma_{x,k}\,(x_k - \underline{x}_{\mathrm{tr}}) \\ l_{\mathrm{abs},v,k} &\coloneqq (v_{\mathrm{pos},k} - v_{\mathrm{neg},k}) - \sigma_{v,k}\,v_k \end{aligned} \tag{6.8b}$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{c}_{\mathrm{coll},i} \overset{(5.4)}{=} \mathbf{0} \;\text{ with }\; \mathbf{f}_k = \begin{bmatrix} v_k \\ \frac{1}{m}\left(m\,g - F_{d,k}^{\mathrm{SSC}} - F_{c,2,k}^{\mathrm{SSC}}\right) \end{bmatrix}, \\ & F_{d,k}^{\mathrm{SSC}} = u_{\mathrm{neg},k} + u_{\mathrm{pos},k},\; F_{c,2,k}^{\mathrm{SSC}} = c_{\mathrm{mid}}\,x_{\mathrm{mid},k} + c_{\mathrm{low}}\,(x_{\mathrm{low},k} - \underline{x}_{\mathrm{tr}}) \end{aligned} \tag{6.8c}$$

$$\begin{aligned} & x_k \in \mathcal{X} \coloneqq [\underline{x}, \overline{x}]\,,\; v_k \in \mathcal{V} \coloneqq [\underline{v}, \overline{v}]\,, \\ & u_{\mathrm{pos},k} \in \mathcal{U}_{\mathrm{pos}} \coloneqq \left[0, \overline{F}_d\right],\; x_{\mathrm{mid},k} \in \mathcal{X}_{\mathrm{mid}} \coloneqq [\underline{x}_{\mathrm{tr}}, \overline{x}]\;\; v_{\mathrm{pos},k} \in \mathcal{V}_{\mathrm{pos}} \coloneqq [0, \overline{v}]\;, \\ & u_{\mathrm{neg},k} \in \mathcal{U}_{\mathrm{neg}} \coloneqq [\underline{F}_d, 0],\; x_{\mathrm{low},k} \in \mathcal{X}_{\mathrm{low}} \coloneqq [\underline{x}, \underline{x}_{\mathrm{tr}}]\;,\; v_{\mathrm{neg},k} \in \mathcal{V}_{\mathrm{neg}} \coloneqq [\underline{v}, 0] \end{aligned} \tag{6.8d}$$

$$\hat{\mathbf{h}}_{\mathrm{zcs},p,k} = \begin{bmatrix} -\overline{d} & 1 \\ \underline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\mathrm{pos},k} \\ u_{\mathrm{pos},k} \end{bmatrix} \leq \mathbf{0},\; \hat{\mathbf{h}}_{\mathrm{zcs},n,k} = \begin{bmatrix} -\underline{d} & 1 \\ \overline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\mathrm{neg},k} \\ u_{\mathrm{neg},k} \end{bmatrix} \leq \mathbf{0}, \tag{6.8e}$$

$$\begin{aligned} g_{\mathrm{aff},x,k} &= x_{\mathrm{mid},k} + x_{\mathrm{low},k} - (x_k + \underline{x}_{\mathrm{tr}}) = 0 \\ g_{\mathrm{aff},v,k} &= v_{\mathrm{pos},k} + v_{\mathrm{neg},k} - v_k = 0 \end{aligned}, \tag{6.8f}$$

$$\mathbf{h}_{\mathrm{abs},x,k} = \begin{bmatrix} l_{\mathrm{abs},x,k}\big|_{\sigma_{x,k}=+1} \\ l_{\mathrm{abs},x,k}\big|_{\sigma_{x,k}=-1} \end{bmatrix} \geq \mathbf{0}, \quad \mathbf{h}_{\mathrm{abs},v,k} = \begin{bmatrix} l_{\mathrm{abs},v,k}\big|_{\sigma_{v,k}=+1} \\ l_{\mathrm{abs},v,k}\big|_{\sigma_{v,k}=-1} \end{bmatrix} \geq \mathbf{0}, \tag{6.8g}$$

$$\mathbf{u}_0 = \begin{bmatrix} \max(\underline{d}\,v_{\mathrm{iv}}, 0) \\ \min(\underline{d}\,v_{\mathrm{iv}}, 0) \end{bmatrix}, \quad \mathbf{x}_0 = \mathbf{x}_{\mathrm{iv}}. \tag{6.8h}$$

The QP problem (6.8) is comprised of following parts: augmented objective function (6.8a), collocation constraints (6.8c), bounds for the decision variables (6.8d), ZCS-constraints (6.8e), the additional space splitting constraints (6.8f)-(6.8g) and initial value condition (6.8h). Due to the splitting approach, the right-hand side of the differential equation system in (6.8c) is an affine function in the decision variables. Thus, all constraints are affine in the decision variables and therefore convex. The additional objectives with (6.8b) are affine in the decision variables representing convex functions. Hence, (6.8) represents a convex QP problem.
Using (6.8b), the termination criterion in line 9 of Algorithm 2 is computed via

$$\bar{l}_{\mathrm{abs}} = \left\| \begin{bmatrix} l_{\mathrm{abs},x,0} \\ l_{\mathrm{abs},v,0} \end{bmatrix}, \ldots, \begin{bmatrix} l_{\mathrm{abs},x,n_{\mathrm{seg}}} \\ l_{\mathrm{abs},v,n_{\mathrm{seg}}} \end{bmatrix} \right\|_{\max}. \tag{6.9}$$

As illustrated in line 6 of Algorithm 2, the penalty weight τ is linearly increased with a maximum iteration number $q_{\max} = 6$ using the lower and upper bound $\underline{\tau}$ and $\overline{\tau}$,

respectively. Both OPs are now fully defined. The results generated by solving NLP problem (6.5) and by applying the SSC algorithm with QP problem (6.8) are compared in the following section.

6.1.2 Results

In order to inspect the performance of the proposed SSC algorithm, 100 OPs are analysed in this section using a desktop computer with Intel® Core™ i7-9850H CPU to determine the solutions. The OPs vary in size and the initial value condition. The number of collocation segments is gradually increased with $n_{\text{seg}} \in \{10, 50, 100, 250, 500\}$. For each problem size, the OPs (6.5) and (6.8) are solved using the 10 random, admissible initial values $\mathbf{x}_{\text{iv}}$ listed in Table 6.2. This procedure is also executed for a simplified QP problem (6.8) with a fully linear spring of stiffness c_{mid}. This simplifies the right-hand side of the differential equation system in (6.8c) and eliminates the need for the auxiliary variables $\mathbf{X}_{\text{aux}}$ in (6.6c), the corresponding additional objectives in (6.8b) as well as the corresponding constraints in (6.8f) and (6.8g). The optimisation results are displayed in Table 6.3.
Firstly, the OPs with initial value $\mathbf{x}_{\text{iv}} = \mathbf{x}_{\text{iv},1}$ and $n_{\text{seg}} = 250$ segments are studied in more detail. Fig. 6.2a depicts the piecewise linear characteristic curve of the spring force and Fig. 6.2b the admissible set for the damper force. The resulting trajectories of the states and damper force are illustrated in Fig. 6.2c. The mean absolute deviance between the damper force trajectory of the NLP solution and the SSC solution is defined as

$$\overline{e}_{F_d} \coloneqq \frac{1}{n_{\text{coll}}} \sum_{k \in \mathcal{K}} \left| F_{d,k}^{\text{NLP}} - F_{d,k}^{\text{SSC}} \right| \tag{6.10}$$

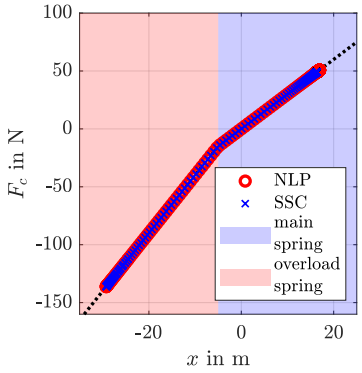
for the n_{coll} collocation points. With $\overline{e}_{F_d} = 1.85\text{N}$, the control trajectories differ only marginally considering the damper force domain $F_d \in \mathcal{F} \coloneqq \left[\underline{F}_d, \overline{F}_d\right] = [-400\text{N}, 400\text{N}]$. The mean absolute deviance between the position and the steady-state position

$$\overline{e}_{\text{ss}} \coloneqq \frac{1}{n_{\text{coll}}} \sum_{k \in \mathcal{K}} |x_k - x_{\text{ss},k}| \tag{6.11}$$

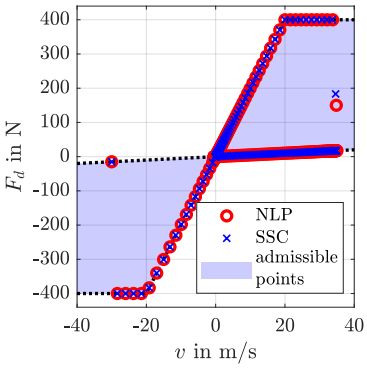
is $\overline{e}_{\text{ss}}^{\text{SSC}} = 7.12\text{m}$ and $\overline{e}_{\text{ss}}^{\text{NLP}} = 7.18\text{m}$ for the SSC solution and NLP solution, respectively. However, the value for the main cost function (6.2) is $J_{\text{ss}}^{\text{SSC}} = 2550.73\text{m}^2\text{s}$ and $J_{\text{ss}}^{\text{NLP}} = 2549.86\text{m}^2\text{s}$ for the SSC solution and NLP solution, respectively. Thus, the SSC objective value is higher but the mean deviance in the steady-state position error is lower. The reason for this is that the cost function is quadratic in the errors, which

Table 6.2: Initial values of robustness analysis.

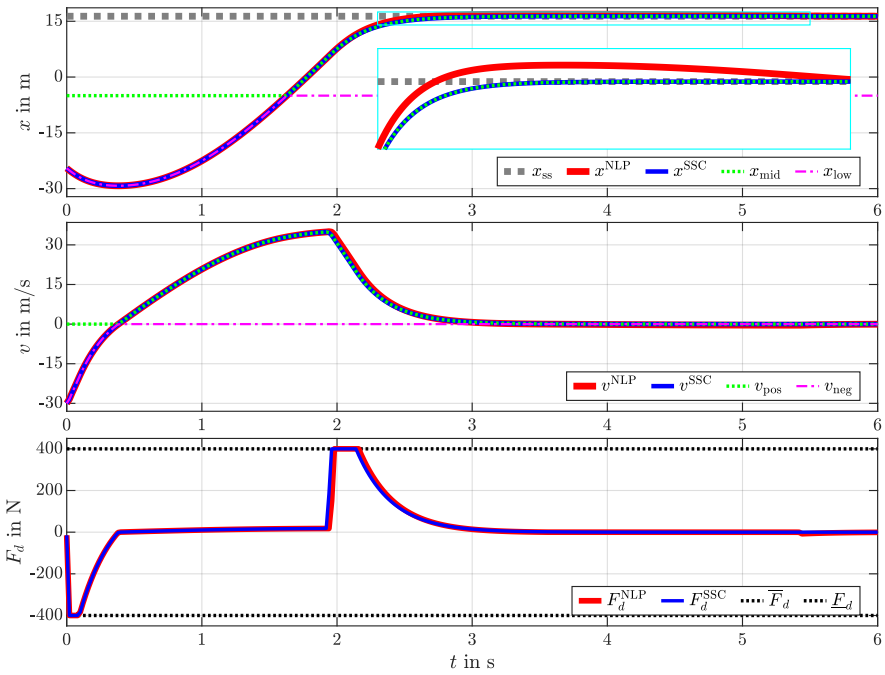
	$\mathbf{x}_{\text{iv},1}$	$\mathbf{x}_{\text{iv},2}$	$\mathbf{x}_{\text{iv},3}$	$\mathbf{x}_{\text{iv},4}$	$\mathbf{x}_{\text{iv},5}$	$\mathbf{x}_{\text{iv},6}$	$\mathbf{x}_{\text{iv},7}$	$\mathbf{x}_{\text{iv},8}$	$\mathbf{x}_{\text{iv},9}$	$\mathbf{x}_{\text{iv},10}$
x	-24.5	-15.1	-51.4	27.9	59.4	12.3	56.1	15.4	-73.7	-56.9
v	-30.0	-44.9	-15.4	39.3	18.7	64.3	-15.5	-32.7	16.8	73.1



(a) Piecewise linear spring force.



(b) ZCS for damper force.



(c) State and damper force trajectories.

Figure 6.2: Solution of OPs with initial value $\mathbf{x}_{\mathrm{iv},1}$ and $n_{\mathrm{seg}} = 250$ collocation segments.

penalises large errors unevenly more than small errors. As illustrated in the top subplot of Fig. 6.2c, the position of the NLP solution reaches the steady-state position faster however with an overshoot that slowly reduces. Thus, the larger errors at the beginning are reduced more quickly resulting in a lower cost function value. However, the mean position error, which is a better metric for the actual goal of minimising the deviance from the steady-state position, is worse than for the SSC solution. Although an objective penalising the absolute value of the errors would be more adequate, it is not used, since it would prevent continuous differentiability of the objective function. Nevertheless, both

algorithms reach the steady-state position x_{ss} in about 3 seconds resulting in similar state trajectories. The trajectories of the auxiliary variables verify correct switching at $x = \underline{x}_{\mathrm{tr}} = -5\mathrm{m}$ and $v = v_{\mathrm{tr}} = 0\frac{\mathrm{m}}{\mathrm{s}}$ for the position and velocity, respectively. Furthermore, Fig. 6.2a and Fig. 6.2b show that the spring forces lie on the characteristic line and the computed damper forces lie within the admissible set proving the compliance of the prescribed constraints. The bang-bang-like control strategy confirms correctness since it is expected due to the main objective (6.2) which demands to reduce the steady-state position deviation as fast as possible. With $q = 4$ superordinate iterations, the SSC method required a total optimisation time of $t_{\Sigma}^{\mathrm{SSC}} = 0.329\mathrm{s}$, which is 47.71% of the computation time of $t_{\Sigma}^{\mathrm{NLP}} = 0.690\mathrm{s}$ required by the NLP solver. The cumulative optimisation time $t_{\mathrm{opt},\Sigma}$, which represents the time spent in the QP solver in line 8 of the SSC algorithm for all passed loops, is $t_{\mathrm{opt},\Sigma}^{\mathrm{SSC}} = 0.262\mathrm{s}$ representing 37.97% of the NLP time. Since the SSC algorithm converged in $q < q_{\max} = 6$ iterations, the largest relaxation value of the AVCs is smaller than ε_g ensuring conformity with the original problem.
Subsequently, the remaining results of Table 6.3 are analysed to identify tendencies. As stated in Section 5.2.1, the additional objective terms (6.8b) represent a relaxation of the constraints comparable with slack variables. Besides serving as feedback for determining convergence, this relaxation provides robustness regarding the initialisation. This is reflected in the results by a successful convergence of the SSC algorithm in 100% of the test cases, even though a rather poor initial guess was used. The accuracy of the computed solutions is evaluated using the mean values of the main cost function (6.2) and of the averaged errors (6.10) and (6.11) defined as

$$\overline{J}_{\mathrm{ss}} \coloneqq \sum_{i=1}^{10} J_{\mathrm{ss},i}, \quad \overline{\overline{e}_{F_d}} \coloneqq \sum_{i=1}^{10} \overline{e}_{F_d,i} \quad \text{and} \quad \overline{\overline{e}_{\mathrm{ss}}} \coloneqq \sum_{i=1}^{10} \overline{e}_{\mathrm{ss},i}. \tag{6.12}$$

These values represent the respective mean value over all 10 OPs, due to 10 different initial values, for one specified number of collocation segments n_{seg}. With $\overline{\overline{e}_{F_d}} \leq 3.51\mathrm{N}$ being small compared to the damper force domain $\mathcal{F}$, the control trajectories of the SSC approach are close to the trajectories computed by the NLP solver. The small deviation of the mean values in (6.12) between the SSC solution and the NLP solution indicates that both algorithms converged to a similar solution. Considering the mean value $\overline{J}_{\mathrm{ss}}$ of the main objective function, the NLP solver yields lower objective values than the SSC algorithm. As previously discussed, the mean deviance $\overline{\overline{e}_{\mathrm{ss}}}$ from the steady-state position is smaller when using the SSC algorithm.
The SSC approach greatly reduces the overall computation time $t_{\Sigma}^{\mathrm{SSC}}$ and requires up to only 18.20% of the computation time needed by the NLP solver. The time advantage of the SSC algorithm over the NLP optimiser will be even bigger for asymmetric zonally convex damper sets. Such asymmetric sets require additional discontinuities in the NLP problem but can be easily implemented within the SSC procedure by using differing parameters for the individual convex subsets. As mentioned in Section 5.4, various aspects influence the runtime of the SSC algorithm, which is reflected in the results: With rising number of collocation segments, the mean solution time increases and the computation

Table 6.3: Optimisation results of robustness analysis.

	spring		**method**		**results**					
n_{seg}[-]	LIN	NL	NLP	SSC	$\overline{J}_{\text{ss}}$[m^{2}s]	$\overline{\overline{e}_{\text{ss}}}$[m]	$\overline{\overline{e}_{F_d}}$[N]	$\overline{q}$[-]	$t_{\text{opt},\Sigma}$[s]	t_{Σ}[s]
10	✓		✓		815.26	7.64	-	1.00	0.074	0.074
10	✓			✓	818.92	7.50	3.51	2.00	0.005 (6.45%)	0.021 (27.77%)
10		✓	✓		597.24	7.11	-	1.00	0.075	0.075
10		✓		✓	601.84	6.96	2.72	2.40	0.008 (10.90%)	0.025 (33.67%)
50	✓		✓		2135.09	6.67	-	1.00	0.125	0.125
50	✓			✓	2139.30	6.55	2.04	2.00	0.016 (12.87%)	0.036 (28.51%)
50		✓	✓		1842.48	6.16	-	1.00	0.127	0.127
50		✓		✓	1849.97	6.03	2.28	2.60	0.036 (28.46%)	0.059 (46.34%)
100	✓		✓		2333.84	6.52	-	1.00	0.229	0.229
100	✓			✓	2338.26	6.41	1.94	2.00	0.026 (11.18%)	0.050 (21.77%)
100		✓	✓		2043.91	6.02	-	1.00	0.252	0.252
100		✓		✓	2051.33	5.89	2.20	2.70	0.064 (25.44%)	0.095 (37.53%)
250	✓		✓		2454.05	6.43	-	1.00	0.587	0.587
250	✓			✓	2458.40	6.32	1.94	2.00	0.068 (11.64%)	0.105 (18.20%)
250		✓	✓		2167.57	5.93	-	1.00	0.695	0.695
250		✓		✓	2174.81	5.80	2.22	2.70	0.183 (26.30%)	0.235 (33.83%)
500	✓		✓		2494.96	6.40	-	1.00	1.080	1.080
500	✓			✓	2499.36	6.29	1.97	2.10	0.173 (16.03%)	0.235 (21.72%)
500		✓	✓		2209.03	5.91	-	1.00	1.373	1.373
500		✓		✓	2216.23	5.77	2.22	2.80	0.460 (33.50%)	0.555 (40.39%)

n_{seg}: # of collocation segments. LIN/NL: linear/nonlinear spring. NLP: *IPOPT* solving NLP problem (6.5). SSC: SSC Algorithm 2 with QP problem (6.8). $\overline{J}_{\text{ss}}/\overline{\overline{e}_{\text{ss}}}/\overline{\overline{e}_{F_d}}$: main objective and error metrics (6.12)(average for 10 initial value conditions). $\overline{q}$: average # of superordinate iterations. $t_{\text{opt},\Sigma}$: cumulative time spent in NLP/QP solver. t_{Σ}: total computation time.

time of SSC increases less than the NLP solution time. Another mentioned effect on runtime is the quality of the initial guess. The provided guess with zero vectors is of varying quality for the individual initial value conditions. A better initial guess in terms of the correct signs of the state trajectories reduces the number of superordinate iterations and thus the overall computation time. Hence, the sensitivity in regards to the initial guess is elaborated in Section 6.1.3.

The pure optimisation time $t_{\text{opt},\Sigma}^{\text{SSC}}$ reduces up to 6.45% of the corresponding NLP time and is generally significantly shorter than its overall computation time t_{Σ}^{SSC}. Since the projection step of the SSC algorithm employs only simple computations using minimum and maximum functions with linear time complexity, the time spent outside of the QP solver, which solves convex QP-problems in polynomial time, seems quite long. Unfortunately, substantial time losses occur at building the updated optimisation model via

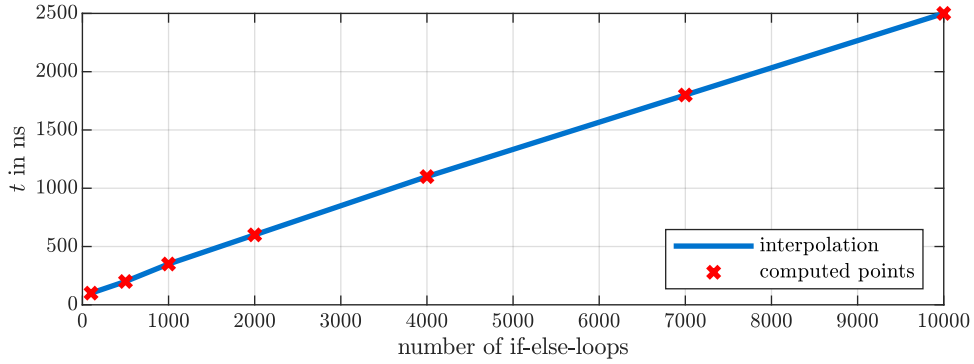


Figure 6.3: Computation time of if-else-operations in C-code.

JuMP and the *Gurobi*-wrapper. For future implementations, it is advisable to implement the SSC algorithm via C-code to circumvent this problem and minimise the overall computation time. This would enable a direct access to the solver and thus eliminate the communication time between *Julia* and the *Gurobi*-solver. An estimation for the computation time using C-code is illustrated in Fig. 6.3: The figure illustrates the required time to execute a number of if-else-loops that assign values to an array by determining if a randomly generated number exceeds a certain threshold value. The minimum, maximum and sign functions for the projection routine (6.7) can be implemented via such if-else-loops. This would require a loop for each $\breve{\mathbf{u}}_k$, $\breve{\mathbf{x}}_{\mathrm{aux},k}$, $\breve{\mathbf{v}}_{\mathrm{aux},k}$, $\sigma_{x,k}$ and $\sigma_{v,k}$ resulting in $n_{\mathrm{loop}} = 5\,n_{\mathrm{coll}}$ if-else-loops. For $n_{\mathrm{seg}} = 500$ with $n_{\mathrm{coll}} = 2\,n_{\mathrm{seg}} + 1$, this would require 5005 if-else-loops, which equates to about 1.335 microseconds and is thus significantly smaller than the discrepancy $t_\Sigma - t_{\mathrm{opt},\Sigma} = 95000$ microseconds for the nonlinear spring example. Although further time must be considered for the adaptation of the optimisation problem, this additional time will be small if the adjustment of the matrices and vectors is implemented via call-by-reference.

6.1.3 Influence of Initial Guess on Results

In this section, the OPs (6.5) and (6.8) are modified to consider the nonlinear spring characteristic with three piecewise linear segments depicted in Fig. 5.5b. The NLP problem (6.5) is adjusted by replacing the spring force $F_{c,2,k}^{\mathrm{NLP}}$ in the differential equation system considered in constraint (6.5b) by

$$F_{c,3,k}^{\mathrm{NLP}} = \max_\varepsilon \left[\min_\varepsilon \left(c_{\mathrm{mid}} x_k, c_{\mathrm{low}} x_k - \underline{x}_{\mathrm{tr}}(c_{\mathrm{low}} - c_{\mathrm{mid}})\right),\ c_{\mathrm{up}} x_k + \overline{x}_{\mathrm{tr}}(c_{\mathrm{mid}} - c_{\mathrm{up}})\right] \quad (6.13)$$

using the smooth maximum function (3.3a). The adaptation of the QP problem (6.8) requires slightly more changes. The auxiliary decision variables for the position (6.6c) are augmented resulting in

$$\mathbf{x}_{\mathrm{aux},k} \coloneqq \begin{bmatrix} x_{\overline{\mathrm{low}},k} & x_{\mathrm{low},k} & x_{\mathrm{up},k} & x_{\mathrm{mid},k} \end{bmatrix}^T. \quad (6.14)$$

Thus, the position space is bisected into a region below and a region above $x = \underline{x}_{\mathrm{tr}}$ whereas the upper region is again bisected into a region below and above $x = \overline{x}_{\mathrm{tr}}$. Compared to (6.7), the update procedure remains unchanged for the decision variables of the inputs, states and auxiliary velocities:

$$\begin{bmatrix} \breve{u}_{\mathrm{pos},k} \\ \breve{u}_{\mathrm{neg},k} \end{bmatrix} = \begin{bmatrix} \max(u^\star_{\mathrm{neg},k} + u^\star_{\mathrm{pos},k}, 0) \\ \min(u^\star_{\mathrm{neg},k} + u^\star_{\mathrm{pos},k}, 0) \end{bmatrix}, \quad \breve{\mathbf{x}}_k = \mathbf{x}^\star_k, \quad \begin{bmatrix} \breve{v}_{\mathrm{pos},k} \\ \breve{v}_{\mathrm{neg},k} \end{bmatrix} = \begin{bmatrix} \max(v^\star_k, 0) \\ \min(v^\star_k, 0) \end{bmatrix}. \tag{6.15a}$$

However, since the position space is partitioned into three segments, the updating routine changes for the auxiliary position states according to

$$\begin{bmatrix} \breve{x}_{\overline{\mathrm{low}},k} \\ \breve{x}_{\mathrm{low},k} \end{bmatrix} = \begin{bmatrix} \max(x^\star_k, \underline{x}_{\mathrm{tr}}) \\ \min(x^\star_k, \underline{x}_{\mathrm{tr}}) \end{bmatrix}, \quad \begin{bmatrix} \breve{x}_{\mathrm{up},k} \\ \breve{x}_{\mathrm{mid},k} \end{bmatrix} = \begin{bmatrix} \max(\breve{x}_{\overline{\mathrm{low}},k}, \overline{x}_{\mathrm{tr}}) \\ \min(\breve{x}_{\overline{\mathrm{low}},k}, \overline{x}_{\mathrm{tr}}) \end{bmatrix}. \tag{6.15b}$$

Furthermore, the correct signs are now given by

$$\sigma_{x_{\mathrm{mu}},k} = \mathrm{sign}(\breve{x}_{\overline{\mathrm{low}},k} - \overline{x}_{\mathrm{tr}}), \quad \sigma_{x_{\mathrm{lm}},k} = \mathrm{sign}(x^\star_k - \underline{x}_{\mathrm{tr}}), \quad \sigma_{v,k} = \mathrm{sign}(v^\star_k). \tag{6.15c}$$

With $i \in \mathcal{I}_{\mathrm{coll}}$ and $k \in \mathcal{K}$, the convex QP problem is formulated as

$$\min_{\mathbf{P}} \quad J_{\mathrm{ss}} + \tau \sum_{k \in \mathcal{K}} l_{\mathrm{abs},x_{\mathrm{mu}},k} + l_{\mathrm{abs},x_{\mathrm{lm}},k} + l_{\mathrm{abs},v,k} \quad \text{with} \tag{6.16a}$$

$$\begin{aligned} l_{\mathrm{abs},x_{\mathrm{mu}},k} &\coloneqq (x_{\mathrm{up},k} - x_{\mathrm{mid},k}) - \sigma_{x_{\mathrm{mu}},k}\,(x_{\overline{\mathrm{low}},k} - \overline{x}_{\mathrm{tr}}) \\ l_{\mathrm{abs},x_{\mathrm{lm}},k} &\coloneqq (x_{\overline{\mathrm{low}},k} - x_{\mathrm{low},k}) - \sigma_{x_{\mathrm{lm}},k}\,(x_k - \underline{x}_{\mathrm{tr}}) \\ l_{\mathrm{abs},v,k} &\coloneqq (v_{\mathrm{pos},k} - v_{\mathrm{neg},k}) - \sigma_{v,k}\, v_k \end{aligned} \tag{6.16b}$$

$$\text{s.t.} \quad \mathbf{c}_{\mathrm{coll},i} \overset{(5.4)}{=} \mathbf{0} \;\text{ with }\; \mathbf{f}_k = \begin{bmatrix} v_k \\ \frac{m\,g - F^{\mathrm{SSC}}_{d,k} - F^{\mathrm{SSC}}_{c,3,k}}{m} \end{bmatrix}, F^{\mathrm{SSC}}_{d,k} = u_{\mathrm{neg},k} + u_{\mathrm{pos},k}, \tag{6.16c}$$

$$F^{\mathrm{SSC}}_{c,3,k} = c_{\mathrm{up}}\,(x_{\mathrm{up},k} - \overline{x}_{\mathrm{tr}}) + c_{\mathrm{mid}}\, x_{\mathrm{mid},k} + c_{\mathrm{low}}\,(x_{\mathrm{low},k} - \underline{x}_{\mathrm{tr}})$$

$$\begin{array}{lll} & u_{\mathrm{pos},k} \in \mathcal{U}_{\mathrm{pos}} \coloneqq \left[0, \overline{F}_d\right] & x_{\overline{\mathrm{low}},k} \in \mathcal{X}_{\overline{\mathrm{low}}} \coloneqq [\underline{x}_{\mathrm{tr}}, \overline{x}] \\ x_k \in \mathcal{X} \coloneqq [\underline{x}, \overline{x}]\,, & u_{\mathrm{neg},k} \in \mathcal{U}_{\mathrm{neg}} \coloneqq [\underline{F}_d, 0] & x_{\mathrm{low},k} \in \mathcal{X}_{\mathrm{low}} \coloneqq [\underline{x}, \underline{x}_{\mathrm{tr}}] \\ v_k \in \mathcal{V} \coloneqq [\underline{v}, \overline{v}]\,, & v_{\mathrm{pos},k} \in \mathcal{V}_{\mathrm{pos}} \coloneqq [0, \overline{v}] & x_{\mathrm{up},k} \in \mathcal{X}_{\mathrm{up}} \coloneqq [\overline{x}_{\mathrm{tr}}, \overline{x}] \\ & v_{\mathrm{neg},k} \in \mathcal{V}_{\mathrm{neg}} \coloneqq [\underline{v}, 0] & x_{\mathrm{mid},k} \in \mathcal{X}_{\mathrm{mid}} \coloneqq [\underline{x}_{\mathrm{tr}}, \overline{x}_{\mathrm{tr}}] \end{array}, \tag{6.16d}$$

$$\hat{\mathbf{h}}_{\mathrm{zcs},p,k} = \begin{bmatrix} -\overline{d} & 1 \\ \underline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\mathrm{pos},k} \\ u_{\mathrm{pos},k} \end{bmatrix} \leq \mathbf{0}, \; \hat{\mathbf{h}}_{\mathrm{zcs},n,k} = \begin{bmatrix} -\underline{d} & 1 \\ \overline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\mathrm{neg},k} \\ u_{\mathrm{neg},k} \end{bmatrix} \leq \mathbf{0}, \tag{6.16e}$$

$$\begin{aligned} g_{\mathrm{aff},x_{mu},k} &= x_{\mathrm{mid},k} + x_{\mathrm{up},k} - (x_{\overline{\mathrm{low}},k} + \overline{x}_{\mathrm{tr}}) = 0 \\ g_{\mathrm{aff},x_{lm},k} &= x_{\mathrm{low},k} + x_{\overline{\mathrm{low}},k} - (x_k + \underline{x}_{\mathrm{tr}}) = 0 \\ g_{\mathrm{aff},v,k} &= v_{\mathrm{pos},k} + v_{\mathrm{neg},k} - v_k = 0 \end{aligned}, \tag{6.16f}$$

$$\underbrace{\begin{bmatrix} l_{\mathrm{abs},x_{\mathrm{mu}},k}\big|_{\sigma_{x_{\mathrm{mu}},k}=+1} \\ l_{\mathrm{abs},x_{\mathrm{mu}},k}\big|_{\sigma_{x_{\mathrm{mu}},k}=-1} \end{bmatrix}}_{=\mathbf{h}_{\mathrm{abs},x_{\mathrm{mu}},k}} \geq \mathbf{0}, \underbrace{\begin{bmatrix} l_{\mathrm{abs},x_{\mathrm{lm}},k}\big|_{\sigma_{x_{\mathrm{lm}},k}=+1} \\ l_{\mathrm{abs},x_{\mathrm{lm}},k}\big|_{\sigma_{x_{\mathrm{lm}},k}=-1} \end{bmatrix}}_{=\mathbf{h}_{\mathrm{abs},x_{\mathrm{lm}},k}} \geq \mathbf{0}, \underbrace{\begin{bmatrix} l_{\mathrm{abs},v,k}\big|_{\sigma_{v,k}=+1} \\ l_{\mathrm{abs},v,k}\big|_{\sigma_{v,k}=-1} \end{bmatrix}}_{=\mathbf{h}_{\mathrm{abs},v,k}} \geq \mathbf{0}, \tag{6.16g}$$

$$\mathbf{u}_0 = \begin{bmatrix} \max(\underline{d}\, v_{\mathrm{iv}}, 0) \\ \min(\underline{d}\, v_{\mathrm{iv}}, 0) \end{bmatrix}, \quad \mathbf{x}_0 = \mathbf{x}_{\mathrm{iv}}. \tag{6.16h}$$

The termination criterion is computed via

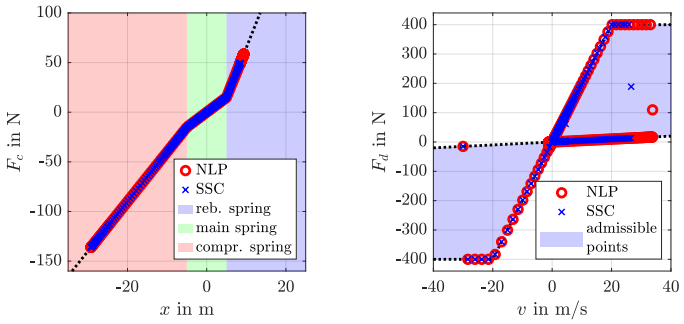
$$\bar{l}_{\text{abs}} = \left\| \begin{bmatrix} l_{\text{abs},x_{\text{mu}},0} \\ l_{\text{abs},x_{\text{lm}},0} \\ l_{\text{abs},v,0} \end{bmatrix}, \ldots, \begin{bmatrix} l_{\text{abs},x_{\text{mu}},n_{\text{seg}}} \\ l_{\text{abs},x_{\text{lm}},n_{\text{seg}}} \\ l_{\text{abs},v,n_{\text{seg}}} \end{bmatrix} \right\|_{\max} . \tag{6.17}$$

In order to analyse the effect of the initialisation on the computed solution, several initial guesses are fed to the SSC algorithm aiming at solving the OP with initial value $\mathbf{x}_{\text{iv},1}$ considering $n_{\text{seg}} = 250$ collocation segments. The maximum number of iterations is increased to $q_{\max} = 25$ to check if poor approximations eventually converge. Using the solution of the NLP optimiser $\mathbf{u}_{\text{opt}}^{\text{NLP}}$ and $\mathbf{X}_{\text{opt}}^{\text{NLP}}$ as reference, the initial guesses are generated using

$$u_{\text{pos},k}^{\star} = k_{\text{init}} \max\left(u_{\text{opt},k}^{\text{NLP}}, 0\right), \; u_{\text{neg},k}^{\star} = k_{\text{init}} \min\left(u_{\text{opt},k}^{\text{NLP}}, 0\right), \; \mathbf{x}_k^{\star} = k_{\text{init}} \, \mathbf{x}_{\text{opt},k}^{\text{NLP}} \tag{6.18}$$

with $k_{\text{init}} \in \{0.0, 0.5, 0.75, 0.95, 1.0, 1.05, 10.0\}$ and fed to the SSC update routine. The computations are compared with the results of the NLP problem, which is initialised with the same guesses. The results of the individual optimisations are listed in Table 6.4. The computed values for the spring force and damper force are depicted for $k_{\text{init}} = 0$ in Fig. 6.4a and Fig. 6.4b, respectively. Trajectories for selected solutions are illustrated in Fig. 6.5.

As Fig. 6.4a and Fig. 6.4b indicate for $k_{\text{init}} = 0$, all solutions satisfy the prescribed constraints: The spring forces lie on the piecewise linear branches and the damper forces are within the admissible area. Considering the objective value J_{ss} and mean error $\bar{e}_{\text{ss}}$ in Table 6.4, the NLP solver converges to the same solution independently which initial guess is supplied. Thus, it is assumed that the computed NLP solution represents the global optimum or is at least very close to it. However, initialisation can have a massive impact on the computation time. Generally, the guesses close to the solution result in shorter computation times. It is striking that both algorithms have the most difficulty converging for $k_{\text{init}} = 0.5$, which is not the guess with the biggest difference to the optimal



(a) Piecewise linear spring force with three segments.

(b) ZCS for damper force.

Figure 6.4: Optimisation results for SMO application with three-segmented nonlinear spring for $k_{\text{init}} = 0$.

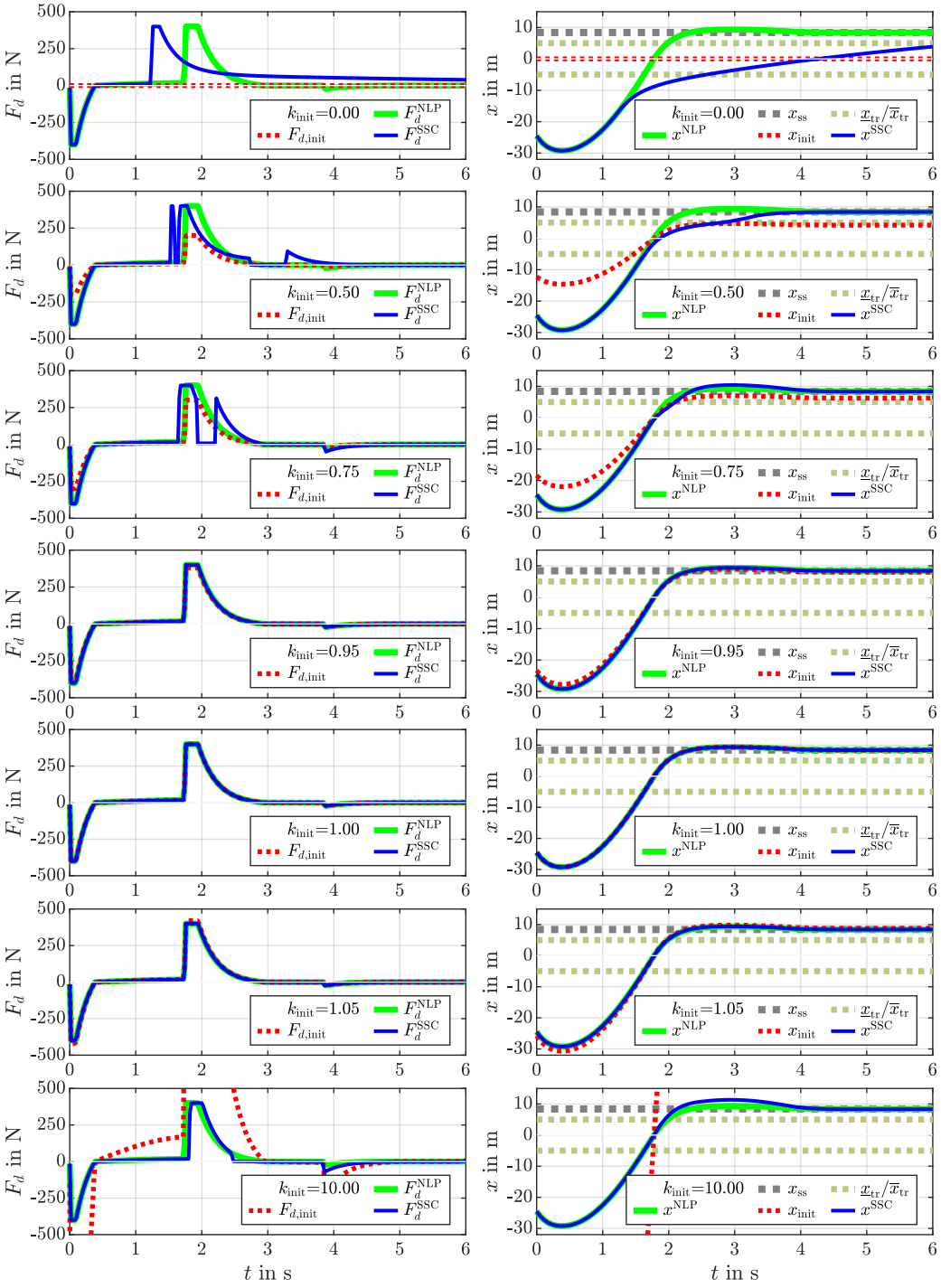


Figure 6.5: Solutions of SSC optimisations with initial value $\mathbf{x}_{\text{iv},1}$ and $n_{\text{seg}} = 250$ collocation segments for varying initial guesses.

Table 6.4: Analysis of sensitivity regarding initial guess.

	method		**results**					
k_{init}[-]	NLP	SSC	J_{ss}[m^{2}s]	$\overline{e}_{\text{ss}}$[m]	$\overline{e}_{F_d}$[N]	q[-]	$t_{\text{opt},\Sigma}$[s]	t_Σ[s]
0.0	✓		1603.95	5.47	-	1	4.589	4.589
0.0		✓	2118.49	9.91	46.63	7	0.493 (10.74%)	0.798 (17.38%)
0.5	✓		1603.95	5.47	-	1	6.347	6.347
0.5		✓	1635.20	5.93	14.43	19	1.450 (22.85%)	2.136 (33.65%)
0.75	✓		1603.95	5.47	-	1	0.557	0.557
0.75		✓	1612.62	5.67	15.77	2	0.179 (32.14%)	0.324 (58.26%)
0.95	✓		1603.95	5.47	-	1	0.473	0.473
0.95		✓	1603.95	5.47	0.01	1	0.075 (15.86%)	0.199 (42.02%)
1.0	✓		1603.95	5.47	-	1	0.417	0.417
1.0		✓	1603.95	5.47	0.01	1	0.072 (17.28%)	0.191 (45.79%)
1.05	✓		1603.95	5.47	-	1	0.427	0.427
1.05		✓	1603.96	5.47	0.20	1	0.084 (19.67%)	0.202 (47.37%)
10.0	✓		1603.95	5.47	-	1	0.981	0.981
10.0		✓	1610.74	5.72	6.71	2	0.149 (15.19%)	0.306 (31.22%)

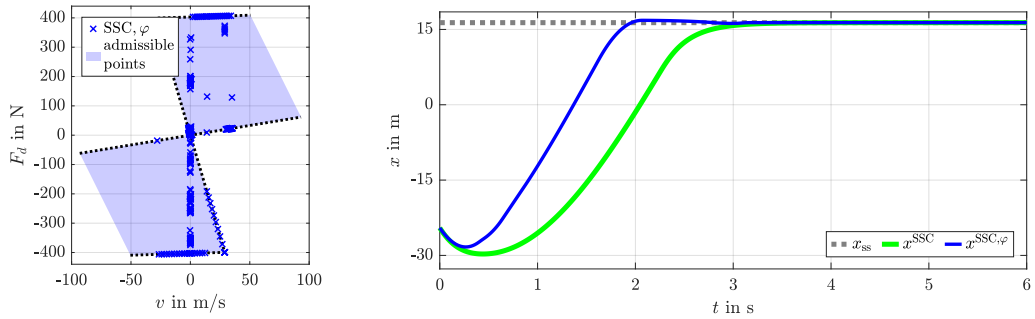
k_{init}: perturbation parameter for initialisation error. NLP: *IPOPT* solving NLP problem. SSC: SSC Algorithm 2 with QP problem (6.16). J_{ss}: main objective (6.2). $\overline{e}_{\text{ss}}/\overline{e}_{F_d}$: error metrics (6.11) and (6.10). q: # of superordinate iterations. $t_{\text{opt},\Sigma}$: cumulative time spent in NLP/QP solver. t_Σ: total computation time.

solution. The SSC algorithm converges significantly faster than the NLP solver for all initial guesses. The number of superordinate iterations required by the SSC algorithm is small for good initial guesses. As for $k_{\text{init}} = 0.5$, a high number of superordinate iterations can still yield a faster convergence compared to the NLP solver. As already mentioned, the SSC algorithm represents a local method and does not necessarily provide the global optimum, which is reflected in the results. Considering the objective value J_{ss} and mean error $\overline{e}_{\text{ss}}$, only the SSC solutions for $k_{\text{init}} \geq 0.5$ are roughly in the vicinity of the global solution. The mean deviance between the input trajectories $\overline{e}_{F_d}$ is then rather small. However, the general shape of the input trajectories in Fig. 6.5 differs for $k_{\text{init}} \leq 0.75$ from the NLP solution, which is reflected in the deviance value. Even though the SSC solutions may not represent the global optimum, the solutions satisfy the prescribed constraints since $q < q_{\max}$. Thus, they represent a local solution of the piecewise linear problem, which corresponds to the original problem for the example at hand. The results for $k_{\text{init}} = \{0.95, 1.0, 1.05\}$ confirm that the SSC algorithm converges to the global optimum if the initial guess is in a close vicinity of it.
As mentioned in Section 5.4, increasing the number of space divisions to depict the three-segmented spring raises computation time. This can be confirmed by comparing solution times of the SSC algorithm for $k_{\text{init}} = 0.0$: For the two-segmented spring $t^{\text{SSC}}_{\text{opt},\Sigma} = 0.262$s,

$t_{\Sigma}^{\mathrm{SSC}} = 0.329\mathrm{s}$ and $q = 4$ holds whereas $t_{\mathrm{opt},\Sigma}^{\mathrm{SSC}} = 0.493\mathrm{s}$, $t_{\Sigma}^{\mathrm{SSC}} = 0.798\mathrm{s}$ and $q = 7$ holds for the spring with three segments. Thus, more iterations are required and the average computation time per iteration is increased by 7.52% and 38.6% for $t_{\mathrm{opt},\Sigma}^{\mathrm{SSC}}$ and $t_{\Sigma}^{\mathrm{SSC}}$, respectively. This highlights the importance of an implementation in C-Code to reduce the time required for the updating routine and therefore the overall computation time.

6.1.4 Rotated Space Splitting

As elaborated in Section 5.3.1, some sets require to be split using a rotated straight line. Without going into detail, the approach is applied to the example of a hanging SMO with fully linear spring presented in Section 6.1.1. The OP is parametrised with initial value $\mathbf{x}_{\mathrm{iv},1}$ and $n_{\mathrm{seg}} = 250$ collocation segments. The damper set is rotated by $\varphi = 7°$ resulting in the set depicted in Fig. 6.6a. This is implemented by applying the changes (5.34) to the convex QP problem with fully linear spring. As depicted in Fig. 6.6a, the SSC algorithm computes damper forces that all lie within the admissible area. Although this example represents an artificial use-case, the corresponding position trajectories in Fig. 6.6b prove that the rotated set yields a reduction of the objective. Starting with $x_{\mathrm{iv},1} < x_{\mathrm{ss}}$ in a compression phase with $v_{\mathrm{iv},1} < 0$, the mass is first decelerated and then accelerated using $F_d < 0$ at $v > 0$, hence active forces. These active forces enable a faster attainment of the steady-state position. Due to the similarity to the previous examples, no further discussion is given.



(a) Rotated ZCS for damper force.

(b) Position trajectory. x^{SSC}: original damper force set. $x^{\mathrm{SSC},\varphi}$: rotated damper force set.

Figure 6.6: SSC optimisation results for SMO application with fully linear spring and rotated damper set.

6.2 Velocity-Maximisation for Vehicular Drag Race

In this section, a longitudinal single-track vehicle model with static wheel loads is considered in an OP that aims at maximising the velocity on a straight track. The corresponding parameters for model and optimisation are listed and described in Table 6.5. Assuming steady-state conditions for the wheel dynamics while neglecting rolling resistances, (4.3c) simplifies to

$$J_f\,\dot{\omega}_f = \cancelto{0}{T_f} - F_{x,f}\,r_f \overset{!}{=} 0 \quad \Rightarrow \quad F_{x,f} = 0 \tag{6.19a}$$

$$J_r\,\dot{\omega}_r = T_r - F_{x,r}\,r_r \overset{!}{=} 0 \quad \Rightarrow \quad T_r = F_{x,r}\,r_r \tag{6.19b}$$

for the front and rear wheel, respectively. Applying (6.19a) to (4.3a) results in

$$\dot{v}_x = \frac{1}{m}\left(\cancelto{0}{F_{x,f}} + F_{x,r} - F_{x,\mathrm{air}}\right). \tag{6.19c}$$

A simplified rear wheel tire force $F_{x,r}$ is computed using the shape curve $f_{x,r}$:

$$F_{x,r} = \mu\,F_{z,s,r}\,D_x\,f_{x,r} \quad \text{with} \quad f_{x,r} = f_{x,r,\mathrm{nl}} \coloneqq \sin\Big(C_x\,\arctan\left(B_x\,\lambda_{x,r}\right)\Big). \tag{6.19d}$$

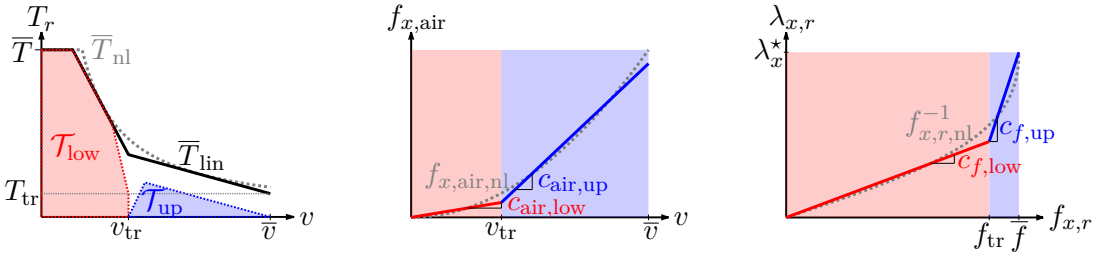
Therein the static rear wheel load $F_{z,s,r}$ is computed according to (4.8a). Considering (6.19b) and (6.19d), the shape curve satisfies

$$f_{x,r} = \frac{1}{\mu\,F_{z,s,r}\,D_x\,r_r}\,T_r =: k_f\,T_r. \tag{6.19e}$$

According to (4.6a), the aerodynamic drag force is given by

$$F_{x,\mathrm{air}} = \frac{1}{2}\,\rho_{\mathrm{air}}\,c_{\mathrm{air},x}\,A_{\mathrm{air}}\,f_{x,\mathrm{air}} =: k_{\mathrm{air}}\,f_{x,\mathrm{air}} \quad \text{with} \quad f_{x,\mathrm{air}} = f_{x,\mathrm{air,nl}} \coloneqq v_x^2. \tag{6.19f}$$

Thus, the system behaviour can be described using the rear wheel torque T_r as input and the longitudinal velocity $v \coloneqq v_x$ as state. However, the aerodynamic drag force (6.19f)



(a) ZCS for rear-axle torque. **(b)** Piecewise linear shape curve for air resistance force. **(c)** Piecewise linear shape curve for inverted tire force.

Figure 6.7: Nonconvexities in drag race application.

represents a nonlinearity resulting in nonconvex equality constraints when optimal control is applied. Furthermore, the torque is restricted by the nonconvex set

$$0 \leq T_r \leq \overline{T}_{\mathrm{nl}} \coloneqq \min\left(\overline{T},\ \frac{\overline{P}\, r_r}{v_x}\right) \tag{6.20}$$

depicted in Fig. 6.7a. The application of the SSC approach to solve the nonconvex OP is presented in the following sections.

Table 6.5: Parameters for drag race application.

	Symbol	Value	Unit	Description
system	g	9.81	$\mathrm{m/s^2}$	gravitational acceleration
	m	2000.00	kg	total mass of vehicle
	r_r	0.33	m	rear tire radius
	μ	1.00	–	road friction coefficient
	ρ_{air}	1.20	$\mathrm{kgm^{-3}}$	air density
	$c_{\mathrm{air},x}$	0.31	–	longitudinal drag coefficient
	A_{air}	2.44	$\mathrm{m^2}$	cross-span area of vehicle
	$D_x/C_x/B_x$	1.50/1.80/1.90	–	tire friction coefficient
	$F_{z,s,r}$	9704.73	N	static wheel load of rear axle
optimisation	$\overline{T}$	10446.00	Nm	maximum torque
	$\overline{P}$	390560.00	W	maximum power
	$\overline{v}$	69.44	m/s	maximum velocity
	v_{tr}	26.40	m/s	transition point for velocity
	$c_{\mathrm{air,low}}/c_{\mathrm{air,up}}$	16.23/92.90	m/s	slope of lower/upper aerodynamic force
	$\overline{f}$	1.00	-	maximum value for tire force shape curve
	f_{tr}	0.87	-	transition point for tire force shape curve
	$c_{f,\mathrm{low}}/c_{f,\mathrm{up}}$	0.33/2.66	-	slope of lower/upper inverse tire force shape curve
	$a_{\mathrm{up},1}$	-243759.69	Ns	slope of upper torque subset
	$b_{\mathrm{up},1}$	-6435255.81	Nm	vertical intercept of upper torque subset
	$a_{\mathrm{up},2}$	56.70	Ns	slope of upper torque subset
	$b_{\mathrm{up},2}$	3997.60	Nm	vertical intercept of upper torque subset
	$a_{\mathrm{low},1}$	387.38	Ns	slope of lower torque subset
	$b_{\mathrm{low},1}$	14121.19	Nm	vertical intercept of lower torque subset
	$a_{\mathrm{low},2}$	244203.78	Ns	slope of lower torque subset
	$b_{\mathrm{low},2}$	6448435.76	Nm	vertical intercept of lower torque subset
	ε_g	10^{-6}	-	constraint violation tolerance
	$\underline{\tau}/\overline{\tau}$	$1.00/10^4$	-	initial/final value for penalty parameter

6.2.1 Optimisation Problems

The drag race application is solved using the NLP solver *IPOPT* as well as the SSC algorithm with *Gurobi* as underlying QP-solver. The NLP problem and convex QP problem are listed subsequently.

Nonlinear, Nonconvex Optimisation Problem For the solution of the OP via NLP, the decision variables are chosen as

$$\mathbf{u} = \begin{bmatrix} T_{r,0} & T_{r,0.5} & \dots & T_{r,n_{\text{seg}}} \end{bmatrix} \in \mathbb{R}^{1 \times n_{\text{coll}}} \tag{6.21a}$$

$$\mathbf{x} = \begin{bmatrix} v_0 & v_{0.5} & \dots & v_{n_{\text{seg}}} \end{bmatrix} \in \mathbb{R}^{1 \times n_{\text{coll}}}. \tag{6.21b}$$

With $i \in \mathcal{I}_{\text{coll}}$ and $k \in \mathcal{K}$, the nonconvex OP is given by

$$\min_{\mathbf{u},\,\mathbf{x}} \quad J_{\text{main}} = \sum_{i=0}^{n_{\text{seg}}-1} -\frac{\Delta_i}{6} \left(v_i + 4\, v_{i+\frac{1}{2}} + v_{i+1} \right) \tag{6.22a}$$

$$\text{s.t.} \quad c_{\text{coll},i} \overset{(5.4)}{=} 0 \;\text{ with }\; f_k = \frac{1}{m} \left(\frac{T_{r,k}}{r_r} - k_{\text{air}}\, v_k^2 \right), \tag{6.22b}$$

$$T_{r,k} \in \mathcal{U} \coloneqq \left[0, \overline{T} \right], \quad v_k \in \mathcal{V} \coloneqq [0, \overline{v}], \tag{6.22c}$$

$$T_{r,k}\, \frac{v_k}{r_r} \leq \overline{P}, \tag{6.22d}$$

$$\lambda_{x,r,k} \coloneqq \frac{1}{B_x} \tan \left(\frac{1}{C_x} \arcsin(k_f\, T_{r,k}) \right) \leq 0.95\, \lambda_x^\star =: \overline{\lambda}_{x,r}, \tag{6.22e}$$

$$T_{r,0} = 0, \quad v_0 = 0. \tag{6.22f}$$

Using Simpson quadrature (3.14), the objective function (6.22a) aims at maximising the velocity over the time grid $t \in [t_0, t_f] = [0, 20]$ with constant segment width $\Delta_i = \Delta = \frac{t_f - t_0}{n_{\text{seg}}}$. The collocation constraints (6.22b) ensure satisfaction of the differential equation. Bounds on decision variables and limitations on the usable power are given by (6.22c) and (6.22d), respectively. In (6.22e), the tire slip is limited from above to 95% of the optimal slip value

$$\lambda_x^\star \coloneqq \frac{1}{B_x} \tan \left(\frac{\pi}{2\, C_x} \right) \tag{6.23}$$

to maintain a stability margin in presence of model uncertainties. Thus, the tire force remains on the stable branch below the optimal slip. The initial condition is prescribed via (6.22f). The NLP problem (6.22) is nonconvex due to the nonlinear right-hand side of the differential equation in collocation constraints (6.22b) and the nonconvex input restriction (6.22d).

Convex QP Problem for Iterative Solution In order to generate a convex QP subproblem, the state space is split at $v = v_{\mathrm{tr}}$ using the auxiliary state variables $v_{\mathrm{low}} \in [0, v_{\mathrm{tr}}]$ and $v_{\mathrm{up}} \in [v_{\mathrm{tr}}, \overline{v}]$. Via the auxiliary input variables $T_{\mathrm{low}} \in \left[0, \overline{T}\right]$ and $T_{\mathrm{up}} \in \left[0, \overline{T}\right]$, the rear axle torque is represented by the affine mapping function

$$T_r = \Phi_u = T_{\mathrm{low}} + T_{\mathrm{up}}. \tag{6.24}$$

The admissible torque set is defined using the two convex subsets $\mathcal{T}_{\mathrm{low}}$ and $\mathcal{T}_{\mathrm{up}}$ illustrated in Fig. 6.7a: The goal of the splitting procedure is to approximate $\overline{T}_{\mathrm{nl}}$ by the piecewise linear limit $\overline{T}_{\mathrm{lin}}$. As mentioned in Section 5.3.1, ZCSs that are not connected exclusively in a single point require further precautions at the transition. Since torque is required to increase velocity, pulling both subsets to $T_{\mathrm{low}} = 0 = T_{\mathrm{up}}$ at $v = v_{\mathrm{tr}}$ can result in longer transition periods with $v = v_{\mathrm{tr}}$ or even limit the solution to $v < v_{\mathrm{tr}}$. Thus, the subset $\mathcal{T}_{\mathrm{low}}$ enables the use of $T_{\mathrm{low}} = T_{\mathrm{tr}} \coloneqq -a_{\mathrm{low},2}\, v_{\mathrm{tr}} + b_{\mathrm{low},2}$ at $v \geq v_{\mathrm{tr}}$.
Furthermore, the nonlinearity in the air resistance formula (6.19f) is depicted using piecewise linearities as shown in Fig. 6.7b. For a stability margin, the OP imposes a limit on the tire slip. Within the stable tire region, the tire force is strictly increasing over the slip. Thus, this could be implemented using the corresponding limit on the tire force (6.19d) and posing an upper boundary on the torque (6.19b):

$$T_r \leq r_r\, F_{x,r}(\lambda_{x,r} = \overline{\lambda}_{x,r}) = r_r\, \mu\, F_{z,s,r}\, D_x\, \sin\Big(C_x\, \arctan\Big(B_x\, \overline{\lambda}_{x,r}\Big)\Big). \tag{6.25}$$

However, another approach is pursued subsequently. The tire force shape curve is also split into two linear segments as depicted in Fig. 6.7c in order to demonstrate that the SSC approach is capable of considering multiple univariate nonlinearities. Using relation (6.19e), the tire force shape curve is split at $f_{x,r} = f_{\mathrm{tr}}$ via the auxiliary variables $f_{\mathrm{low}} \in [0, f_{\mathrm{tr}}]$ and $f_{\mathrm{up}} \in \left[f_{\mathrm{tr}}, \overline{f}\right]$. Hence, following decision variables are used for the OP:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_{0.5} & \dots & \mathbf{u}_{n_{\mathrm{seg}}} \end{bmatrix} \in \mathbb{R}^{2 \times n_{\mathrm{coll}}} \text{ with } \mathbf{u}_k \coloneqq \begin{bmatrix} T_{\mathrm{up},k} \\ T_{\mathrm{low},k} \end{bmatrix} \tag{6.26a}$$

$$\mathbf{x} = \begin{bmatrix} v_0 & v_{0.5} & \dots & v_{n_{\mathrm{seg}}} \end{bmatrix} \in \mathbb{R}^{1 \times n_{\mathrm{coll}}} \tag{6.26b}$$

$$\mathbf{V}_{\mathrm{aux}} = \begin{bmatrix} \mathbf{v}_{\mathrm{aux},0} & \mathbf{v}_{\mathrm{aux},0.5} & \dots & \mathbf{v}_{\mathrm{aux},n_{\mathrm{seg}}} \end{bmatrix} \in \mathbb{R}^{2 \times n_{\mathrm{coll}}} \text{ with } \mathbf{v}_{\mathrm{aux},k} \coloneqq \begin{bmatrix} v_{\mathrm{up},k} \\ v_{\mathrm{low},k} \end{bmatrix} \tag{6.26c}$$

$$\mathbf{F}_{\mathrm{aux}} = \begin{bmatrix} \mathbf{f}_{\mathrm{aux},0} & \mathbf{f}_{\mathrm{aux},0.5} & \dots & \mathbf{f}_{\mathrm{aux},n_{\mathrm{seg}}} \end{bmatrix} \in \mathbb{R}^{2 \times n_{\mathrm{coll}}} \text{ with } \mathbf{f}_{\mathrm{aux},k} \coloneqq \begin{bmatrix} f_{\mathrm{up},k} \\ f_{\mathrm{low},k} \end{bmatrix}. \tag{6.26d}$$

Accumulating the decision variables in $\mathbf{P} \coloneqq \{\mathbf{U}, \mathbf{x}, \mathbf{V}_{\text{aux}}, \mathbf{F}_{\text{aux}}\}$, the convex QP problem is given with $i \in \mathcal{I}_{\text{coll}}$ and $k \in \mathcal{K}$ by

$$\min_{\mathbf{P}} \quad \sum_{i=0}^{n_{\text{seg}}-1} j_{\text{main},i} + \tau \sum_{k\in\mathcal{K}} \left(l_{\text{abs},v,k} + l_{\text{abs},f,k}\right) \quad \text{with} \tag{6.27a}$$

$$\begin{aligned} j_{\text{main},i} &\coloneqq -\tfrac{\Delta_i}{6}\left(v_i + 4\, v_{i+\frac{1}{2}} + v_{i+1}\right) \\ l_{\text{abs},v,k} &\coloneqq (v_{\text{up},k} - v_{\text{low},k}) - \sigma_{v,k}\,(v_k - v_{\text{tr}}) \\ l_{\text{abs},f,k} &\coloneqq (f_{\text{up},k} - f_{\text{low},k}) - \sigma_{f,k}\,\left(k_f\,\left(T_{\text{up},k} + T_{\text{low},k}\right) - f_{\text{tr}}\right) \end{aligned} \tag{6.27b}$$

$$\text{s.t.} \quad c_{\text{coll},i} \overset{(5.4)}{=} 0 \;\text{ with }\; f_k = \tfrac{1}{m}\left(\tfrac{T_{\text{up},k}+T_{\text{low},k}}{r_r} - k_{\text{air}}\, f_{x,\text{air},k}\right), \tag{6.27c}$$

$$f_{x,\text{air},k} \coloneqq c_{\text{air,low}}\, v_{\text{low},k} + c_{\text{air,up}}\,(v_{\text{up},k} - v_{\text{tr}})$$

$$\begin{aligned} &v_k \in \mathcal{V} \coloneqq [0, \overline{v}]\,, \\ &T_{\text{up},k} \in \mathcal{U}_{\text{up}} \coloneqq \left[0, \overline{T}\right] \quad v_{\text{up},k} \in \mathcal{V}_{\text{up}} \coloneqq [v_{\text{tr}}, \overline{v}] \quad f_{\text{up},k} \in \mathcal{F}_{\text{up}} \coloneqq \left[f_{\text{tr}}, \overline{f}\right] \\ &T_{\text{low},k} \in \mathcal{U}_{\text{low}} \coloneqq \left[0, \overline{T}\right], \; v_{\text{low},k} \in \mathcal{V}_{\text{low}} \coloneqq [0, v_{\text{tr}}], \; f_{\text{low},k} \in \mathcal{F}_{\text{low}} \coloneqq [0, f_{\text{tr}}] \end{aligned} \tag{6.27d}$$

$$\underbrace{\begin{bmatrix} a_{\text{low},1} & 1 \\ a_{\text{low},2} & 1 \end{bmatrix}\begin{bmatrix} v_{\text{low},k} \\ T_{\text{low},k} \end{bmatrix} - \begin{bmatrix} b_{\text{low},1} \\ b_{\text{low},2} \end{bmatrix}}_{=\hat{\mathbf{h}}_{\text{zcs,low},k}} \leq \mathbf{0}, \quad \underbrace{\begin{bmatrix} a_{\text{up},1} & 1 \\ a_{\text{up},2} & 1 \end{bmatrix}\begin{bmatrix} v_{\text{up},k} \\ T_{\text{up},k} \end{bmatrix} - \begin{bmatrix} b_{\text{up},1} \\ b_{\text{up},2} \end{bmatrix}}_{=\hat{\mathbf{h}}_{\text{zcs,up},k}} \leq \mathbf{0}, \tag{6.27e}$$

$$\begin{aligned} g_{\text{aff},v,k} &= v_{\text{up},k} + v_{\text{low},k} - (v_k + v_{\text{tr}}) = 0 \\ g_{\text{aff},f,k} &= f_{\text{up},k} + f_{\text{low},k} - (k_f\,\left(T_{\text{up},k} + T_{\text{low},k}\right) + f_{\text{tr}}) = 0\,, \end{aligned} \tag{6.27f}$$

$$\mathbf{h}_{\text{abs},v,k} = \begin{bmatrix} l_{\text{abs},v,k}\big|_{\sigma_{v,k}=+1} \\ l_{\text{abs},v,k}\big|_{\sigma_{v,k}=-1} \end{bmatrix} \geq \mathbf{0}, \quad \mathbf{h}_{\text{abs},f,k} = \begin{bmatrix} l_{\text{abs},f,k}\big|_{\sigma_{f,k}=+1} \\ l_{\text{abs},f,k}\big|_{\sigma_{f,k}=-1} \end{bmatrix} \geq \mathbf{0}, \tag{6.27g}$$

$$\lambda_{x,r,k} \coloneqq c_{f,\text{low}}\, f_{\text{low},k} + c_{f,\text{up}}\,(f_{\text{up},k} - f_{\text{tr}}) \leq 0.95\,\lambda_x^\star =: \overline{\lambda}_{x,r}, \tag{6.27h}$$

$$\mathbf{u}_0 = \mathbf{0}, \quad v_0 = 0. \tag{6.27i}$$

The main objective in (6.27a) represents (6.22a) and aims at maximising the velocity. The maximisation of the velocity is used instead of a minimum-time objective since it yields a convex cost function without requiring the nonlinear transformation presented in Section 3.3.1. Since the main objective is linear, the resulting OP (6.27) represents an LP problem. Together with constraints (6.27f)-(6.27g), the remaining objective terms in (6.27a) ensure the correct switching for $\mathbf{V}_{\text{aux}}$ and $\mathbf{F}_{\text{aux}}$. The collocation constraints, bounds and initial value condition are considered via (6.27c), (6.27d) and (6.27i), respectively. The convex subsets $\mathcal{T}_{\text{low}}$ and $\mathcal{T}_{\text{up}}$ depicted in Fig. 6.7a are represented via the affine inequality constraints (6.27e) together with the inputs bounds $\mathcal{U}_{\text{low}}$ and $\mathcal{U}_{\text{up}}$. In (6.27h), the tire slip is limited from above to 95% of the optimal slip value $\lambda_x^\star$ to maintain a stability margin in presence of model uncertainties. Furthermore, the inclusion of a slip limit aims at demonstrating that multiple univariate nonlinear equalities can be depicted via SSC.

Using $T_k^\star \coloneqq T_{\text{low},k}^\star + T_{\text{up},k}^\star$, the updating routine is implemented as follows. The state

and auxiliary states are adjusted according to

$$\breve{v}_k = v_k^\star \tag{6.28a}$$

$$\breve{\mathbf{v}}_{\mathrm{aux},k} = \begin{bmatrix} \breve{v}_{\mathrm{up},k} \\ \breve{v}_{\mathrm{low},k} \end{bmatrix} = \begin{bmatrix} \max(v_k^\star, v_{\mathrm{tr}}) \\ \min(v_k^\star, v_{\mathrm{tr}}) \end{bmatrix}, \; \breve{\mathbf{f}}_{\mathrm{aux},k} = \begin{bmatrix} \breve{f}_{\mathrm{up},k} \\ \breve{f}_{\mathrm{low},k} \end{bmatrix} = \begin{bmatrix} \max(k_f\, T_k^\star, f_{\mathrm{tr}}) \\ \min(k_f\, T_k^\star, f_{\mathrm{tr}}) \end{bmatrix}. \tag{6.28b}$$

The correct signs are computed via

$$\sigma_{v,k} = \mathrm{sign}(v_k^\star - v_{\mathrm{tr}}), \quad \sigma_{f,k} = \mathrm{sign}(k_f\, T_k^\star - f_{\mathrm{tr}}). \tag{6.28c}$$

Due to the adjustment of the torque subsets at the transition, the initial guess for the auxiliary inputs is updated under consideration of (6.27e):

$$\breve{T}_{\mathrm{low},k} = \begin{cases} \min\left(-a_{\mathrm{low},1}\, v_k^\star + b_{\mathrm{low},1},\; -a_{\mathrm{low},2}\, v_k^\star + b_{\mathrm{low},2},\; \overline{T},\; T_k^\star\right) & \forall v \le v_{\mathrm{tr}}, \\ \min(T_k^\star,\; T_{\mathrm{tr}}) & \text{else} \end{cases} \tag{6.28d}$$

$$\breve{T}_{\mathrm{up},k} = \begin{cases} 0 & \forall v \le v_{\mathrm{tr}}, \\ \min\left(-a_{\mathrm{up},1}\, v_k^\star + b_{\mathrm{up},1},\; -a_{\mathrm{up},2}\, v_k^\star + b_{\mathrm{up},2},\; T_k^\star - \breve{T}_{\mathrm{low},k}\right) & \text{else} \end{cases}. \tag{6.28e}$$

The termination criterion is computed via

$$\bar{l}_{\mathrm{abs}} = \left\| \begin{bmatrix} l_{\mathrm{abs},v,0} \\ l_{\mathrm{abs},f,0} \end{bmatrix}, \ldots, \begin{bmatrix} l_{\mathrm{abs},v,n_{\mathrm{seg}}} \\ l_{\mathrm{abs},f,n_{\mathrm{seg}}} \end{bmatrix} \right\|_{\max}. \tag{6.29}$$

The results for the presented OPs are discussed in the following Section 6.2.2.

6.2.2 Results

The NLP problem (6.22) and the convex QP subproblem (6.27) within the SSC approach are solved for a number of $n_{\mathrm{seg}} \in \{10, 50, 100, 250\}$ collocation segments. Both algorithms are initialised assuming maximum velocity in all collocation points with corresponding torque limitations:

$$v_k^\star = \overline{v} \quad \text{and} \quad T_{r,k}^\star = \min\left(\overline{T}, \frac{\overline{P}\, r_r}{\overline{v}}\right) \quad \forall k \in \mathcal{K}. \tag{6.30}$$

The results are summarised in Table 6.6. The objective function values indicate that the NLP and SSC algorithm achieve similar results. As will be elaborated below, both methods identify the optimal solution of the problem provided to them. However, the approximation of the nonlinearities via piecewise linear segments introduces errors, which result in a marginally higher cost function for the SSC approach. The quality of the solutions can be graphically evaluated considering the velocity, torque and force trajectories depicted for $n_{\mathrm{seg}} = 100$ collocation segments in Fig. 6.8. The first subplot shows the trajectories for the velocity and auxiliary velocity variables. The switching occurs correctly at $v = v_{\mathrm{tr}}$. Analogously, the trajectories for the tire force shape in the second

subplot switch at $f_{x,r} = f_{\mathrm{tr}}$ as planned. After reaching the maximum velocity $\overline{v}$, the tire force oscillates in order to keep the velocity at the maximum level. If needed, these oscillations can be avoided by introducing an additional objective term penalising the derivative of the tire force. The third subplot depicts the convex subsets for the rear wheel torque. When accelerating, the upper tire slip bounds (6.22e) and (6.27h) limit the possible rear axle torque until the limiting factor is the torque characteristic curve $\overline{T}_{\mathrm{nl}}$ respectively $\overline{T}_{\mathrm{lin}}$ and finally the maximum velocity $\overline{v}$. This confirms the correctness of the results. As planned, the lower torque variable is kept constant at $T_{\mathrm{low}} = T_{\mathrm{tr}}$ when the velocity exceeds the transition point $v \geq v_{\mathrm{tr}}$. Then, both the lower and the upper torque variable are at their maximum limit and the cumulative torque reaches the desired linearly approximated torque boundary $T_r^{\mathrm{SSC}} = \overline{T}_{\mathrm{lin}}$. In contrast, the NLP problem follows the nonlinear maximum torque boundary $\overline{T}_{\mathrm{nl}} \geq \overline{T}_{\mathrm{lin}}$, which enables a higher torque. The fourth and fifth subplot illustrates the shape curve for the tire force and for the air resistance force, respectively. Both subplots confirm the correct switching and depiction of the piecewise linearities. The piecewise linear approximations of the tire force and air resistance force introduce errors. However, the main cause for the discrepancies between the NLP solution and SSC solution is the approximated torque set. The velocity trajectories of both solutions v^{NLP} and v^{SSC} roughly coincide as long as the slip constraint limits the applicable torque. Discrepancies between the solutions occur afterwards since the admissible torque set of the SSC approach is an approximation of the original set: The SSC torque set underestimates the original set so that it represents a subset and thus results in admissible torques. After reaching $v \approx 24\frac{\mathrm{m}}{\mathrm{s}}$ at $t \approx 3.4\mathrm{s}$, the admissible torque for the SSC approach is defined by the torque set and thus lower than for the NLP problem. This is the primary cause for the different velocity trajectories and thus different objective function values. For comparison, the NLP solution reaches the maximum velocity after 16.9 seconds and the SSC solution after 18.6 seconds. Since the convex torque subsets represent approximations of the original set and the nonlinear

Table 6.6: Optimisation results for drag race application.

n_{seg}[-]	OPT	J_{main}[m/s]	q[-]	$t_{\mathrm{opt},\Sigma}$[s]	t_{Σ}[s]
10	NLP	-930.46	1	0.019	0.019
10	SSC	-907.22	1	0.001 (5.26%)	0.088 (461.21%)
50	NLP	-958.78	1	0.055	0.055
50	SSC	-936.36	1	0.008 (14.55%)	0.098 (177.28%)
100	NLP	-961.33	1	0.096	0.096
100	SSC	-940.19	2	0.024 (25.04%)	0.127 (131.79%)
250	NLP	-962.81	1	0.232	0.232
250	SSC	-941.72	2	0.102 (43.97%)	0.235 (101.14%)

n_{seg}: # of collocation segments. NLP: *IPOPT* solving NLP problem (6.22). SSC: SSC Algorithm 2 with QP problem (6.27). J_{main}: main objective (6.22a). q: # of superordinate iterations. $t_{\mathrm{opt},\Sigma}$: cumulative time spent in NLP-/QP-solver. t_{Σ}: total computation time.

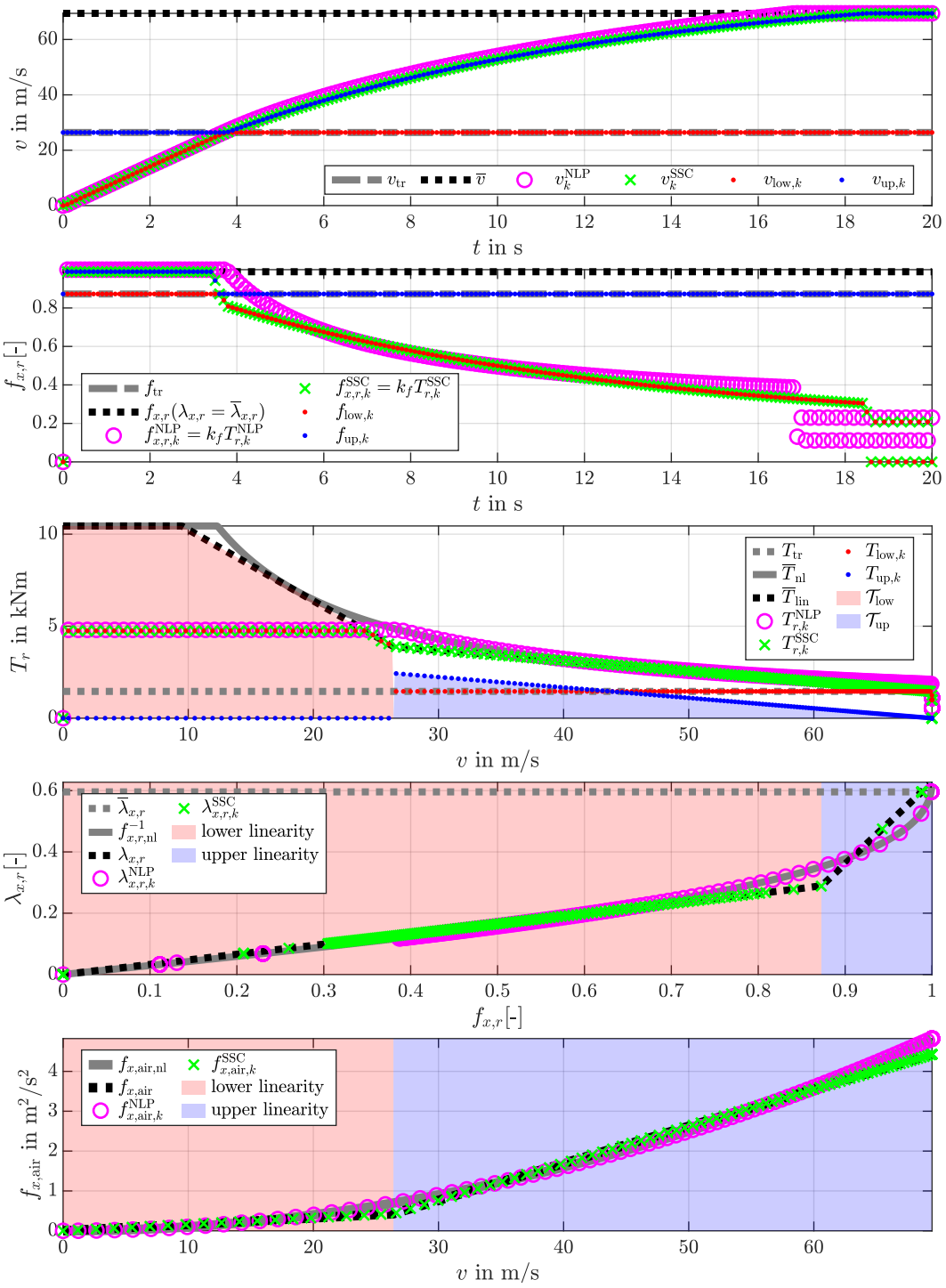


Figure 6.8: NLP and SSC solution for drag race application with $n_{\text{seg}} = 100$ collocation segments.

shape curves of air resistance force and tire force are also approximated, the solution can only be suboptimal. However, the accuracy can be increased by splitting the spaces into more segments, though this can increase computation time.
The SSC algorithm solves the problem requiring only a small number of $q \leq 2$ superordinate iterations, even though the rather simple initial guess (6.30) was provided. The benefits in pure optimisation time $t_{\text{opt},\Sigma}$ of the SSC algorithm reduce with increasing number of collocation segments. The reason for this are the additional decision variables introduced for space splitting, which scale with the number of collocation segments. This increases the OP size and thus the computational complexity according to (5.44). Comparing the cumulative optimisation time $t_{\text{opt},\Sigma}$ and the total computation time t_{Σ} in Table 6.6, the current example illustrates that the updating routine can substantially increase the total runtime such that the SSC algorithm takes longer than the NLP solver. However, only simple maximum, minimum and sign functions with linear computational complexity are used in the projection routine. As illustrated with Fig. 6.3, an implementation in C-Code would significantly reduce the computation time for this step. Furthermore, it would enable a direct access to the solver and thus eliminate the additional time required for the communication between *Julia* and the *Gurobi*-solver. Thus, a real-time implementation requires rewriting the algorithm in C-code, which is however subject of future work.

7 Conclusion and Outlook

Numerical optimisation is a powerful tool that is becoming more and more important due to the rapidly progressing development of computers. It enables making smart decisions in the context of control and can be used for evaluation and design purposes. This thesis has presented two methods for solving nonconvex NLP problems. Firstly, a general framework capable of solving complicated optimal control tasks for highly nonlinear systems has been presented. Secondly, a successive convexification procedure greatly reducing computation time for a special class of OCPs has been proposed. These two main research areas are concluded below offering possible directions for future works. This chapter partly echoes remarks from our previous publications [183, 185, 186].

7.1 Nonconvex Optimal Control for Highly Nonlinear Systems

Many OCPs for engineering applications require the solution of nonconvex and highly nonlinear OPs. The first part of this thesis has presented a procedure that is capable of solving these difficult problems. The method illustrated in Chapter 3 aims at solving minimum-time OCPs, however the approach is also applicable to problems with other objectives. A spatial transformation enables the reformulation of the problem in terms of a path parameter instead of the time. This simplifies some path constraints and casts the free final time OP to a problem with a fixed independent variable at the final point. A scaling procedure enhances convergence by normalising the domains of decision variables and transforming the domains of constraints and objective to suitable ranges. Adequate smoothing functions have been introduced to ensure the required differentiability of the OP, which is necessary for solvers that employ derivative information. Furthermore, a novel preprocessing approach for the generation of smooth reference paths has been presented. The transcription of the dynamic OP into a static one has been achieved via Hermite-Simpson collocation. This collocation method represents a good compromise between accuracy and sparsity of the OP, which improves computation time.
A vehicular application aiming at computing minimum lap times has been solved in Chapter 4 using the presented framework. The vehicle model has been validated using measurement data from a test vehicle. Model augmentations for BEVs have been proposed with special focus on a simple representation of EOL in OCPs. An initialisation routine using a triple PID-controller has been suggested for the generation of a suitable initial guess. After solving the OP, the optimal input trajectories can be interpreted to derive race strategies. The results quantify the relative benefit of EOL, which mainly reduces lap time at the race start and after corners by enabling higher drive torques. The low limit for the charging current of the battery pack also contributes to the preference for overloading on acceleration phases rather than deceleration phases. Moreover,

allocating the torques in accordance with the wheel loads has been identified to be optimal. In this regard, overloading is mainly applied to the rear motors respectively the outer motors during cornering. In order to recuperate as much energy as possible, the friction brake torques are only used when the charging current limit prohibits higher motor brake torques. Furthermore, a concurrent optimisation of passive vehicle parameters enables design optimisations. This procedure has been used to identify optimal gear ratios for the front and rear powertrain units as well as the dimensioning of the front and rear motors. The results show that the concurrent optimisation provides additional performance benefits highlighting the importance of considering the passive setup when deriving performance-optimal strategies. The approach can be used for an objective comparison of various powertrain topologies facilitating an accelerated development process of automotive vehicles. Due to the generality of the optimisation approach, such accelerated design processes can be applied to many machines or robotic systems.
Possible topics for future research are listed below:

- The convergence speed could be improved by introducing a mesh refinement strategy. The meshing can be concurrently optimised using additional decision variables. However, other approaches that avoid introducing further decision variables can be more expedient. The OP can be solved using an automatic scheme with gradually increasing number of collocation segments. The error between the original differential equation and its polynomial approximation on each segment can be used to determine if the segment needs to be bisected increasing the number of collocation segments for a subsequent optimisation [97]. The initial meshing can be designed using system knowledge: Collocation points should be positioned at inflexion points of the differential equation trajectories. The initial solution for the subsequent optimisation can be computed via interpolation of the preceding optimal solution.
- Alternatively, highly accurate solutions can be computed via indirect methods. The convergence of these methods tends to be quite sensitive in regards to the initial guess. Using the presented collocation method to compute a solution on a coarse grid, the solution can be fed to the indirect method as an initial guess. However, this requires a suitable approach for deriving adequate initial trajectories for the adjoint variables.
- An interesting direction from an vehicular application standpoint would be the consideration of temperature dependencies of electrical components. Although these relations would increase nonlinearity of the equations, the consideration of temperature effects increases the model accuracy. It would be useful to quantify the lap time impact of these effects and possibly find simple substitute models.

7.2 Space Splitting Convexification

The real-time application of optimal control methods represents a challenging task. As already mentioned, OPs for real-life applications are often nonconvex and thus hard to solve. Aiming at reducing computation time, a convexification approach called SSC has been presented in Chapter 5. The algorithm iteratively solves convex substitute problems that are updated after each iteration based on the preceding solution. The method is capable of considering two types of nonconvexities: ZCSs as well as equality constraints with possibly multiple univariate nonlinearities. Therein the nonlinearity can be represented without losses if it is piecewise linear and is approximated otherwise. The basic notion behind the procedure is representing the nonconvexities using piecewise affine segments that are interconnected using adequate constraints. Similar to previous techniques, a linearisation is used to simplify the equations. However, the linearisation is reduced to the binary result of a correct sign. On convergence, the accuracy of the solution is thus predefined by the design of the piecewise affine segments and less problematic regarding linearisation errors. This fact and the use of constraint relaxations result in a robust and generally quick convergence independent of the initially provided solution. Being a local method, the algorithm converges to a local optimum of the piecewise linearly approximated problem.
SSC has been used in Chapter 6 to solve nonconvex OPs for a SMO. Compared to an NLP solver, the SSC approach has provided similar results while greatly reducing computation time. Moreover, it has been confirmed that the initial guess can greatly influence the outcome however the global optimum is retrieved for guesses in a close vicinity around it. Furthermore, SSC has been employed to compute optimal trajectories for a vehicular drag race application. Using a conservative approximation of the original sets, the computed solution is suboptimal however follows the same general control strategy like the NLP solution.
Subsequently, possible directions for future research are disclosed:

- Even though it would not change the algorithm, using nondifferentiable constraint qualifications [230] could potentially avoid the AVC-smoothing required for continuous differentiability regarding the convergence proof.
- In order to mitigate the problem of suboptimal solutions, a perturbation mechanism could be used after converging in order to initiate an additional optimisation loop using the perturbed preceding solution as initial guess.
- Applying a more sophisticated updating procedure for the penalty parameter would improve computation times [135, 140, 148].
- Due to the low computation time and robust convergence, the SSC method seems to be a promising approach for real-time optimal control applications. In this context, using MPC could provide sufficiently good initial guesses [48, 71] resulting in good SSC solutions. For time-critical applications, it is recommended to implement the algorithm via tailored C-code. This would reduce the computation time of the projection routine and avoid the overhead introduced by the high-level routines of

JuMP and the *Gurobi*-wrapper. Then, the required time for the updating step can be greatly reduced to linear time complexity resulting in shorter runtimes.

- The current SSC approach can only consider univariate nonlinearities in equality constraints. Extending this approach to bivariate functions would enlarge the possible scope of applications.
- An extension of the approach to consider nonconvex keep-out zones would be valuable for collision avoidance applications. This could be implemented via an iterative linearisation around predefined or predicted trajectories [119]. Alternatively, the LnP approach [134] could be used requiring only the projection onto the collision avoidance constraints. Another possibility is to map the nonconvex, prohibited areas onto admissible, convex, inner approximations [28, 191].
- The comparison of the SSC algorithm with GO techniques for solving MIP problems is still to come. Furthermore, the piecewise linearly approximated problem could also be posed as MIP problem with special ordered sets. The solution computed via existing GO techniques would provide useful insights regarding the convergence domain of the SSC algorithm to the global optimum and comparative values for runtime.

Bibliography

[1] B. Acikmese and L. Blackmore. “Lossless convexification of a class of optimal control problems with non-convex control constraints”. In: Automatica 47.2, 2011, pp. 341–347.

[2] B. Acikmese and S. R. Ploen. “Convex programming approach to powered descent guidance for mars landing”. In: Journal of Guidance, Control, and Dynamics 30.5, 2007, pp. 1353–1366.

[3] S. Albrecht. “Modeling and numerical solution of inverse optimal control problems for the analysis of human motions”. PhD thesis. Technical University of Munich, 2014.

[4] A. Arab et al. “Motion planning for aggressive autonomous vehicle maneuvers”. In: IEEE International Conference on Automation Science and Engineering. 2016, pp. 221–226.

[5] A. Armaou and A. Ataei. “Piece-wise constant predictive feedback control of nonlinear systems”. In: Journal of Process Control 24.4, 2014, pp. 326–335.

[6] D. Axehill. *Controlling the level of sparsity in MPC*. 2014. arXiv: `1401.1369 [math.OC]`.

[7] R. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[8] A. Bemporad and M. Morari. “Control of systems integrating logic, dynamics, and constraints”. In: Automatica 35.3, 1999, pp. 407–427.

[9] A. Ben-Tal and A. Nemirovski. *Lectures on modern convex optimization*. Society for Industrial and Applied Mathematics, 2001.

[10] E. Bertolazzi, F. Biral, and M. Da Lio. “Symbolic-numeric efficient solution of optimal control problems for multibody systems”. In: Journal of Computational and Applied Mathematics 185.2, 2006, Special Issue: International Workshop on the Technological Aspects of Mathematics, pp. 404–421.

[11] E. Bertolazzi, F. Biral, and M. Lio. “Real-time motion planning for multibody systems - real life application examples”. In: Multibody System Dynamics 17, 2007,

[12] D. P. Bertsekas. *Dynamic programming and optimal control*. Vol. 1. Athena Scientific, 2005.

[13] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*. Society for Industrial and Applied Mathematics, 2010.

[14] J. Bezanson et al. “Julia: A fresh approach to numerical computing”. In: CoRR abs/1411.1607, 2014, arXiv: 1411.1607.

[15] N. D. Bianco, R. Lot, and M. Gadola. “Minimum time optimal control simulation of a GP2 race car”. In: Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 232.9, 2018, pp. 1180–1195.

[16] N. D. Bianco et al. “Comparison of direct and indirect methods for minimum lap time optimal control problems”. In: Vehicle System Dynamics 57.5, 2019, pp. 665–696.

[17] H. Bock and K. Plitt. “A multiple shooting algorithm for direct solution of optimal control problems”. In: IFAC Proceedings Volumes 17.2, 1984, 9th IFAC World Congress: A Bridge Between Control Science and Technology, pp. 1603–1608.

[18] C. d. Boor. *A practical guide to splines*. Springer New York, 2001.

[19] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[20] F. Braghin et al. “Race driver model”. In: Computers and Structures 86.13–14, 2008, pp. 1503–1516.

[21] C. Brandt, H.-P. Seidel, and K. Hildebrandt. “Optimal spline approximation via $\mathcal{L}_0$-minimization”. In: Computer Graphics Forum 34.2, 2015, pp. 617–626.

[22] D. Brayshaw and M. Harrison. “A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location”. In: Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 219.6, 2005, pp. 725–739.

[23] R. Bulirsch et al. “An optimal control approach to real-time vehicle guidance”. In: Mathematics - Key Technology for the Future: Joint Projects between Universities and Industry. Springer Berlin Heidelberg, 2003, pp. 84–102.

[24] T. Butz and M. Ehmann. “Virtuelle Rundenzeitoptimierung mittels realistischer Rennfahrzeugsimulation”. In: Tagungsunterlagen race.tech. TÜV Süd Gruppe, 2005.

[25] A. Candelpergher, M. Gadola, and D. Vetturi. “Developments of a method for lap time simulation”. In: SAE Technical Paper. SAE International, 2000, pp. 241–246.

[26] M. Cannon et al. “Robust tubes in nonlinear model predictive control”. In: IEEE Transactions on Automatic Control 56.8, 2011, pp. 1942–1947.

[27] M. Cannon, D. Ng, and B. Kouvaritakis. “Successive linearization NMPC for a class of stochastic nonlinear systems”. In: Nonlinear Model Predictive Control: Towards New Challenging Applications. Springer Berlin Heidelberg, 2009, pp. 249–262.

[28] A. Carvalho et al. “Predictive control of an autonomous ground vehicle using an iterative linearization approach”. In: 16th International IEEE Conference on Intelligent Transportation Systems. 2013, pp. 2335–2340.

[29] D. Casanova, R. Sharp, and P. Symonds. “Minimum time manoeuvring: the significance of yaw inertia”. In: Vehicle System Dynamics 34.2, 2000, pp. 77–115.

[30] P. Casanova. “On minimum time vehicle manoeuvring: the theoretical optimal lap”. PhD thesis. Cranfield University, 2000.

[31] R. de Castro et al. “Design of safety-oriented control allocation strategies for overactuated electric vehicles”. In: Vehicle System Dynamics 52.8, 2014, pp. 1017–1046.

[32] R. de Castro et al. “Minimum-time manoeuvring in electric vehicles with four wheel-individual-motors”. In: Vehicle System Dynamics 52.6, 2014, pp. 824–846.

[33] J. Chen, W. Zhan, and M. Tomizuka. “Constrained iterative LQR for on-road autonomous driving motion planning”. In: IEEE 20th International Conference on Intelligent Transportation Systems. 2017, pp. 1–7.

[34] Z. Chen, Y. Fu, and C. C. Mi. “State of charge estimation of lithium-ion batteries in electric drive vehicles using extended kalman filtering”. In: IEEE Transactions on Vehicular Technology 62.3, 2013, pp. 1020–1030.

[35] F. Christ et al. “Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients”. In: Vehicle System Dynamics, 2019, pp. 1–25.

[36] V. Cibulka, T. Hanis, and M. Hromcik. “Data-driven identification of vehicle dynamics using koopman operator”. In: 2019, arXiv: `1903.06103 [math.OC]`.

[37] V. Cibulka et al. “Model predictive control of a vehicle using koopman operator”. In: IFAC-PapersOnLine 53.2, 2020, 21st IFAC World Congress, pp. 4228–4233.

[38] D. Cole, A. Pick, and A. Odhams. “Predictive and linear quadratic methods for potential application to modelling driver steering control”. In: Vehicle System Dynamics 44.3, 2006, pp. 259–284.

[39] V. Cossalter et al. “A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles”. In: Vehicle System Dynamics 31.2, 1999, pp. 113–135.

[40] J. D’Errico. *Arclength*. `https://www.mathworks.com/matlabcentral/fileexchange/34871-arclength`. Accessed 6. September 2020.

[41] J. D’Errico. *Fminsearchcon*. `https://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon`. Accessed 10. October 2019.

[42] A. d’Aspremont and S. Boyd. *Relaxations and randomized methods for nonconvex QCQPs*. `https://web.stanford.edu/class/ee392o/relaxations.pdf`. Accessed 24. November 2020. 2003.

[43] R. de Castro et al. “Torque allocation in electric vehicles with in-wheel motors: a performance-oriented approach”. In: 52nd IEEE Conference on Decision and Control. 2013, pp. 1538–1543.

[44] R. de Castro et al. “Minimum-time path-following for highly redundant electric vehicles”. In: IEEE Transactions on Control Systems Technology 24.2, 2016, pp. 487–501.

[45] L. Del Re, J. Chapuis, and V. Nevistic. “Predictive control with embedded feedback linearization for bilinear plants with input constraints”. In: Proceedings of 32nd IEEE Conference on Decision and Control. 1993, 2984–2989 vol.4.

[46] G. Di Pillo and L. Grippo. “Exact penalty functions in constrained optimization”. In: SIAM Journal on Control and Optimization 27.6, 1989, pp. 1333–1360.

[47] M. Diehl and S. Gros. *Numerical optimal control*. `https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf`. Accessed 24. November 2020.

[48] M. Diehl, H. G. Bock, and J. P. Schlöder. “A real-time iteration scheme for nonlinear optimization in optimal feedback control”. In: SIAM Journal on Control and Optimization 43.5, 2005, pp. 1714–1736.

[49] M. Diehl, H. J. Ferreau, and N. Haverbeke. “Efficient numerical methods for nonlinear MPC and moving horizon estimation”. In: Nonlinear model predictive control. Springer, 2009, pp. 391–417.

[50] E. W. Dijkstra. “A note on two problems in connexion with graphs”. In: Numerische Mathematik 1.1, 1959, pp. 269–271.

[51] D. Dolgov et al. “Practical search techniques in path planning for autonomous driving”. In: AAAI Workshop - Technical Report, 2008,

[52] I. Dunning, J. Huchette, and M. Lubin. “JuMP: a modeling language for mathematical optimization”. In: SIAM Review 59.2, 2017, pp. 295–320.

[53] S. Ebbesen. “Optimal sizing and control of hybrid electric vehicles”. PhD thesis. ETH Zürich, 2012.

[54] S. Ebbesen et al. “Time-optimal control strategies for a hybrid electric race car”. In: IEEE Transactions on Control Systems Technology 26.1, 2018, pp. 233–247.

[55] M. Eckert. “Energieoptimale Fahrdynamikregelung mehrmotoriger Elektrofahrzeuge”. PhD thesis. Karlsruhe Institute of Technology, 2015.

[56] P. Elbert et al. “Stochastic dynamic programming for the energy management of a serial hybrid electric bus”. In: International Journal of Vehicle Design 69, 2015, pp. 88–112.

[57] P. Falcone et al. “Predictive active steering control for autonomous vehicle systems”. In: IEEE Transactions on Control Systems Technology 15.3, 2007, pp. 566–580.

[58] P. Falugi. “Model predictive control: a passive scheme”. In: IFAC Proceedings Volumes 47.3, 2014, 19th IFAC World Congress, pp. 1017–1022.

[59] P. Falugi, E. Kerrigan, and E. V. Wyk. *Imperial college london optimal control software user guide (ICLOCS)*. `http://www.ee.ic.ac.uk/ICLOCS`.

[60] Y. Fang and A. Armaou. “Nonlinear model predictive control using a bilinear carleman linearization-based formulation for chemical processes”. In: American Control Conference. 2015, pp. 5629–5634.

[61] H. Ferreau et al. “Embedded optimization methods for industrial automatic control”. In: IFAC-PapersOnLine 50.1, 2017, pp. 13194–13209.

[62] R. Fischer et al. “Fahrermodellierung für Fahrdynamik und Verbrauchsberechnungen”. In: Fahrermodellierung in Wissenschaft und Wirtschaft, Fortschritt-Berichte VDI, Reihe 22 Mensch-Maschine-Systeme Nr. 28. 2009, pp. 1–14.

[63] A. Forsgren, P. E. Gill, and M. H. Wright. “Interior methods for nonlinear optimization”. In: SIAM Rev. 44.4, 2002, pp. 525–597.

[64] J. Funke et al. “Up to the limits: autonomous Audi TTS”. In: IEEE Intelligent Vehicles Symposium. 2012, pp. 541–547.

[65] M. Gerdts. “A moving horizon technique for the simulation of automobile test-drives”. In: ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 83.3, 2003, pp. 147–162.

[66] M. Gerdts. “Solving mixed-integer optimal control problems by branch and bound: a case study from automobile test-driving with gear shift”. In: Optimal Control Applications and Methods 26.1, 2005, pp. 1–18.

[67] M. Gerdts et al. “Generating locally optimal trajectories for an automatically driven car”. In: Optimization and Engineering 10, 2009, pp. 439–463.

[68] N. Giorgetti et al. “Hybrid model predictive control application towards optimal semi-active suspension”. In: Proceedings of the IEEE International Symposium on Industrial Electronics. Vol. 1. 2005, pp. 391–398.

[69] R. Goldenthal and M. Bercovier. “Spline curve approximation and design by optimal control over the knots”. In: Geometric Modelling. Springer Vienna, 2004, pp. 53–64.

[70] D. González et al. “A review of motion planning techniques for automated vehicles”. In: IEEE Transactions on Intelligent Transportation Systems 17.4, 2016, pp. 1135–1145.

[71] S. Gros et al. “From linear to nonlinear MPC: bridging the gap via the real-time iteration”. In: International Journal of Control 93.1, 2020, pp. 62–80.

[72] S. Gudmundsson. *General aviation aircraft design - applied methods and procedures*. Butterworth-Heinemann, 2013.

[73] I. Gundlach and U. Konigorski. “Modellbasierte Online-Trajektorienplanung für zeitoptimale Rennlinien”. In: at - Automatisierungstechnik 67.9, 2019, pp. 799–813.

[74] Gurobi Optimization LLC. *Gurobi optimizer reference manual*. `https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.1/refman.pdf`. Accessed 24. November 2020.

[75] S.-P. Han and O. L. Mangasarian. “Exact penalty functions in nonlinear programming”. In: Mathematical programming 17.1, 1979, pp. 251–269.

[76] P. E. Hart, N. J. Nilsson, and B. Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: IEEE Transactions on Systems Science and Cybernetics 4.2, 1968, pp. 100–107.

[77] A. Heilmeier, M. Geisslinger, and J. Betz. “A quasi-steady-state lap time simulation for electrified race cars”. In: 14th International Conference on Ecological Vehicles and Renewable Energies. 2019, pp. 1–10.

[78] A. Heilmeier et al. “Minimum curvature trajectory planning and control for an autonomous race car”. In: Vehicle System Dynamics, 2019, pp. 1–31.

[79] J. R. Hendershot and T. J. E. Miller. *Design of brushless permanent-magnet machines*. Motor Design Books, 2010.

[80] J. Hendrikx, T. Meijlink, and R. Kriens. “Application of optimal control theory to inverse simulation of car handling”. In: Vehicle System Dynamics 26.6, 1996, pp. 449–461.

[81] T. Herrmann et al. “Energy management strategy for an autonomous electric racecar using optimal control”. In: IEEE Intelligent Transportation Systems Conference. 2019, pp. 720–725.

[82] T. Herrmann et al. *Minimum race-time planning-strategy for an autonomous electric racecar*. 2020. arXiv: `2005.07127 [eess.SY]`.

[83] D. Hsu, J. Latombe, and R. Motwani. “Path planning in expansive configuration spaces”. In: Proceedings of International Conference on Robotics and Automation. Vol. 3. 1997, pp. 2719–2726.

[84] J. h. Jeon, S. Karaman, and E. Frazzoli. “Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*”. In: 50th IEEE Conference on Decision and Control and European Control Conference. 2011, pp. 3276–3282.

[85] J. h. Jeon et al. “Optimal motion planning with the half-car dynamical model for autonomous high-speed driving”. In: American Control Conference. 2013, pp. 188–193.

[86] E. John and E. A. Yıldırım. “Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension”. In: Computational Optimization and Applications 41.2, 2008, pp. 151–183.

[87] H. Kang et al. “Knot calculation for spline fitting via sparse optimization”. In: Computer-Aided Design 58, 2015, pp. 179–188.

[88] N. R. Kapania and J. C. Gerdes. “Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control”. In: American Control Conference. 2015, pp. 2753–2758.

[89] N. Kapania, J. Subosits, and J. Gerdes. “A sequential two-step algorithm for fast generation of vehicle racing trajectories”. In: Journal of Dynamic Systems, Measurement, and Control 138, 2016,

[90] N. R. Kapania and J. C. Gerdes. "Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling". In: Vehicle System Dynamics 53.12, 2015, pp. 1687–1704.

[91] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: Proceedings of the 16th Annual ACM Symposium on Theory of Computing. Association for Computing Machinery, 1984, pp. 302–311.

[92] C. Katrakazas et al. "Real-time motion planning methods for autonomous on-road driving: state-of-the-art and future research directions". In: Transportation Research Part C: Emerging Technologies 60, 2015, pp. 416–442.

[93] S. D. Keen and D. J. Cole. "Application of time-variant predictive control to modelling driver steering skill". In: Vehicle System Dynamics 49.4, 2011, pp. 527–559.

[94] S. D. Keen and D. J. Cole. "Steering control using model predictive control and multiple internal models". In: 8th International Symposium on Advanced Vehicle Control. 2006.

[95] D. Kelly. "Lap time simulation with transient vehicle and tyre dynamics". PhD thesis. Cranfield University, 2008.

[96] D. Kelly and R. Sharp. "Time-optimal control of the race car: influence of a thermodynamic tyre model". In: Vehicle System Dynamics 50.4, 2012, pp. 641–662.

[97] M. Kelly. "An introduction to trajectory optimization: how to do your own direct collocation". In: SIAM Review 59.4, 2017, pp. 849–904.

[98] H. K. Khalil. *Nonlinear systems*. Prentice Hall, 2002.

[99] J. Kim, J. Oh, and H. Lee. "Review on battery thermal management system for electric vehicles". In: Applied Thermal Engineering, 2019, pp. 192–212.

[100] C. Kirches. *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Springer Science and Business Media, 2011.

[101] C. Kirches et al. "Time-optimal control of automobile test drives with gear shifts". In: Optimal Control Applications and Methods 31.2, 2010, pp. 137–153.

[102] S. Kirchhoff. "Ueber den Durchgang eines elektrischen Stromes durch eine Ebene, insbesondere durch eine kreisförmige". In: Annalen der Physik 140.4, 1845, pp. 497–514.

[103] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for optimization*. MIT Press, 2019.

[104] M. Korda and I. Mezić. "Linear predictors for nonlinear dynamical systems: koopman operator meets model predictive control". In: Automatica 93, 2018, pp. 149–160.

[105] S. van Koutrik. "Optimal control for race car minimum time maneuvering". `http://resolver.tudelft.nl/uuid:c3a0dfec-530e-4672-b4cd-55374817e686`; Accessed 6. September 2020. MA thesis. Technical University of Delft, 2015.

[106] B. Kouvaritakis and M. Cannon. *Model predictive control - classical, robust and stochastic*. Springer, 2015.

[107] M. Kozlov, S. Tarasov, and L. Khachiyan. "The polynomial solvability of convex quadratic programming". In: USSR Computational Mathematics and Mathematical Physics 20.5, 1980, pp. 223–228.

[108] K. Kritayakirana and J. Gerdes. "Autonomous vehicle control at the limits of handling". In: International Journal of Vehicle Autonomous Systems 10, 2012, pp. 271–296. DOI: 10.1504/IJVAS.2012.051270.

[109] K. Kritayakirana and J. C. Gerdes. "Autonomous cornering at the limits: maximizing a "g-g" diagram by using feedforward trail-braking and throttle-on-exit". In: IFAC Proceedings Volumes 43.7, 2010, 6th IFAC Symposium on Advances in Automotive Control, pp. 548–553.

[110] J. Kronqvist et al. "A review and comparison of solvers for convex MINLP". In: Optimization and Engineering 20.2, 2019, pp. 397–455.

[111] M. J. Kurtz and M. A. Henson. "Input-output linearizing control of constrained nonlinear processes". In: Journal of Process Control 7.1, 1997, pp. 3–17.

[112] S. M. Lavalle. *Rapidly-exploring random trees: a new tool for path planning*. Tech. rep. Computer Science Department, Iowa State University, 1998.

[113] Y. Li, Z. Littlefield, and K. E. Bekris. "Sparse methods for efficient asymptotically optimal kinodynamic planning". In: Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics. Springer International Publishing, 2015, pp. 263–282.

[114] L. Liberti. *Introduction to global optimization*. https://www.lix.polytechnique.fr/~liberti/teaching/globalopt-lima.pdf. Accessed 23. January 2021. 2008.

[115] L. Liberti and N. Maculan. *Global optimization - from theory to implementation*. Springer Science and Business Media, 2006.

[116] D. J. N. Limebeer and G. Perantoni. "Optimal control of a formula one car on a three-dimensional track - Part 2: optimal control". In: Journal of Dynamic Systems, Measurement, and Control 137.5, 2015,

[117] D. J. N. Limebeer and A. V. Rao. "Faster, higher, and greener: vehicular optimal control". In: IEEE Control Systems Magazine 35.2, 2015, pp. 36–56.

[118] D. Limebeer, G. Perantoni, and A. Rao. "Optimal control of formula one car energy recovery systems". In: International Journal of Control 87.10, 2014, pp. 2065–2080.

[119] A. Liniger, A. Domahidi, and M. Morari. "Optimization-based autonomous racing of 1:43 scale RC cars". In: Optimal Control Applications and Methods 36, 2015, pp. 628–647.

[120] T. Lipp and S. Boyd. "Minimum-time speed optimisation over a fixed path". In: International Journal of Control 87.6, 2014, pp. 1297–1311.

[121] T. Lipp and S. Boyd. “Variations and extension of the convex–concave procedure”. In: Optimization and Engineering 17, 2016, pp. 263–287.

[122] J. Löfberg. “Oops! I cannot do it again: testing for recursive feasibility in MPC”. In: Automatica 48.3, 2012, pp. 550–555.

[123] R. Lot, M. Massaro, and R. Sartori. “Advanced motorcycle virtual rider”. In: Vehicle System Dynamics 46.sup1, 2008, pp. 215–224.

[124] R. Lot and N. Bianco. “The significance of high-order dynamics in lap time simulations”. In: 24th International Symposium on Dynamics of Vehicles on Roads and Tracks. 2015.

[125] R. Lot and S. Evangelou. “Lap time optimization of a sports series hybrid electric vehicle”. In: Proceedings of the World Congress on Engineering 2013. 2013.

[126] S. Lovato and M. Massaro. “A three-dimensional free-trajectory quasi-steady-state optimal-control method for minimum-lap-time of race vehicles”. In: Vehicle System Dynamics, 2021, pp. 1–19.

[127] S. Lupberger et al. “Control allocation concept for hybrid braking considering dynamic battery behaviour”. In: IFAC World Congress 21, 2020,

[128] A. Mahmoudi et al. “Efficiency maps of electrical machines”. In: IEEE Energy Conversion Congress and Exposition. 2015, pp. 2791–2799.

[129] A. Mahmoudi et al. “Loss function modeling of efficiency maps of electrical machines”. In: IEEE Transactions on Industry Applications 53.5, 2017, pp. 4221–4231.

[130] O. L. Mangasarian. “A hybrid algorithm for solving the absolute value equation”. In: Optimization Letters 9, 2015, pp. 1469–1474.

[131] M. Maniowski. “Optimisation of driver actions in RWD race car including tyre thermodynamics”. In: Vehicle System Dynamics 54.4, 2016, pp. 526–544.

[132] M. Maniowski. “Optimization of driver and chassis of FWD racing car for faster cornering”. In: The Dynamics of Vehicles on Roads and Tracks. 2016, pp. 77–86.

[133] Y. Mao, M. Szmuk, and B. Açıkmeşe. “Successive convexification of non-convex optimal control problems and its convergence properties”. In: IEEE 55th Conference on Decision and Control. 2016, pp. 3636–3641.

[134] Y. Mao et al. “Successive convexification of non-convex optimal control problems with state constraints”. In: IFAC-PapersOnLine 50.1, 2017, 20th IFAC World Congress, pp. 4063–4069.

[135] N. Maratos. “Exact penalty function algorithms for finite dimensional and control optimization problems”. PhD thesis. Imperial College London, 1978.

[136] M. I. Masouleh. “Optimal control and stability of four-wheeled vehicles”. PhD thesis. University of Oxford, 2017.

[137] M. I. Masouleh and D. J. N. Limebeer. “Optimizing the aero-suspension interactions in a formula one car”. In: IEEE Transactions on Control Systems Technology 24.3, 2016, pp. 912–927.

[138] *MATLAB mathematics documentations*. `https://de.mathworks.com/help/pdf_doc/matlab/math.pdf`. Accessed 11. November 2019. 2019.

[139] D. Mayne et al. “Constrained model predictive control: stability and optimality”. In: Automatica 36.6, 2000, pp. 789–814.

[140] D. Q. Mayne and N. Maratos. “A first order, exact penalty function algorithm for equality constrained optimization problems”. In: Mathematical Programming 16.1, 1979, pp. 303–324.

[141] D. Q. Mayne. “Differential dynamic programming–a unified approach to the optimization of dynamic systems”. In: Control and Dynamic Systems. Vol. 10. Academic Press, 1973, pp. 179–254.

[142] W. F. Milliken and D. L. Milliken. *Race car vehicle dynamics*. SAE International, 1995.

[143] S. Mollov et al. “MIMO predictive control by multiple-step linearization of takagi-sugeno fuzzy models”. In: IFAC Proceedings Volumes 31.29, 1998, 7th IFAC Symposium on Artificial Intelligence in Real Time Control, pp. 197–202.

[144] S. Mollov et al. “Effective optimization for fuzzy model predictive control”. In: IEEE Transactions on Fuzzy Systems 12.5, 2004, pp. 661–675.

[145] M. Mühlmeier and N. Müller. “Optimization of the driving line on a race track”. In: SAE Technical Paper. SAE International, 2002.

[146] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[147] M. Neunert et al. “Fast nonlinear model predictive control for unified trajectory optimization and tracking”. In: IEEE International Conference on Robotics and Automation. 2016, pp. 1398–1404.

[148] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science and Business Media, 2006.

[149] H. Pacejka. *Tyre and vehicle dynamics*. Butterworth-Heinemann, 2006.

[150] B. Paden et al. *A survey of motion planning and control techniques for self-driving urban vehicles*. 2016. arXiv: `1604.07446 [cs.RO]`.

[151] A. Pakniyat and P.E.Caines. “Time optimal hybrid minimum principle and the gear changing problem for electric vehicles”. In: IFAC-PapersOnLine 48.27, 2015, Analysis and Design of Hybrid Systems, pp. 187–192.

[152] M. Papageorgiou, M. Leibold, and M. Buss. *Optimierung - Statische, dynamische, stochastische Verfahren für die Anwendung*. Springer-Verlag, 2015.

[153] J. Park and S. Boyd. *General heuristics for nonconvex quadratically constrained quadratic programming*. 2017. arXiv: `1703.07870 [math.OC]`.

[154] B. Passenberg. “Theory and algorithms for indirect methods in optimal control of hybrid systems”. PhD thesis. Technical University of Munich, 2012.

[155] M. A. Patterson and A. V. Rao. “GPOPS-II: a MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming”. In: ACM Transactions on Mathematical Software 41.1, 2014,

[156] C. Patton. “Development of vehicle dynamics tools for motorsports”. PhD thesis. Oregon State University, 2013.

[157] G. Perantoni and D. J. N. Limebeer. “Optimal control of a formula one car on a three-dimensional track - Part 1: track modeling and identification”. In: Journal of Dynamic Systems, Measurement, and Control 137.5, 2015,

[158] G. Perantoni and D. J. Limebeer. “Optimal control for a formula one car with variable parameters”. In: Vehicle System Dynamics 52.5, 2014, pp. 653–678.

[159] P. Pillay and R. Krishnan. “Modeling, simulation, and analysis of permanent-magnet motor drives. i. the permanent-magnet synchronous motor drive”. In: IEEE Transactions on Industry Applications 25.2, 1989, pp. 265–273.

[160] L. Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[161] W. B. Powell. *Approximate dynamic programming - solving the curses of dimensionality*. John Wiley and Sons, 2007.

[162] W. B. Powell. “What you should know about approximate dynamic programming”. In: Naval Research Logistics 56.3, 2009, pp. 239–249.

[163] G. Prokop. “Modeling human vehicle driving by model predictive online optimization”. In: Vehicle System Dynamics 35.1, 2001, pp. 19–53.

[164] T. Raff, C. Ebenbauer, and F. Allgöwer. “Nonlinear model predictive control: a passivity-based approach”. In: Assessment and Future Directions of Nonlinear Model Predictive Control. Springer Berlin Heidelberg, 2007, pp. 151–162.

[165] S. V. Raković and W. S. Levine. *Handbook of model predictive control*. Springer, 2018.

[166] A. V. Rao. “A survey of numerical methods for optimal control”. In: Astrodynamics 2009: proceedings of the AAS/AIAA Astrodynamics Specialist Conference. 2010.

[167] C. V. Rao, S. J. Wright, and J. B. Rawlings. “Application of interior-point methods to model predictive control”. In: Journal of Optimization Theory and Applications 99, 1998, pp. 723–757.

[168] E. Reichensdörfer, D. Odenthal, and D. Wollherr. “On the stability of nonlinear wheel-slip zero dynamics in traction control systems”. In: IEEE Transactions on Control Systems Technology 28.2, 2020, pp. 489–504.

[169] J. Revels, M. Lubin, and T. Papamarkou. “Forward-mode automatic differentiation in Julia”. In: CoRR abs/1607.07892, 2016, URL: `http://arxiv.org/abs/1607.07892`.

[170] R. S. Rice. *Measuring car-driver interaction with the g-g diagram*. Society of Automotive Engineers, 1973.

[171] T. Rizano et al. "Global path planning for competitive robotic cars". In: 52nd IEEE Conference on Decision and Control. 2013, pp. 4510–4516.

[172] N. Robuschi et al. "Multiphase mixed-integer nonlinear optimal control of hybrid electric vehicles". In: optimization-online, 2019, URL: `http://www.optimization-online.org/DB_HTML/2019/05/7223.html`.

[173] M. do Rosario de Pinho, I. L. Domini, and G. N. Silva. "On free time optimal control problems". In: Proceedings of the 9th Brazilian Conference on Dynamics Control and their Applications. 2010, pp. 41–50.

[174] H. Ryoo and N. Sahinidis. "Global optimization of nonconvex NLPs and MINLPs with applications in process design". In: Computers and Chemical Engineering 19.5, 1995, pp. 551–566.

[175] Z. M. Salameh, M. A. Casacca, and W. A. Lynch. "A mathematical model for lead-acid batteries". In: IEEE Transactions on Energy Conversion 7.1, 1992, pp. 93–98.

[176] M. Salazar et al. "Real-time control algorithms for a hybrid electric race car using a two-level model predictive control scheme". In: IEEE Transactions on Vehicular Technology 66.12, 2017, pp. 10911–10922.

[177] M. Salazar et al. "Time-optimal control policy for a hybrid electric race car". In: IEEE Transactions on Control Systems Technology 25.6, 2017, pp. 1921–1934.

[178] S. M. Savaresi and M. Tanelli. *Active braking control systems design for vehicles*. Springer Science and Business Media, 2010.

[179] S. M. Savaresi et al. *Semi-active suspension control design for vehicles*. Elsevier, 2010.

[180] F. Schnelle and P. Eberhard. "Constraint mapping in a feedback linearization/MPC scheme for trajectory tracking of underactuated multibody systems". In: IFAC-PapersOnLine 48.23, 2015, 5th IFAC Conference on Nonlinear Model Predictive Control NMPC, pp. 446–451.

[181] D. Schramm, M. Hiller, and R. Bardini. *Vehicle dynamics - modeling and simulation*. Springer, 2014.

[182] J. Schützhold and W. Hofmann. "Analysis of the temperature dependence of losses in electrical machines". In: IEEE Energy Conversion Congress and Exposition. 2013, pp. 3159–3165.

[187] H. Seki, S. Ooyama, and M. Ogawa. "Nonlinear model predictive control using successive linearization". In: Transactions of the Society of Instrument and Control Engineers E-3.1, 2004, pp. 66–72.

[188] R. S. Sharp. "Application of optimal preview control to speed-tracking of road vehicles". In: Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 221.12, 2007, pp. 1571–1578.

[189] R. Sharp, D. Casanova, and P. Symonds. “A mathematical model for driver steering control, with design, tuning and performance results”. In: Vehicle System Dynamics 33.5, 2000, pp. 289–326.

[190] B. Siegler, A. Deakin, and D. Crolla. “Lap time simulation: comparison of steady state, quasi-static and transient racing car cornering strategies”. In: SAE Transactions 109, 2000, pp. 2575–2581.

[191] D. Simon, J. Löfberg, and T. Glad. “Nonlinear model predictive control using feedback linearization and local inner convex constraint approximations”. In: European Control Conference. 2013, pp. 2056–2061.

[192] H. T. Smakman. “Functional integration of slip control with active suspension for improved lateral vehicle dynamics”. PhD thesis. Technical University of Delft, 2000.

[193] E. Smith and C. Pantelides. “A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs”. In: Computers and Chemical Engineering 23.4, 1999, pp. 457–478.

[194] T. Stahl et al. “Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios”. In: IEEE Intelligent Transportation Systems Conference. 2019, pp. 3149–3154.

[195] A. Stentz. “Optimal and efficient path planning for partially-known environments”. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation. Vol. 4. 1994, pp. 3310–3317.

[196] O. von Stryk and R. Bulirsch. “Direct and indirect methods for trajectory optimization”. In: Annals of Operations Research 37.1–4, 1992, pp. 357–373.

[197] P. Suchomski. “Method of optimal variable-knot spline interpolation in the $\mathcal{L}_2$ discrete norm”. In: International Journal of Systems Science 22.11, 1991, pp. 2263–2274.

[198] O. Sundström, L. Guzzella, and P. Soltic. “Optimal hybridization in two parallel hybrid electric vehicles using dynamic programming”. In: IFAC Proceedings Volumes 41.2, 2008, 17th IFAC World Congress, pp. 4642–4647.

[199] L. Svensson et al. *Adaptive trajectory planning and optimization at limits of handling*. 2019. arXiv: `1903.04240 [cs.RO]`.

[200] K. Tanaka and H. O. Wang. *Fuzzy control systems design and analysis - a linear matrix inequality approach*. John Wiley and Sons, 2004.

[201] Y. Tassa, T. Erez, and E. Todorov. “Synthesis and stabilization of complex behaviors through online trajectory optimization”. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012, pp. 4906–4913.

[202] Y. Tassa, N. Mansard, and E. Todorov. “Control-limited differential dynamic programming”. In: IEEE International Conference on Robotics and Automation. 2014, pp. 1168–1175.

[203] D. Tavernini et al. “Minimum time cornering: the effect of road surface and car transmission layout”. In: Vehicle System Dynamics 51.10, 2013, pp. 1533–1547.

[204] M. Tawarmalani and N. V. Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming - theory, algorithms, software, and applications*. Springer Science and Business Media, 2002.

[205] P. Theodosis and J. Gerdes. “Generating a racing line for an autonomous racecar using professional driving techniques”. In: Proceedings of the ASME 2011 Dynamic Systems and Control Conference. Vol. 2. 2011.

[206] M. Thommyppillai, S. Evangelou, and R. Sharp. “Car driving at the limit by adaptive linear optimal preview control”. In: Vehicle System Dynamics 47.12, 2009, pp. 1535–1550.

[207] J. P. Timings and D. J. Cole. “Efficient minimum manoeuvre time optimisation of an oversteering vehicle at constant forward speed”. In: Proceedings of the 2011 American Control Conference. 2011, pp. 5267–5272.

[208] J. Timings and D. Cole. “Minimum manoeuvre time of a nonlinear vehicle at constant forward speed using convex optimisation”. In: Proceedings of the 10th International Symposium on Advanced Vehicle Control. 2010, pp. 21–26.

[209] J. Timings and D. Cole. “Vehicle trajectory linearisation to enable efficient optimisation of the constant speed racing line”. In: Vehicle System Dynamics 50.6, 2012, pp. 883–901.

[210] J. Timings and D. Cole. “Robust lap-time simulation”. In: Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 228.10, 2014, pp. 1200–1216.

[211] J. P. Timings and D. J. Cole. “Minimum maneuver time calculation using convex optimization”. In: Journal of Dynamic Systems, Measurement, and Control 135.3, 2013,

[212] A. J. Tremlett and D. J. N. Limebeer. “Optimal tyre usage for a formula one car”. In: Vehicle System Dynamics 54.10, 2016, pp. 1448–1473.

[213] A. Tremlett et al. “Optimal control of motorsport differentials”. In: Vehicle System Dynamics 53.12, 2015, pp. 1772–1794.

[214] C. Urmson et al. “Autonomous driving in urban environments: boss and the urban challenge”. In: Journal of Field Robotics 25.8, 2008, pp. 425–466.

[215] W. Van Loock et al. “A convex optimization approach to curve fitting with b-splines”. In: IFAC Proceedings Volumes 44.1, 2011, pp. 2290–2295.

[216] L. Vandenberghe and S. Boyd. “Semidefinite programming”. In: SIAM Review 38.1, 1996, pp. 49–95.

[217] M. Veneri and M. Massaro. “A free-trajectory quasi-steady-state optimal-control method for minimum lap-time of race vehicles”. In: Vehicle System Dynamics 58.6, 2020, pp. 933–954.

[218] D. Verscheure et al. “Time-optimal path tracking for robots: a convex optimization approach”. In: IEEE Transactions on Automatic Control 54.10, 2009, pp. 2318–2327.

[219] R. Verschueren. “Convex approximation methods for nonlinear model predictive control”. PhD thesis. University of Freiburg, 2018.

[220] R. Verschueren et al. “Towards time-optimal race car driving using nonlinear MPC in real-time”. In: 53rd IEEE Conference on Decision and Control. 2014, pp. 2505–2510.

[221] R. Verschueren. “Design and implementation of a time-optimal controller for model race cars”. MA thesis. Catholic University of Leuven, 2014.

[222] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: Math. Program. 106.1, 2006, pp. 25–57.

[223] P. Waldmann. *Entwicklung eines Fahrzeugführungssystems zum Erlernen der Ideallinie auf Rennstrecken*. Shaker, 2009.

[224] X. Wang et al. “Application study on the dynamic programming algorithm for energy management of plug-in hybrid electric vehicles”. In: Energies 8.4, 2015, pp. 3225–3244.

[225] M. Werling. “Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien”. PhD thesis. Karlsruher Institut für Technologie, 2010.

[226] M. Werling et al. “Optimal trajectories for time-critical street scenarios using discretized terminal manifolds”. In: The International Journal of Robotics Research 31.3, 2012, pp. 346–359.

[227] W. J. West and D. J. N. Limebeer. “Optimal tyre management for a high-performance race car”. In: Vehicle System Dynamics, 2020, pp. 1–19.

[228] P. Wolfe. “The simplex method for quadratic programming”. In: Econometrica 27.3, 1959, pp. 382–398.

[229] Z. Xie, C. K. Liu, and K. Hauser. “Differential dynamic programming with nonlinear constraints”. In: IEEE International Conference on Robotics and Automation. 2017, pp. 695–702.

[230] J. J. Ye. “Nondifferentiable multiplier rules for optimization and bilevel optimization problems”. In: SIAM Journal on Optimization 15.1, 2004, pp. 252–274.

[231] J. Ziegler and C. Stiller. “Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios”. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. 2009, pp. 1879–1884.

[232] V. A. Zorich. *Mathematical analysis I*. Springer Berlin Heidelberg, 2016.

Own Publications

[183] T. Sedlacek, D. Odenthal, and D. Wollherr. “Minimum-time optimal control for battery electric vehicles with four wheel-independent drives considering electrical overloading”. In: Vehicle System Dynamics 60.2, 2022, pp. 491–515.

[184] T. Sedlacek, D. Odenthal, and D. Wollherr. “Convexification of semi-activity constraints applied to minimum-time optimal control for vehicles with semi-active limited-slip differential”. In: Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics. SciTePress, 2020, pp. 15–25.

[185] T. Sedlacek, D. Odenthal, and D. Wollherr. “Minimum-time optimal control for vehicles with active rear-axle steering, transfer case and variable parameters”. In: Vehicle System Dynamics 59.8, 2021, pp. 1227–1255.

[186] T. Sedlacek, D. Odenthal, and D. Wollherr. “Space splitting convexification: a local solution method for nonconvex optimal control problems”. In: International Journal of Control, 2021, pp. 1–24.

A Modelling Addendum

A.1 Transformation of Model Equations into Moving Coordinate Frame

When integrating differential equations for values in a moving coordinate frame, an additional term must be considered for correct results. The derivation of this term is treated in this section. Its is assumed that the COG position of a body with mass m is given by the position vector $\mathbf{p}^I = [x^I, y^I]^T$ in the inertial coordinate frame. The orientation of the body is described using the yaw angle ψ, which depicts the angle between the body coordinate frame and the inertial frame. The acceleration of the COG is given by

$$\ddot{\mathbf{p}}^I = \frac{1}{m}\mathbf{R}_z(-\psi)\mathbf{F}^B \quad \Rightarrow \quad \mathbf{R}_z(\psi)\ddot{\mathbf{p}}^I = \frac{1}{m}\mathbf{F}^B. \tag{A.1a}$$

using the forces $\mathbf{F}^B$ in the body coordinate frame acting on it. Inspecting the left-hand side of the right equation in (A.1a) results in

$$\begin{aligned}\mathbf{R}_z(\psi)\ddot{\mathbf{p}}^I &= \mathbf{R}_z(\psi)\frac{d}{dt}\Big(\mathbf{R}_z(-\psi)\dot{\mathbf{p}}^B\Big) = \mathbf{R}_z(\psi)\left(\frac{\partial \mathbf{R}_z(-\psi)}{\partial \psi}\dot{\psi}\dot{\mathbf{p}}^B + \mathbf{R}_z(-\psi)\ddot{\mathbf{p}}^B\right)\\ &= \underbrace{\mathbf{R}_z(\psi)\frac{\partial \mathbf{R}_z(-\psi)}{\partial \psi}\dot{\psi}}_{:=\tilde{\mathbf{R}}}\dot{\mathbf{p}}^B + \ddot{\mathbf{p}}^B. \end{aligned}\tag{A.1b}$$

Inserting (A.1b) into (A.1a) leads to

$$\ddot{\mathbf{p}}^B = \frac{1}{m}\mathbf{F}^B - \underbrace{\tilde{\mathbf{R}}\dot{\mathbf{p}}^B}_{:=\mathbf{c}_{add}} \quad \text{with} \tag{A.1c}$$

$$\tilde{\mathbf{R}} = \begin{bmatrix}\cos(\psi) & \sin(\psi)\\ -\sin(\psi) & \cos(\psi)\end{bmatrix}\begin{bmatrix}-\sin(\psi) & -\cos(\psi)\\ \cos(\psi) & -\sin(\psi)\end{bmatrix}\dot{\psi} = \begin{bmatrix}0 & -\dot{\psi}\\ \dot{\psi} & 0\end{bmatrix} \quad \text{and} \tag{A.1d}$$

$$\mathbf{c}_{add} = \begin{bmatrix}0 & -\dot{\psi}\\ \dot{\psi} & 0\end{bmatrix}\begin{bmatrix}v_x\\ v_y\end{bmatrix} = \begin{bmatrix}-\dot{\psi}v_y\\ \dot{\psi}v_x\end{bmatrix} \tag{A.1e}$$

with v_x and v_y representing the longitudinal and lateral velocity in the body coordinate frame.

A.2 Translational Velocities at Wheel Centre Points

The computation of the translational velocities at the wheel centre points given in the body coordinate frame requires considering the yaw rate of the vehicle. Assuming only planar movements, these quantities are given by

$$\mathbf{v}_{P_1}^B = \begin{bmatrix} v_{x,1} \\ v_{y,1} \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} l_f \\ \frac{b_f}{2} \\ -h_v + r_1 \end{bmatrix} = \begin{bmatrix} v_x - \dot{\psi}\,\frac{b_f}{2} \\ v_y + \dot{\psi}\,l_f \\ 0 \end{bmatrix} \tag{A.2a}$$

$$\mathbf{v}_{P_2}^B = \begin{bmatrix} v_{x,2} \\ v_{y,2} \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} l_f \\ -\frac{b_f}{2} \\ -h_v + r_2 \end{bmatrix} = \begin{bmatrix} v_x + \dot{\psi}\,\frac{b_f}{2} \\ v_y + \dot{\psi}\,l_f \\ 0 \end{bmatrix} \tag{A.2b}$$

$$\mathbf{v}_{P_3}^B = \begin{bmatrix} v_{x,3} \\ v_{y,3} \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} -l_r \\ \frac{b_r}{2} \\ -h_v + r_3 \end{bmatrix} = \begin{bmatrix} v_x - \dot{\psi}\,\frac{b_r}{2} \\ v_y - \dot{\psi}\,l_r \\ 0 \end{bmatrix} \tag{A.2c}$$

$$\mathbf{v}_{P_4}^B = \begin{bmatrix} v_{x,4} \\ v_{y,4} \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} -l_r \\ -\frac{b_r}{2} \\ -h_v + r_4 \end{bmatrix} = \begin{bmatrix} v_x + \dot{\psi}\,\frac{b_r}{2} \\ v_y - \dot{\psi}\,l_r \\ 0 \end{bmatrix}. \tag{A.2d}$$

A.3 Derivation of Frenet-Equations

The Frenet-Equations (3.6) can be derived by analysing the kinematic relations using Fig. 3.2 as graphical support. The velocity of the vehicle in the coordinate system of the reference curve[11] is given by

$$\mathbf{v}_{P_0}^{\mathcal{R}} = \mathbf{R}_z(-\Theta)\mathbf{v}_{P_0}^B = \mathbf{R}_z(-\Theta) \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \cos(\Theta) - v_y \sin(\Theta) \\ v_x \sin(\Theta) + v_y \cos(\Theta) \\ 0 \end{bmatrix}. \tag{A.3}$$

This velocity vector can be put together by

$$\mathbf{v}_{P_0}^{\mathcal{R}} = \mathbf{v}_{P_I P_S}^{\mathcal{R}} + \mathbf{v}_{P_S P_0}^{\mathcal{R}} = \begin{bmatrix} \dot{s}_{\mathcal{R}} \\ 0 \\ 0 \end{bmatrix} + \mathbf{v}_{P_S P_0}^{\mathcal{R}}. \tag{A.4a}$$

Using the chain rule

$$\dot{\Theta}_{\mathcal{R}} = \frac{\partial \Theta_{\mathcal{R}}}{\partial s_{\mathcal{R}}} \frac{ds_{\mathcal{R}}}{dt} = \kappa_{\mathcal{R}} \dot{s}_{\mathcal{R}} \tag{A.4b}$$

[11] The current point on the reference curve P_S represents the origin. The tangent and normal in P_S represent the abscissa and ordinate, respectively. Thus, the reference frame is rotated by the angle $-\Theta$ around the z-axis compared to the body frame.

when considering the rotation of the reference frame during differentiation yields

$$\begin{aligned}\mathbf{v}^{\mathcal{R}}_{P_S P_0} &= \dot{\mathbf{r}}^{\mathcal{R}}_{P_S P_0} + \boldsymbol{\omega}_{\mathcal{R}} \times \mathbf{r}^{\mathcal{R}}_{P_S P_0} = \\ &= \begin{bmatrix} 0 \\ \dot{d}_{\mathcal{R}} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\Theta}_{\mathcal{R}} \end{bmatrix} \times \begin{bmatrix} 0 \\ d_{\mathcal{R}} \\ 0 \end{bmatrix} = \begin{bmatrix} -d_{\mathcal{R}} \dot{\Theta}_{\mathcal{R}} \\ \dot{d}_{\mathcal{R}} \\ 0 \end{bmatrix} = \begin{bmatrix} -d_{\mathcal{R}} \kappa_{\mathcal{R}} \dot{s}_{\mathcal{R}} \\ \dot{d}_{\mathcal{R}} \\ 0 \end{bmatrix}. \end{aligned} \tag{A.4c}$$

Inserting (A.4c) into (A.4a) results in the total speed vector

$$\mathbf{v}^{\mathcal{R}}_{P_0} = \begin{bmatrix} \dot{s}_{\mathcal{R}}(1 - d_{\mathcal{R}} \kappa_{\mathcal{R}}) \\ \dot{d}_{\mathcal{R}} \\ 0 \end{bmatrix}. \tag{A.4d}$$

From the comparison of (A.3) and (A.4d) follow the Frenet-Equations (3.6a) and (3.6b). Differentiating $\Theta = \psi - \Theta_{\mathcal{R}}$ using (A.4b) yields (3.6c).

B Mathematics

B.1 Differentiation

It is assumed that $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ represents a scalar-valued function with $\mathbf{x} \in \mathbb{R}^n$. Furthermore, two vector-valued functions are given by $\mathbf{g}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{h}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$. All functions are assumed to be at least twice continuously differentiable. Moreover, the constant vector $\mathbf{a} \in \mathbb{R}^n$ and constant matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ as well as constant symmetric matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ are presumed.

B.1.1 Derivatives of Functions

The first-order derivative of $f(\mathbf{x})$ is given by

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \dots & \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{1 \times n}, \tag{B.1a}$$

the gradient of $f(\mathbf{x})$ by

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x})^T = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{n \times 1} \tag{B.1b}$$

and the Hessian of $f(\mathbf{x})$ by the symmetric matrix

$$\nabla^2_{\mathbf{xx}} f(\mathbf{x}) = f_{\mathbf{xx}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \in \mathbb{R}^{n \times n}. \tag{B.1c}$$

The first-order derivative of $\mathbf{g}(\mathbf{x})$, also called Jacobian, is given by the generally asymmetric matrix

$$\mathbf{g}_{\mathbf{x}}(\mathbf{x}) = \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x})^T = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}. \tag{B.1d}$$

B.1.2 Basic Rules for Differentiation of Vectors and Matrices

The derivative of the product of a scalar-valued function $f(\mathbf{x})$ and a vector-valued function $\mathbf{g}(\mathbf{x})$ is given by

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}} \Big(f(\mathbf{x})\, \mathbf{g}(\mathbf{x}) \Big) = \mathbf{g}(\mathbf{x})\, f_{\mathbf{x}}(\mathbf{x}) + f(\mathbf{x})\, \mathbf{g}_{\mathbf{x}}(\mathbf{x}) \in \mathbb{R}^{m \times n}. \tag{B.2a}$$

For the product of two vector-valued functions $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$, the derivative is computed via

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}\Big(\mathbf{g}(\mathbf{x})^T\,\mathbf{h}(\mathbf{x})\Big) = \mathbf{g}(\mathbf{x})^T\,\mathbf{h}_\mathbf{x}(\mathbf{x}) + \mathbf{h}(\mathbf{x})^T\,\mathbf{g}_\mathbf{x}(\mathbf{x}) \in \mathbb{R}^{1\times n}. \tag{B.2b}$$

For the linear and quadratic case, following basic rules apply:

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A} \tag{B.2c}$$

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}(\mathbf{a}^T\mathbf{x}) = \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}(\mathbf{x}^T\mathbf{a}) = \mathbf{a}^T \tag{B.2d}$$

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}(\mathbf{x}^T\mathbf{Q}\mathbf{x}) = \mathbf{x}^T(\mathbf{Q}+\mathbf{Q}^T) \tag{B.2e}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}\mathbf{x}^2}(\mathbf{x}^T\mathbf{Q}\mathbf{x}) = \mathbf{Q}+\mathbf{Q}^T. \tag{B.2f}$$

B.1.3 Taylor Series Expansion

For the approximation of functions using Taylor's theorem, the point $\hat{\mathbf{x}}$ marks the centre of the Taylor series expansion and the deviation from this point is denoted by $\delta\mathbf{x} = \mathbf{x}-\hat{\mathbf{x}}$. For small $\delta\mathbf{x}$, and thus small $\|\delta\mathbf{x}\|$, the scalar-valued function $f(\mathbf{x})$ can be approximated by a quadratic function with a third order error term:

$$f(\mathbf{x}) = f(\hat{\mathbf{x}}) + f_\mathbf{x}(\hat{\mathbf{x}})\,\delta\mathbf{x} + \frac{1}{2}\,\delta\mathbf{x}^T\,f_{\mathbf{x}\mathbf{x}}(\hat{\mathbf{x}})\,\delta\mathbf{x} + \varepsilon(\|\delta\mathbf{x}\|^3). \tag{B.3a}$$

For the individual components of the vector-valued function $\mathbf{g}(\mathbf{x})$ similarly holds

$$g_i(\mathbf{x}) = g_i(\hat{\mathbf{x}}) + \nabla_\mathbf{x} g_i(\hat{\mathbf{x}})^T\,\delta\mathbf{x} + \frac{1}{2}\,\delta\mathbf{x}^T\,\nabla_{\mathbf{x}\mathbf{x}} g_i(\hat{\mathbf{x}})\,\delta\mathbf{x} + \varepsilon(\|\delta\mathbf{x}\|^3) \quad \forall i = 1,\ldots,m. \tag{B.3b}$$

Thus, the linear approximation of the vector-valued function $\mathbf{g}(\mathbf{x})$ in vector form is given by

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}(\hat{\mathbf{x}}) + \mathbf{g}_\mathbf{x}(\hat{\mathbf{x}})\,\delta\mathbf{x} = \mathbf{g}(\hat{\mathbf{x}}) + \nabla_\mathbf{x}\mathbf{g}(\hat{\mathbf{x}})^T\,\delta\mathbf{x}. \tag{B.3c}$$

B.2 Continuity

The smoothness of a path can be assessed considering its continuity. A curve is called $\mathcal{C}^1$-continuous or of class $\mathcal{C}^1$ if its first derivative exists and is continuous. However, its curvature trajectory is discontinuous. In contrast, a $\mathcal{C}^2$-continuous path has also a continuous second derivative thus curvature. These relations are illustrated in Fig. B.1 for a spline of class $\mathcal{C}^1$ and $\mathcal{C}^2$, respectively.

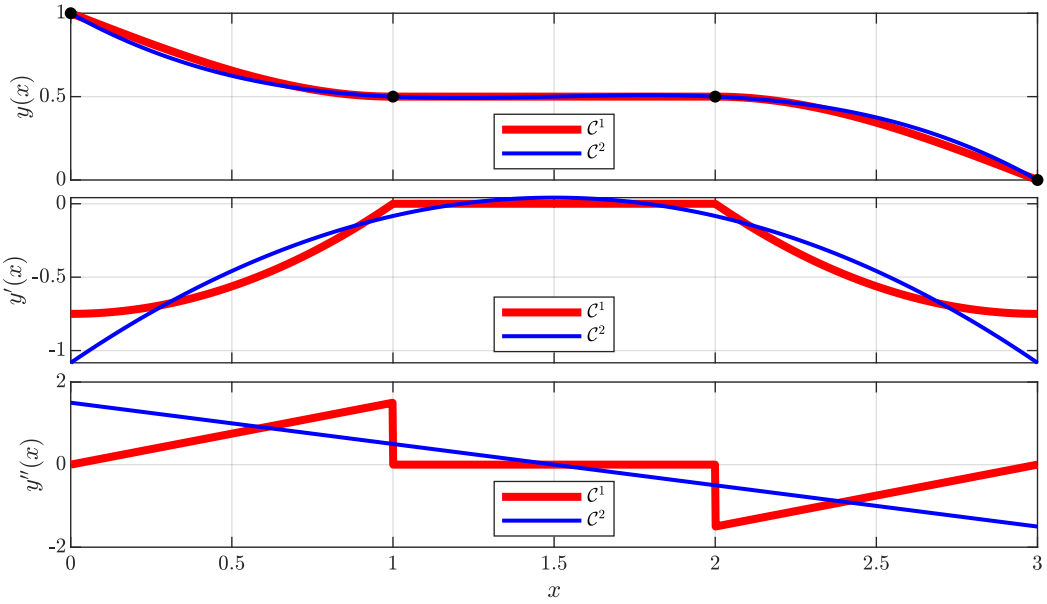


Figure B.1: Trajectory with corresponding first and second derivative for $\mathcal{C}^1$-continuous and $\mathcal{C}^2$-continuous curve.

B.3 Linearisation of Absolute Value Inequality Constraint

The SSC procedure presented in Section 5.3 requires the linearisation of a scalar absolute value constraint:

$$y \geq |x - x_0| =: f_{\text{abs}}(x). \tag{B.4a}$$

The linearisation of the right-hand side $f_{\text{abs}}(x)$ around a point $x = \hat{x}$ is shown below:

$$\begin{aligned}
f_{\text{abs}}(x) &\approx f_{\text{abs}}(\hat{x}) + \frac{\mathrm{d}}{\mathrm{d}x} f_{\text{abs}}(x)\Big|_{x=\hat{x}} (x - \hat{x}) \\
&= |\hat{x} - x_0| + \operatorname{sign}(\hat{x} - x_0)\,(x - \hat{x}) \\
&= \operatorname{sign}(\hat{x} - x_0)\, x + \Big(|\hat{x} - x_0| - \operatorname{sign}(\hat{x} - x_0)\,\hat{x}\Big) \\
&= \operatorname{sign}(\hat{x} - x_0)\, x + \left(\frac{\hat{x} - x_0}{\operatorname{sign}(\hat{x} - x_0)} - \operatorname{sign}(\hat{x} - x_0)\,\hat{x}\right) \\
&= \operatorname{sign}(\hat{x} - x_0)\, x + \Big((\hat{x} - x_0)\operatorname{sign}(\hat{x} - x_0) - \operatorname{sign}(\hat{x} - x_0)\,\hat{x}\Big) \\
&= \operatorname{sign}(\hat{x} - x_0)\ (x - x_0)\,.
\end{aligned} \tag{B.4b}$$